POWERING AND EVALUATING DEEP LEARNING-BASED SYSTEMS USING GREEN ENERGY

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea Politehnica Timişoara
în domeniul Calculatoare și Tehnologia Informației
de către

ing. **Sorin Liviu JURJ**

Conducător ştiinţific: Prof.em.dr.ing Mircea VLĂDUȚIU
Referenţi ştiinţifici: Acad. Mircea PETRESCU – UPB București
Prof.dr.ing. Liviu MICLEA – UTC Cluj-Napoca
Prof.dr.ing. Nicolae ROBU – UPT Timișoara

Ziua susţinerii tezei: 09.07.2020

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor
12. Ingineria sistemelor
13. Inginerie energetică
14. Calculatoare şi tehnologia informaţiei
15. Ingineria materialelor
16. Inginerie şi Management

Universitatea Politehnica Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul Şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
Tel./fax 0256 403823
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Teza de doctorat a fost elaborată pe parcursul activităţii mele în cadrul Departamentului de Calculatoare și Tehnologia Informației al Universităţii Politehnica Timişoara.

Mulţumiri deosebite se cuvin conducătorului de doctorat prof.em.dr.ing. Mircea Vlăduțiu, în primul rând, pentru că a crezut în mine și m-a încurajat să urmez un doctorat încă din perioada studiilor mele de licență în Inginerie Calculatoare. Încă de la început, el a modelat modul în care percepeam, defineam și rezolvam o problemă, precum și modul în care răspândeam noile cunoștințe în articolele mele de cercetare, în rapoarte cât și în prezenta teză de doctorat. Experiența vastă a profesorului Mircea Vlăduțiu a făcut ca sfaturile sale să fie nu numai constructive, sincere și practice, ci și orientate spre succes. Am fost foarte norocos și mă simt onorat să am un conducător de doctorat atât de grozav în această călătorie.

De asemenea doresc să mulțumesc sincer lui Flavius Opriţoiu. După ce am venit la Universitatea Politehnica din Timișoara, am avut ocazia să primesc îndrumările și sfaturile sale referitoare la predarea materiei „Arhitectura Calculatoarelor” în laboratorul Advanced Computer Systems and Architectures (ACSA), precum și referitoare la cercetarea mea în domeniul de Deep Learning și cel de testare hardware și software. Problemele întâmpinate în timpul studiilor mele de doctorat au fost ușor rezolvate datorită sprijinului său.

Aș dori, de asemenea, să le mulțumesc lui Lucian Prodan, Mihai Udrescu, Alexandru Topîrceanu, Alexandru Iovanovici și Raul Rotar de la laboratorul ACSA pentru sfaturile și suportul lor deosebit în timpul studiilor mele de cercetare. A fost o onoare să fac parte din echipa ACSA încă de la început.

Și sunt recunoscător celei care mi-a fost mai mult decât o familie, Dita Merkler-Poenaru. Nu voi uita niciodată sprijinul uimitor pe care mi l-ai oferit întotdeauna, precum și toate sfaturile educaționale. Tu ai fost mereu modelul meu în societate. Îți mulțumesc pentru toată dragostea și prezența ta în viața mea.

Aș mulțumi, de asemenea, prietenului meu, Franz Löer, că a fost mereu acolo pentru mine cu sfaturi deosebite, în special celor legate de comunicare, precum și pentru tot sprijinul acordat.

Mulțumiri speciale prietenei mele Dipty Sharma, pentru toată energia pozitivă, dragostea, râsul, mâncarea delicioasă, încurajările și sprijinul acordat pe toată durata studiilor mele de doctorat. Am fost binecuvântat că te-am întâlnit.

În cele din urmă, aș dori să le mulțumesc tuturor profesorilor și studenților din întreaga lume care fac tot posibilul pentru a menține viu spiritul de cercetare și care, prin ideile lor inovatoare, transformă societatea noastră într-un loc mai creativ, mai pașnic, mai prietenos și mai deschis pentru toată lumea.

Timişoara, Iulie 2020 ing. Sorin Liviu Jurj

Jurj, Sorin Liviu

"Powering and Evaluating Deep Learning-based Systems using Green Energy"

Teze de doctorat ale UPT, Seria X, Nr. YY, Editura Politehnica, 2020, 196 pagini, 91 figuri, 39 tabele.

Cuvinte cheie: deep learning, green energy, solar energy, metrics, solar tracker, hardware testing, software testing, hash algorithms

Rezumat: În ultimii ani, progresele din domeniul inteligenței artificiale, în special în ceea ce privește algoritmii de învățare profundă, au crescut într-un ritm rapid și vor continua această tendință pentru anii următori. De la implementări hardware până la software, pentru a integra acești algoritmi inspirați de creier în fiecare aspect al vieții noastre, studii de cercetare active sunt realizate în diferite industrii. Cu toate acestea, datorită faptului că acești algoritmi necesită o cantitate mare de timp, energie, date și putere de procesare, impactul lor asupra mediului este o problemă definitorie.

Pentru a rezolva această problemă, în prezenta teză de doctorat construim și testăm la nivel software și hardware un tracker solar cu două axe pe care îl folosim ca sursă autonomă de energie curată pentru un sistem de învățare profundă care clasifică imagini în timp real. Apoi, propunem patru metrici pentru evaluarea performanței modelelor și sistemelor de învățare profundă bazate nu numai pe precizia acestora, ci și pe consum de energie și cost, după care implementăm o aplicație care oferă posibilitatea oricărui utilizator de a folosi metricile propuse într-o interfață prietenoasă și rezolvă probleme legate de colectarea, curățarea și etichetarea datelor necesare pentru antrenarea modelelor de învățare profundă.

În cele din urmă, am construit și un dispozitiv pentru testarea plăcilor de circuite imprimate, care este eficient în ceea ce privește precizia, timpul de testare, consumul de energie și costul, precum și am propus un set de tehnici pentru îmbunătățirea performanțelor de transfer a unei implementări hardware Secure Hash Algorithm-256.

Abstract

In recent years, advancements in the field of Artificial Intelligence, especially regarding Deep Learning algorithms, grew at a rapid pace and will continue this trend for the years to come. From hardware to software implementations, active research studies are conducted across different industries, in order to integrate these brain-inspired algorithms in every aspect of our life. However, due to the fact that these algorithms require a huge amount of time, energy, data, and processing power, their impact on the environment is a defining issue. To solve this problem, considering recent „Green AI" efforts that focus on the energy efficiency of AI systems, we propose four novel environmentally-friendly metrics for evaluating the performance of Deep Learning models and systems based not only on their accuracy but also on their energy consumption and cost.

The current Ph.D. thesis begins by implementing Deep Learning image classification applications that solve problems related to fraud and security. By observing the huge amount of energy consumption and cost as well as the amount of time needed for data curation, we decide to solve these problems using hardware and software approaches. For this, we first build and improve a dual-axis solar tracker which we use to successfully power a real-time Deep Learning-based system. In order to minimize the operation costs of the proposed dual-axis solar tracker and make sure that we will be notified as soon as there is a possible malfunction, we implemented hardware and software testing methods for detecting possible faults that can appear during its operation. Secondly, we implemented a Computer Vision application that not only solves the problem related to data collection, cleaning, and labeling with the help of Deep Learning but also offers the possibility for anyone to use our proposed metrics in a user-friendly interface.

Finally, we also implemented an affordable and sensorless Flying Probe-inspired In-Circuit-Tester for testing Printed Circuit Boards which is efficient regarding precision, test time, power consumption and cost as well as a set of techniques for improving the throughput performance of a Secure Hash Algorithm-256 hardware implementation.

Acknowledgments

First of all, I would like to thank my Ph.D. advisor, Professor Mircea Vlăduțiu, for believing in me and encouraging me to pursue a Ph.D. ever since my Bachelor's degree in Computer Engineering. From the very beginning, he shaped the way I was perceiving, defining and solving a problem as well as spreading the new knowledge in my research articles, Ph.D. Reports and lastly, the present Ph.D. Thesis. The vast experience of Professor Mircea Vladuțiu made his advice be not only constructive, sincere and practical, but also success-oriented. I have been very lucky and feel humbled to have such a great advisor in my Ph.D. journey.

I give my sincere thanks to Flavius Opriţoiu. After coming to the Politehnica University of Timisoara, I had the opportunity to receive his guidance and advice related to teaching Computer Architecture in the Advanced Computing Systems and Architectures (ACSA) laboratory as well as researching in the field of Deep Learning and Hardware and Software Testing. The challenging problems encountered during my Ph.D. studies were easily solved because of his effortless support.

I would also like to thank Lucian Prodan, Mihai Udrescu, Alexandru Topîrceanu, Alexandru Iovanovici and Raul Rotar from the ACSA laboratory for their great advice and support during my research studies. It has been an honor to be part of the ACSA team since the very beginning.

And I am grateful to the one who was more than family to me, Dita Merkler-Poenaru. I will never forget the amazing support you always gave me as well as all the educational advice. You set me a great role model. Thank you for your love and presence in my life.

I would also thank my friend, Franz Löer, for being there for me with great advice, especially the ones related to communication, as well as for all the support.

Special thanks to Dipty Sharma, for all the positive energy, love, laughter, delicious food, encouragements and support during my Ph.D. studies. I have been blessed to have met you.

Finally, I would like to say thank you to all the professors and students around the world who give their best to keep the research spirit alive and who, through their innovative ideas, transform our society into a more creative, peaceful, friendly and open place for everyone.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The present Ph.D. Thesis entitled „Powering and Evaluating Deep Learning-based Systems using Green Energy" describes the research activity carried on between 2016 and 2020 in the Department of Computers and Information Technology at the Politehnica University of Timisoara. Our work is mainly focused on powering Deep Learning (DL)-based systems using green energy as well as on developing environmentally-friendly metrics for DL [1] in order to evaluate the performance of DL models and systems based not only on their accuracy but also on their energy consumption and cost.

The success of DL in the last years in various Artificial Intelligence (AI) applications related to computer vision, speech recognition, machine translation and Natural Language Processing (NLP) is mostly based on the recent advances regarding computing power and the huge amount of digital data available (e.g. photos). From self-driving cars [2] and cancer detection [3] to even arrhythmia detection [4] and robotics [5], the performance of DL is outperforming that of humans, showing significant improvements related to both speed and accuracy. These improvements related to speed and accuracy are happening every year thanks to efforts from both academia as well as the industry. On one side, companies are pushing towards creating more powerful computing platforms and optimized libraries, whereas, on the other side, new specializations and jobs related to DL and Data Science are created, these being also ones of the best paid in the industry [6].

However, in order to increase the training or the inference speed of such DL models, usually, the use of a single Graphics Processing Unit (GPU) is not always enough, with some projects even requiring hundreds (i.e. 512) of power-hungry GPUs [7]. As a consequence, this results in high energy consumption and cost, which ultimately have a negative impact not only on the financial side but also on the environment [8, 9, 10], with the work in [9] even proving that the training of a single NLP model results in a massive carbon footprint equivalent to the amount of carbon emission that five cars have in their lifetime. Additionally, the evaluation of DL models and systems is mainly done with traditional metrics such as accuracy or throughput, without considering their energy consumption or cost.

Another key challenge, especially for data scientists is the time spent preparing the data. In order to increase the accuracy of DL models, the most amount of work time goes into collecting, cleaning and labeling the data, reaching around 80% of the total time allocated for a DL project. This not only slows the process of AI innovation but can also be very costly, especially when the dataset is very large and many people are involved [11].

Additionally, because we as researchers owe our present knowledge due to past enormous efforts by fellow students, professors, as well as scientists who pushed the boundaries and tried their best to share their knowledge, in the domain of test engineering, especially in the case of testing Printed Circuit Boards (PCBs), the available testing devices are still very expensive [12] and non-existent in engineering-related schools or universities. This situation not only makes the delicate and important process of testing PCBs be harder to comprehend, but it can also reduce the number of possible specialists in the testing domain.

Due to the internet, the number of users online, as well as the number of electronic devices, increase year after year, thus the security not only of the information that is stored and exchanged is of great importance but also that of these hardware devices.

In this Ph.D. thesis, we developed and implemented methods for solving the above-mentioned problems by first implementing different novel DL applications that solve different problems related to fraud [13, 14] and security [15]. Then, because we observed that a real-time DL-based system [15] consumes more energy than their non-real-time counterparts, we decided to not only run the same implementation on a platform that consumes 5× less energy, but we also wanted to not pay for this energy consumption [16]. We achieved this by considering the use of green energy and by constructing a novel dual-axis solar tracker that is based on the Cast-Shadow principle [17] and which was later modified with minimal costs [16]. We demonstrated in [16] that our solar tracker is efficient and, to the best of our knowledge, for the first time in literature, that it is possible to completely use solar energy for powering a real-time DL-based system when running inference.

In order to be aware of any possible faults in our dual-axis solar tracking device, we also investigated possibilities to test it at the software and hardware level. For this, we implemented a novel White-Box testing technique in [18] and a novel Online Built-In Self-Test (OBIST) testing technique in [19], both achieving high fault coverage. It is important to mention that, to the best of our knowledge, testing a solar tracking equipment has also never been done before in literature.

As mentioned earlier, because we succeeded in making use of green (solar) energy for powering a DL-based system [16] and because we want to encourage future generations of researchers to consider the impact their DL project can have on our environment, we proposed four novel DL metrics [20] that evaluate the performance of DL models and systems for both inference and training by taking into consideration not only the accuracy but also the energy consumption and cost, proving to be more valuable metrics when compared with the existent ones found in the literature. We also created a Computer Vision application [21] that incorporates the four proposed metrics and offers an easy way to calculate and evaluate the performance of DL-based systems. Additionally, the application consists of multiple features that make use of DL inference in order to speed-up tasks related to data collection, cleaning, and labeling, outperforming existent solutions by a large margin.

Additionally, in order to offer engineering students a chance to have a hands-on approach for testing PCBs, we implemented a low-cost and portable PCB testing device in the form of a sensorless Flying Probe-Inspired In-Circuit-Tester (FPICT) [22] that has a high fault coverage and a very low cost.

Finally, in order to increase the throughput performance of a Secure Hash Algorithm (SHA)-256, we implemented different techniques to speed-up the hash generation in hardware [23].

1.1. Motivation

The main philosophy behind this Ph.D. thesis is to make the evaluation of DL models and systems possible by using environmentally-friendly metrics. Additionally, to significantly reduce the amount of energy consumption and cost of DL-based systems as well as to reduce the time needed to collect, clean and label image datasets needed for training DL models.

The motivation for Implementing Different DL Applications Related to Fraud and Security: First, the demand for transparency, especially regarding money transactions in a Supermarket where the price tag of the products doesn't always reflect the price seen on the receipt and customers end up paying the wrong price because of incorrect prices on the receipt. Also, on-device inference with high accuracy by building our own Optical Character Recognition (OCR) algorithm which is resistive to noise eliminates the need for expensive and commercial cloud-based solutions.

The second reason is the necessity of preventing cultural appropriation by big brands from the fashion industry. By implementing a DL-based algorithm, we can automate the detection and recognition of traditional motifs with high accuracy and reduced processing time.

The third reason is security and the need for automatically classifying and identifying wild and domestic animals present in an area, in order to increase their safety as well as that of humans. With the help of DL algorithms, non-intrusive identification systems can be developed that can perform animal classification in real-time from videos or using a webcam, being of great help, especially for researchers and farmers.

The fourth reason was energy consumption and cost. Running these applications in different platforms results in different power consumptions and costs which ultimately affect the user and the environment.

The motivation for Building, Testing and Deploying a Dual-Axis Solar Tracker to Power a Self-Sufficient Real-Time DL-based system: First, because most of the existent solutions for powering DL-based systems harm the environment and the urgent need of alternative energy sources such as green energy is crucial. Capturing the maximum potential of solar energy by using a dual-axis type of solar tracker that uses no sensors and which, with the help of a blocking system reduces the power consumption of the entire solar tracking equipment and maximizes its energy gains results in a low-cost and portable solution that eliminates the carbon footprint on our environment by using green (solar) energy instead of traditional polluting power sources.

The second reason was the energy cost. By using solar energy, there are 0 electricity costs. Also, by testing a solar tracker at the software and hardware level, a high fault coverage can be achieved, at the same time maintaining low costs and efficient testing solutions that can minimize also operation costs.

The motivation for Proposing Environmentally-Friendly Metrics and Implementing a DL-based Computer Vision Application: First, existent metrics based only on accuracy to evaluate the performance of DL-based systems ignore the economic, environmental and social costs. It is of crucial importance to use environmentally-friendly metrics for evaluating DL models and systems in order to mitigate climate change.

The second reason is energy consumption and cost. Different hardware platforms and DL models consume different amounts of energy, thus having different costs. This is especially the case in AI populated data center workloads where the energy consumption and electricity bill costs are very expensive. By using green energy, there will be no negative impact on the environment and the cost of training or running inference will be 0.

The third reason is encouraging new researchers to use only green energy for powering their DL-based systems.

The fourth reason is the need for reducing execution time for Data Scientists. In order to train a DL image classification model, a huge amount of time

(around 80% of the entire time dedicated to a DL project) is lost when huge amounts of images need to be collected, cleaned and labeled. Thus, a DL-based application with a user-friendly interface is very helpful, being time and costs-efficient. especially with environmentally-friendly metrics calculators.

The motivation for Implementing a Flying Probe-inspired In-Circuit-Tester: First, recent efforts in bringing affordable and equal access to education seen on the United Nations (UN) agenda, one example being the UN Sustainable Development Goals.

The second reason is the inexistence in the academic environment anywhere in the world of an affordable, portable and user-friendly testing device which can give students a chance to have a hands-on experience with the inner workings of a Flying Probe Testing (FPT) and the real parameter values of a PCB.

The third reason is the cost of In-Circuit-Tester (ICT) versions found in the industry, which are very expensive.

The motivation for Implementing an SHA-256 in Hardware: First, the need for security, e.g. for servers that offer services based around Internet Protocol Security (IPsec) and Secure Sockets Layer (SSL)/Transport Layer Security (TLS) and which rely on fast computation when updating hash values. This is especially important for DL-based systems that store or exchange sensitive information that requires confidentiality, e.g. medical data.

The second reason is the execution time. By implementing different throughput improvement techniques of the SHA-256 in hardware instead of software, we relieve the Central Processing Unit (CPU) from latencies that can occur, thus optimizing the clock cycle usage.

1.2. Contribution and Ph.D. Thesis Outline

We reduce the energy consumption and eliminate the electricity costs of a real-time DL-based system by using solar energy. We also propose environmentally-friendly metrics for performance evaluation of DL models and systems. Additionally, we reduce the time needed for collecting, cleaning and labeling image datasets. Finally, we improve the operation costs and fault coverage of the built electrical equipment (dual-axis solar tracker and FPICT) as well as the throughput of a hash algorithm hardware implementation. The contributions of this Ph.D. Thesis are summarized in Fig.1.1 and are as follows:

- We developed a receipt fraud detection method by using two Convolutional Neural Networks (CNNs) models which can recognize multiple digits with decimals from pictures taken by a smartphone camera. The proposed solution can run on-device and proves to be noise-resistant, being able to detect receipt fraud by identifying and comparing the prices of the products from the shelf with the prices of the products from the final receipt given by the cashier in the Supermarket with an overall test accuracy of more than 99%
- We developed a DL model and implemented a webcam-based system that can detect and identify Romanian traditional motifs found on 4 categories (clothes, ceramics, carpets, and painted eggs). By using transfer learning, we achieve 99.4% overall test accuracy and reduced webcam processing time

- We developed an ecology-oriented system that can classify 34 classes representing the most popular species of animals found in domestic areas of Europe in real-time from videos or using a webcam, with an overall test accuracy of 94.5% for the best (i.e. MobileNetV2) out of four trained CNN architectures (VGG-19, InceptionV3, ResNet-50, and MobileNetV2) and which can also generate 2 new datasets in real-time, one dataset containing textual informations (animal class name, date, time) and one dataset containing images of the identified animal classes
- We implemented a position optimization method for a solar tracker by using the Cast-Shadow principle and verifying the software code that runs on Arduino UNO. With the help of a novel approach that makes use of limit switches and blocking elements, we reduce the overall power consumption of the autonomous solar system by 86.93%. When compared to a static solar panel, our method shows a 45.77% voltage, 48.21% current, and 53.62% power increase, resulting in an efficient solution
- We implemented a White-Box testing technique that tests the software code running on a Wi-Fi module of a solar tracker but is also capable of giving the operator the ability to remotely control the stepper motors movements of the solar tracker. This Wireless-Based Software Technique (WBST) achieves a total coverage of 70.12% for all targeted errors, resulting in an efficient and low-cost testing solution
- We implemented an OBIST architecture for testing a solar tracking device for possible hardware faults during its normal operation by identifying its inactive mode (resting state) using an idle state detector. The hardware implementation and software simulation achieve an average of 93.93% coverage for single bit-flip errors (last 8 bits, mutant), 100% coverage for single stuck-at-faults (8, 12 and 16 random bits) as well as 96.96% for all targeted faults, showing that the proposed OBIST architecture is efficient with regard to test coverage and cost points of view
- We implemented a self-sufficient solar-powered real-time DL-based system that runs inference 100% on solar energy and which is composed of an Nvidia Jetson TX2 board and a dual-axis solar tracker based on the Cast-Shadow principle. In order to lower the power consumption, a software motion detection method is also implemented that triggers the inference process only when there is substantial movement in the webcam frame. Experiments prove that real-time DL-based systems can be powered by solar trackers without the need for traditional power plugs or need to pay for electricity bills
- We introduce four novel DL metrics, two regarding inference called Accuracy Per Consumption (APC) and Accuracy Per Energy Cost (APEC) and two regarding training called Time to Closest APC (TTCAPC) and Time to Closest APEC (TTCAPEC), which take into account not only a DL model's accuracy but also its energy consumption, energy cost and the time it takes to train it up to that point. Experimental results prove that all four DL metrics are efficient in benchmarking, encouraging future DL researchers to

adopt and use only green energy when powering their DL-based systems

- We implemented a DL-based Computer Vision application with multiple built-in Data Science-oriented capabilities, mainly for image classification tasks that are able to automatically: a) gather images needed for training DL models with a built-in search engine crawler; b) remove duplicate images; c) sort images using built-in pretrained DL models or user's own trained DL model; d) apply data augmentation; e) train a DL classification model; f) evaluate the performance of a DL model and hardware by using an accuracy calculator as well as the APC, APEC, TTCAPC and TTCAPEC metrics calculators. Experimental results show that the proposed Computer Vision application has several unique features and advantages, proving to be efficient regarding execution time and much easier to use when compared to similar applications. The motivation behind creating this tool was regarding obtaining our experimental results and to also make it available to the scientific community
- Additionally, we also implemented a hybrid sensorless ICT design by combining the features of FPT and the capabilities of a Coordinate Measuring Machine (CMM). The experimental results show that the proposed FPICT is suitable for smaller sized PCBs and proves efficient regarding precision (overall precision of 95.70% for the measurements testing), test time (an average of 10.35s for a single-point test cycle), power consumption (an overall of 3.92W for all considered test cases) and cost (around 25 dollars) points of view being much more affordable and user-friendly when compared to traditional and expensive FPTs found in the industry
- Finally, we also developed hardware acceleration techniques for an SHA-256 algorithm which resulted in up to 18% throughput improvement. The first acceleration technique eliminates one clock cycle used for hash value update, delivering a higher throughput. The second techniques improve the performance by fusing the Carry Propagate Adders (CPAs) of the multi-operand adders to speed up the generation of the round functions. The third technique has a synthesis driven approach that improves the delay balancing in the Carry Save Adder (CSA) tree, resulting in a reduced critical path

Chapter 2 presents the theoretical background for a better understanding of the research papers that comprise this Ph.D. Thesis and which are presented starting with chapter 3. We present some of the neural network architectures, frameworks, and analysis of datasets for different DL applications. We also cover a section related to hardware and software testing which describes different off-line and on-line testing methods we used. Finally, the related works regarding our DL applications, hardware, and software testing, low-power hardware platforms, metrics, data collection, and labeling as well as regarding hardware implementations of SHAs are also presented.

Chapter 3 presents 3 different DL image classification applications: a) a novel method in detecting receipt fraud by using a smartphone application that makes use of an OCR algorithm composed of image processing techniques and CNNs. The proposed method successfully detects prices from product price tags as well as receipts with high accuracy. Additionally, the proposed CNN models

outperform other popular open-source OCR algorithms regarding test accuracy on images with cropped Product and Receipt prices that contain noise; b) a novel method in identifying Romanian traditional motifs found on 4 categories (clothes, ceramics, carpets, and painted eggs) using CNNs.

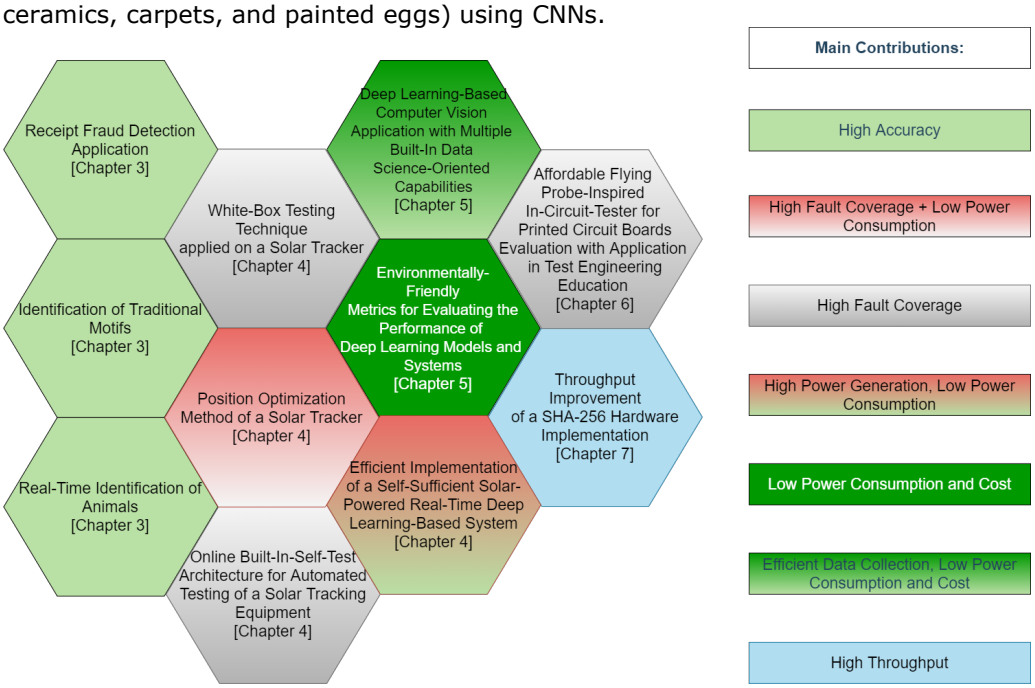


Fig. 1.1. Ph.D. thesis contributions (summarized view).

We also implemented a system that can detect and identify these learned motifs through a webcam with high accuracy and reduced processing time; c) a novel method of identifying animals that belong to the 34 most popular species found in domestic areas of Europe. We implemented a system that can identify these species in images, videos or through a webcam and generate 2 new datasets in real-time, one containing textual information about the animal present in front of the webcam, and one containing images of the identified animal species. Our method has several advantages compared with other related works. This chapter's contents are mainly based on our works in [13-15].

Chapter 4 presents the construction, testing, and deployment of a dual-axis solar tracker in order to power a real-time DL-based system. More exactly: a) a testing technique in verifying the software code that runs on an Arduino UNO microcontroller which optimizes the position of a solar tracking device by resorting to novel elements such as limit switches and blocking elements. Our software method contributed to a significant reduction in power consumption and proved the efficiency of automated versions of solar panels over static ones; b) a novel White-Box Testing technique applied on a solar tracker which tests the software code that runs on a NodeMCU Lua ESP8266 Wi-Fi module and proves that is effective from the point of view of fault coverage and cost. Additionally, we gain the ability to control directly the stepper motor movements of the autonomous solar tracker in a wireless manner; c) a hardware testing technique that makes use of an OBIST which intervenes in testing the electrical equipment of a solar tracking device for possible hardware faults, aiming to minimize the operation costs and being efficient

regarding test coverage; d) a novel method in powering a real-time DL-based system using 100% green energy by using an Nvidia Jetson TX2 embedded platform and an improved dual-axis solar tracker that was connected to a chain of two inverters, one accumulator and one solar charge controller. Our software implementation modifications help in detecting the optimum GPU memory usage and frames-per-second (fps) to run our DL models without any risk of „out of memory" kind of errors and together with a software motion detection method, we succeed to reduce the energy consumption of the entire DL-based system. This chapter's contents are mainly based on our works in [16-19].

Chapter 5 presents the proposed environmentally-friendly metrics for DL as well as a Computer Vision application with multiple built-in Data Science-oriented capabilities. More exactly: a) the four novel APC, APEC, TTCAPC and TTCAPEC metrics for evaluating the performance of DL models and systems not only regarding the accuracy but also their energy consumption and cost, showing that green energy-powered DL-based systems are evaluated as being much more performant compared to existent ones that still use a traditional power grid; b) an application with a user-friendly interface that solves many problems related to data curation and which offers an easy way to evaluate the performance of DL-based systems with the APC, APEC, TTCAPC and TTCAPEC metrics calculators. This chapter's contents are mainly based on our works in [20, 21].

Chapter 6 presents an affordable, portable and user-friendly FPICT that has educational purposes in the domain of test engineering, mainly for testing smaller sized PCBs such as Arduino Uno without the need for sensors. The FPICT can easily be connected to any computing platform that has a USB port and its C written configuration files can easily be modified, providing students easy access to study and experiment with the inner workings of an FPT when operating on a real PCB board. This chapter's contents are mainly based on our work in [22].

Chapter 7 presents several acceleration techniques for improving the throughput of SHA-256 hardware implementation. First, the throughput acceleration technique eliminates one clock cycle used for hash value update and allows delivering a higher throughput. Also, the critical path of a CSA tree structure is considerably reduced by using a fast 32-bit Kogge-Stone adder. With the second technique, we evaluated alternative multi-operand addition structures and implemented the CPAs of the multi-operand adders in a fused manner to speed up the generation of the round functions. The synthesis driven approach for arranging the operands' order (delay balancing improvement) in the CSA tree further reduce the critical path and show that our solution resonates with the increasing demands for a more secure biometric implementation. This chapter's contents are mainly based on our work in [23].

Chapter 8 presents the conclusions of this Ph.D. thesis.

2. THEORETICAL BACKGROUND

In this chapter we will introduce what DL is and the way it works, followed by its applications in the real world. Then, we will introduce the Deep Neural Network (DNN) architectures, the datasets and the DL frameworks we used in order to be able to train and run inference with the proposed DL models. Also, an introduction to green energy is made, especially regarding solar energy and of the dual-axis solar tracking devices that are used to gather this type of energy. After that, we will also introduce the different types of hardware and software testing methods as well as the hashing algorithm we used. Finally, we will present the related works with regard to DL, test engineering as well as regarding hardware implementations of SHA-256 algorithms.

2.1. About Deep Neural Networks

Today's computer architecture is significantly different regarding processing capabilities, organizational structure, and power requirements when compared to the human brain. Because of the approaching end of Moore's law, the feasibility of creating an alternative architecture that is brain-inspired was put in question [24]. This resulted in a pursuit that caused important discoveries in the fields of AI, Machine Learning (ML), Artificial Neural Networks (ANNs), especially in DNNs, due to their high number of hidden layers or neurons that have the ability to train themselves without the need to be specifically programmed. How DL relates to AI can be seen in Fig.2.1.

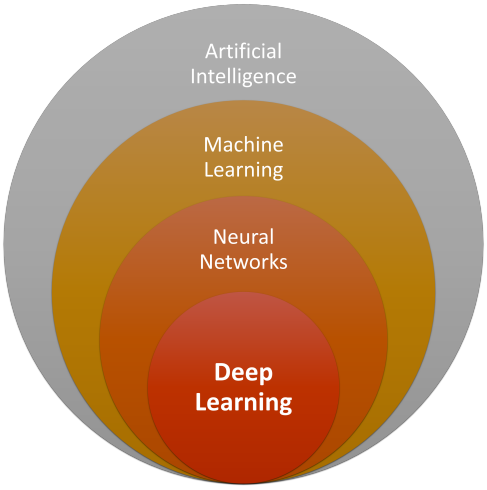


Fig. 2.1. DL in the context of AI.

The methods used in training these ML type of algorithms are divided into supervised (classes are known before training) and unsupervised learning (the

computer itself must determine the classes). DL is currently considered to be state-of-the-art virtually in all AI-related applications, mostly because it speeds the incremental advances in a field, e.g. advances that used to take years to achieve, are now happening much faster [25]. Neural networks are formed of a group of neurons and connections that are organized in layers in order to solve ML tasks. In order to produce an output, a neuron receives many inputs from predecessor neurons that are summed up by their weights followed by an activation function which is usually nonlinear, e.g. the Rectified Linear Unit (ReLU) [26]. Weights are very crucial for ANNs because this is how neural networks learn and depending on their values, the activation function will pass or not the signal to the neuron's output. This adjustment of weights during the learning process is what we call training. The neurons that have no predecessor are known as input neurons and the neurons that have no successor are known as output neurons. In between the input and output neurons, hidden layers of neurons are to be found that are connected to each other by their connections, also called synapses. It is not a rule, but it is considered that when the number of hidden layers are more than eight, it is called a DNN.

DNNs can have not only a few but also hundreds of layers [27] and in order to train them, a common technique called Gradient Descent, e.g. Stochastic Gradient Descent (SGD) [28] is used. Gradient Descent is an optimization method based on a back-propagation algorithm which calculates loss function's gradient. A very important step in calculating the gradient is to first calculate the activation of each layer during the inference, as can be seen in Fig.2.2.

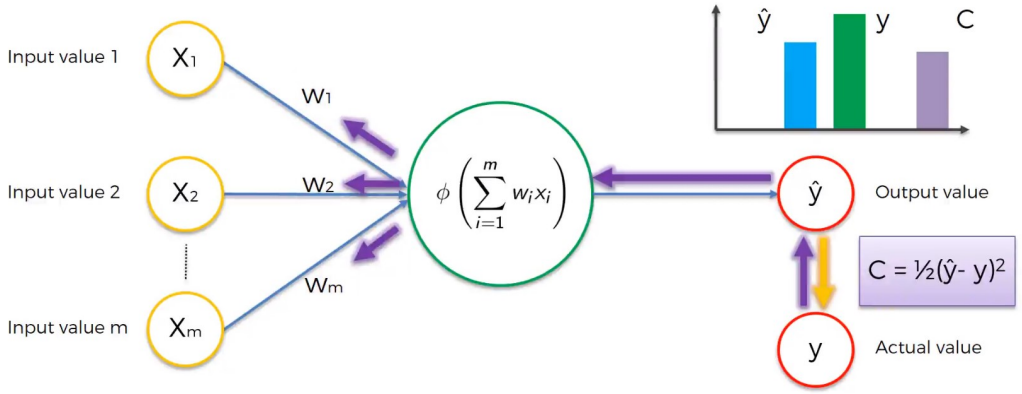


Fig. 2.2. Example of how an ANN learns by minimizing the cost function.

During inference, the value can be either continuous, e.g. regression problems, or it can be discrete, e.g. in classification problems. This feed-forward, back-propagation, and update of the weights using the gradient descent constitute one training iteration. During the training of a DNN, hundreds of iterations are needed, e.g. ResNet-50 [27] takes more than 450 thousand iterations on ImageNet [29], the dataset used by many researchers in the ILSVR (ImageNet Large Scale Visual Recognition) challenge [30].

DNNs are significantly improving many AI applications including computer vision [31], speech recognition [32], gaming [33, 34], self-driving cars [2], cancer detection [3], arrhythmia detection [4] and robotics [5], to name only a few, resulting in a rapid improvement of performance regarding the accuracy also on

ImageNet challenge [30]. An example of how powerful DNNs are, is the success of AlphaGo, the first program to achieve better performance than human players in the ancient game of Go [33] which was, shortly after, surpassed by AlphaGo Zero [34]. While the accuracy surpassed the human's one, the high power consumption and cost behind these computations have a negative impact on the environment. This brings the urgent need of research that focuses on how to not only minimize the carbon footprint and energy needs of DL systems but also on how to correctly evaluate DL models and systems using envinronmentally-friendly metrics.

2.1.1. Deep Neural Network Architectures

This section presents an overview of different types of neural network architectures that we extensively experimented within our research presented in this Ph.D. thesis.

A widely used neural network arhitecture, especially in Computer Vision, is a Convolutional Neural Network (CNN), mainly because of their ability to efficiently extract features from images and classify them. CNNs are typically consisting of 2D convolutional (CONV) layers, pooling layers and classifier layers which take the flattened output of the previous layers as input. The CONV layer creates a specified number of feature maps with the help of feature detectors and applies an activation function in order to increase the non-linearity. By doing this, the CNNs are able to learn the internal feature representations and preserve the spatial relationship between pixels. The purpose of the CONV and the pooling layers is to assure spatial invariance (map retention characteristics needed for classification via translation and rotational invariants) [35] and to significantly reduce the size of the images (by reducing the resolution of the feature maps). Pooling also gets rid of the features that are not important (features we are not looking for), resulting in a reduced computation cost. The neural network will then combine the features extracted by the CONV layers into more attributes that can predict the classes even better. This is done using a Dense layer which contains neurons that are connected to all the neurons in the prior layer. Additionally, because during the training of CNNs neurons develop co-dependency amongst each other, it can result in decreased detection power of each neuron and can lead to overfitting. In order to prevent overfitting and regularize a neural network, a technique called Dropout [36] is used on fully-connected (FC) layers. The Dropout layer deactivates (zeros out) a certain amount (e.g. 50%) of neurons randomly at each update during training time and forces the neural network to learn features in a robust manner. This is needed especially when the neural network is big in size, is training for too long or if there is insufficient data. Another method to regularize and prevent overfitting of a neural network is called Batch Normalization (BN) [37]. The BN layer is inserted between successive CONV layers and gives resistance to the vanishing gradient problem by reducing the training time.

In 2014, an architecture called VGGNet [38] was presented by researchers from Visual Geometry Group (VGG) at Oxford and which can be seen in Fig.2.3. The most common versions of VGG are the ones with 16 (VGG-16) and 19 (VGG-19) layers. In our research we experimented mostly with the VGG-19 version which has 138 million parameters across 16 CONV and 3 FC layers. The VGG-19 uses 3×3 filters with stride and padding of 1 along with 2×2 max-pooling layers with stride of 2, being one of the most influential CNN architectures which proved that deeper layers can help the model learn richer feature representations. The pre-trained

model of the VGG-19 is commonly used in segmentation tasks, detection, and classification of images.

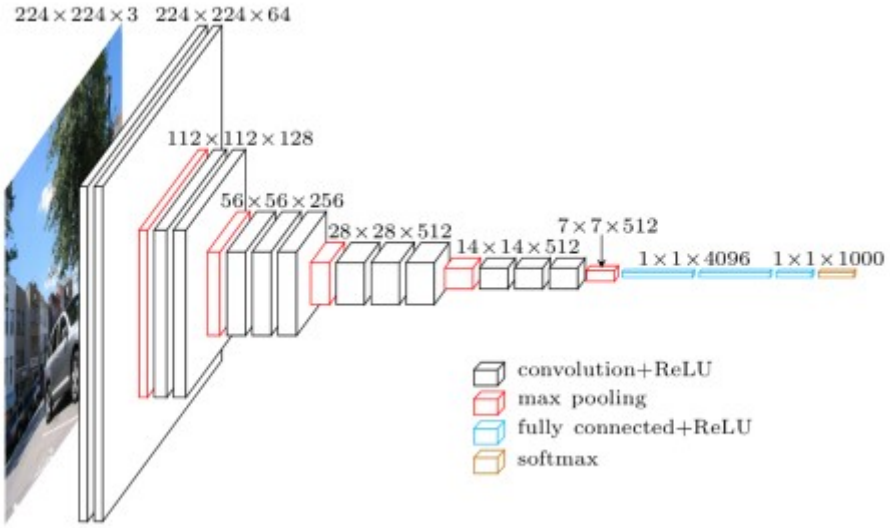


Fig. 2.3. Summarized view of a VGG-16 Architecture [25].

In the same year, 2014, a new architecture that is parameter-efficient came out, called GoogleNet or InceptionV1 [39] which can be seen in Fig.2.4. Similar to the VGGNet, despite there being many versions of Inception architectures, the most common is the InceptionV3 architecture which was proposed a year after, in 2015 [40] and which is also the version with which we experimented in our research.

The InceptionV3 architecture increased the accuracy and reduced the computational complexity of the initial version by using factorization, e.g. reducing the size of CONV parameters by replacing one 5×5 CONV by two 3×3 CONV.

In 2015, a new architecture called Residual Network (ResNet) [27] is introduced. The ResNet architecture was the winner of ILSVRC 2015 in image classification, detection, and localization, as well as winner of MS COCO 2015 [41] detection, and segmentation and can be seen in Fig.2.5. ResNet is known as the first architecture that allowed the accuracy to stay stable or increase even when having deeper layers thanks to the new concept of residual learning. The ResNet architecture has a fundamental building block (Identity) where a previous layer is merged into a future layer (additive), forcing the network to learn residuals by using a skip connection (by fitting the input from a previous layer to the next layer without any modification of the input). A popular version of ResNet is called ResNet-50, having 50 layers and consisting of more than 25 million parameters, balancing computational complexity together with prediction accuracy, this being also one of the main reasons why we used it in our research as well.

In 2017, a new mobile-friendly architecture is introduced by Google called MobileNetV1 [42] which introduces the „Depthwise Separable CONV" block (composed of a 3×3 Depthwise CONV layer that filters the input, followed by a 1×1 pointwise CONV layer that combines these filtered values to create new features by keeping the same number of channels or doublig them) to reduce the complexity (fewer multiplications and additions) and model size (fewer number of parameters).

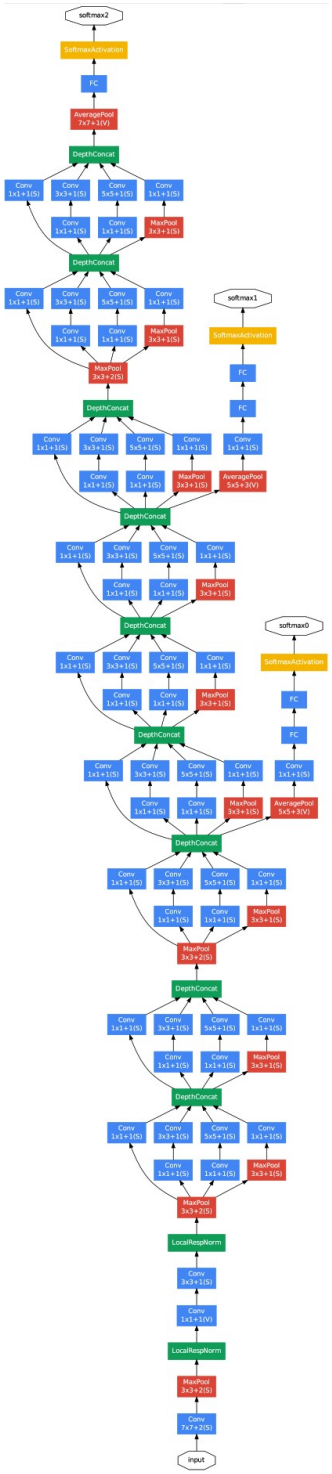


Fig. 2.4. Summarized view of a GoogleNet (Inception) Architecture [39].

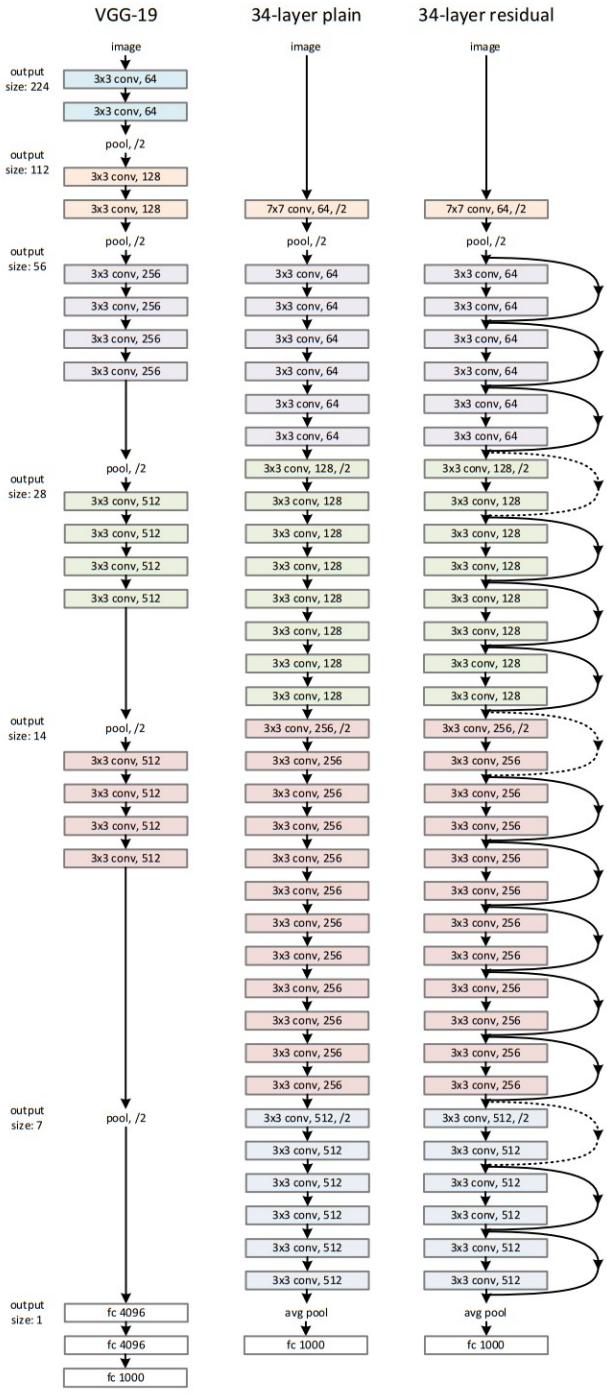


Fig. 2.5. Summarized view of a 34-layer ResNet architecture with Skip / Shortcut Connection (Right) compared to a 34-layer Plain Network (Middle) and a 19-layer VGG-19 architecture (Left) [27].

After a year, in 2018, an updated version of the MobileNet architecture is introduced called MobileNetV2 [43] which adds an extra 1×1 pointwise CONV layer also called „Projection layer" that makes the number of channels smaller, thus making this version of the architecture much smaller in size and faster than the previous one. An example of a MobileNetV1 and MobileNetV2 building block can be seen in Fig.2.6.

MobileNetV2 uses a module with inverted residual structure (instead of narrow/bottleneck layers in between wide layers of a CONV block, MobileNetV2 has wide layers in between narrow/bottleneck layers in a CONV block, resulting in fewer parameters), an expansion factor "*t*" (e.g. if the input has 32 channels, and the expansion factor *t* is 6, then the internal output will be 32×*t*=32×6=192 channels), 2 types of blocks (one with stride of 1 and another one with stride of 2 for downsizing), each MobileNetV2 block having 3 layers. More exactly, the first layer is a 1×1 pointwise CONV (combination step) with RELU6 as the activation function, the second layer is the depthwise (filtering step) 3×3 CONV and the third layer is also a 1×1 pointwise CONV (combination step), but in this case, without any non-linear function. Due to the depthwise separable CONV, which is the combination of depthwise CONV and pointwise CONV, the computation time and the number of parameters are greatly reduced.

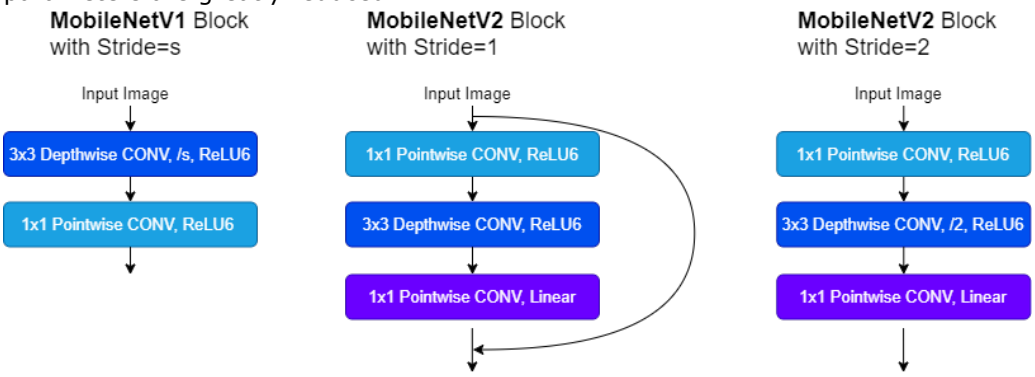


Fig. 2.6. MobileNetV1 and MobileNetV2 CONV Blocks Comparison.

In our research, we make use of MobileNetV2 architecture.

2.1.2. Analysis of Datasets for Different Applications

As mentioned earlier, the huge amount of data available today, gave researchers the possibility to train DNNs in such a way that the accuracy surpasses the human level in many tasks. In our research we use different datasets for a variety of DL applications. Most of the datasets we use are custom datasets, mostly consisting of private images as well as images scrapped from the internet for educational purposes. For this reason, we will introduce these datasets later when we will describe each implemented DL application. However, we make use also of free publicly available datasets such as MNIST [44] and ImageNet [29].

MNIST [44] is one of the most popular dataset and it consists of 70.000 images representing handwritten digits, 60.000 of them are found in the training set, and 10.000 images in the test set, organized in 10 classes, representing the 10

digits (0-9). The size of the images are 28×28 pixels, each image being greyscaled. This dataset is very good for training a model in only a few minutes, with minimum effort in data preprocessing, the entire dataset having a total size of around 50MB.

ImageNet [29] is a large-scale dataset used in the ILSVRC challenge until 2017. It consists of 1.2 million images in the training set and 50.000 images in the test set. It is a very popular dataset which helped researchers increase the accuracy of their models in classifying objects from around 72% up to 97.3% in just seven years during the ILSVRC challenge, proving that human abilities can be surpassed with the help of DL algorithms that can take better decisions when having access to bigger amounts of data. The size of the ImageNet dataset is around 150GB.

2.1.3. Deep Learning Frameworks

Through a high level programming interface, DL frameworks are useful when training and validating DNNs. They help us build algorithms of considerable complexity by abstracting the computation into simple mathematical operations, mostly found in algebra, like matrix multiplications, CONV operations, etc., and which are optimized for the hardware they run on. DL frameworks provide programmers the ability to write just a description of the computation, without the need to program a multi-core CPU or GPU directly. For training and running inference, in our research we made use of two very popular frameworks called Tensorflow [45] and Keras [46].

Tensorflow [45] is a library based on Python programming language which can run on multiple processors like CPUs and GPUs and has support for other programming languages like C/C++, Java, Go and R.

Keras [46] is wrapped around Tensorflow and is a high-level API programmed in Python. Because of its simplicity and popularity, Keras is since 2017 included in the Tensorflow framework.

2.2. Green Energy

Due to the continuous growth of the human population as well as the energy consumption, mainly in the industrial sector, the demand for more energy is expected to increase by up to 28% in the next decades [47]. Even though the current energy model was satisfying the demand for energy for many decades by making use of fossil or nuclear sources, their negative impact on the environment has resulted in many decisions to replace them with renewable energy sources in order to achieve sustainability and reliability [48].

Recent efforts made to move towards 100% clean and renewable energy infrastructure by 2050 [49] are clearly showing the growing interest towards clean and unlimited energy sources. Despite many types of renewable energy sources such as solar, wind, water, biomass, geothermal, some of these sources are not considered „green", as summarized in Fig.2.7, e.g. large hydropower can have a negative impact on land use and fisheries.

Green energy is considered to be the most environmentally-friendly type of renewable energy because it is sustainable and clean (doesn't release greenhouse gases such as CO_2, being able to mitigate the problem of climate change [10, 50]).

2.2.1. Solar Energy

Since millions of years, sun radiates enourmous amounts of energy towards our planet. More exactly, the energy flow on earth's surface is composed of more than 99.9% solar radiation that comes from the sun, making solar energy the most important source of energy our planet has. However, despite being the main life source for life on this planet, when indirectly used, solar energy can also have a negative impact on our life, e.g. because solar energy was stored for millions of years in the form of chemical energy, we developed mechanisms to extract and make use of fossil fuels such as oil, gas and coal, but with a huge health and environmental cost as a result [9, 10].

Solar energy, besides being abundant, inexhaustible, readily available and free of CO_2 emissions, it is also the only source of energy that can be transformed directly into electricity. For achieving this direct transformation into electricity, a phenomenon which is also known as the photovoltaic (PV) effect, usually solar cells made out of silicon, are used.

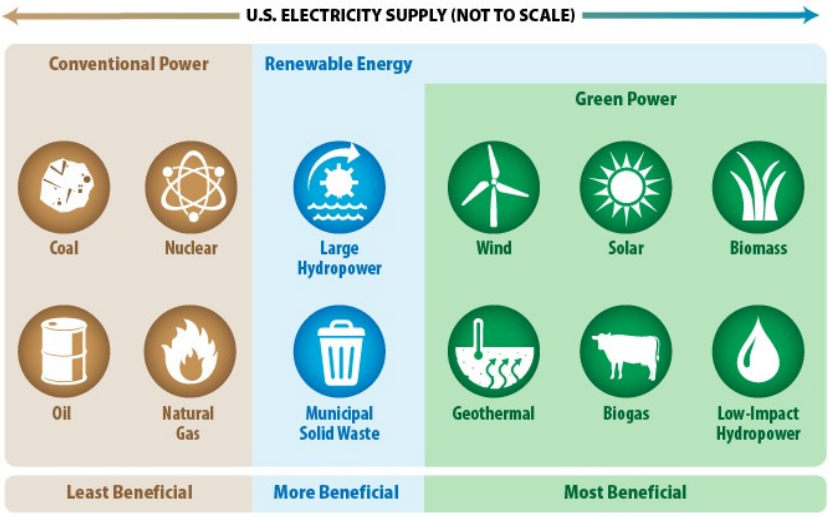


Fig. 2.7. Green power based on its relative environmental benefits [51].

2.2.2. Dual-Axis Solar Tracking Devices

Because of Sun's availability and its unlimited quantities of clean energy, recently, there is a growing interest in the academia and in the industry of developing efficient solar energy collectors. With regard to modern approaches in how energy output can be improved, two major paths can be listed: a) the Maximum Power Point Tracking (MPPT) method and b) active solar panel tracking solutions (or in other cases a mixture between both these techniques). As far as the second method is concerned, solar panel tracking solutions are a more advanced technology for mounting PV panels and appear under two models: single-axis and dual-axis solar tracking devices.

While single-axis solar trackers follow the Sun's position only in the East-West direction [52], its peer, the dual-axis model can also cover the North-South direction, hence presenting a much better option, especially for sunny and cloudy days [53]. An actual overview of solar energy technology [54] validates a 50% energy increase for Sun-tracking designs compared to fixed-tilted PV panels over the year. Nevertheless, solar tracking techniques are efficient and reliable methods that embrace a variety of applications in domains of interest such as railway transportation, AI [55, 16] and Internet of Things (IoT) [56].

2.3. About Hardware and Software Testing

This section presents an overview over some of the on-line (concurrent) as well as off-line (non-concurrent) testing methods existent in the literature and which are also used in our research, either when testing our solar tracking equipment [17] at the hardware [19] and software [18] level or when developing the FPICT device [22].

The non-concurrent nature of off-line testing significantly minimize the use of hardware overhead and can be designed to cover almost if not all of the Circuit Under Test (CUT) area as possible [57]. This is because, in order for the off-line test mechanism to allow a test process to be executed, the entire digital system or at least a part of it, is required to be inoperative/inactive as compared to the online-testing method where the system is required to be in its normal operation mode.

Off-Line Testing, because of their non-concurrent nature, are able to detect defects at a larger set of locations and require that the inputs as well as the state of the system are controllable. The purpose of typical testing processes is to construct test vector sets that are relevant for a given fault model or a set of them, in order to maximize the coverage while minimizing the test application execution time. Because the off-line test is usually applied after the circuit is manufactured as part of a more thorough manufacturing test, it is also used in maintenance tests on a regular basis during the lifetime of the system.

As compared to Off-Line Testing, On-Line Testing, also referred as concurrent checking or concurrent error detection, is a test technique used in permanent validation of a CUT integrity. An example of the on-line testing mechanism used in our research can be seen in Fig.2.8.

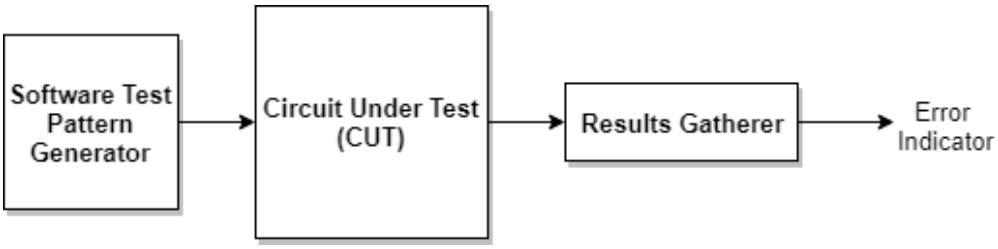


Fig. 2.8. Example of an On-Line Testing mechanism that is used in our research.

A consistent on-line test discipline is very important, especially in assuring the reliability in critical systems such as medical devices, satellites, telecommunications, solar tracking devices, railway control, and automotive systems, to name only a few. The behavior of a circuit can affect the entire system it is part of in many ways, thus the main task of an on-line test is to detect any

modifications as soon as they occur, ideally in a matter of seconds. The modifications in a CUT's behavior can be affected either by a permanent fault or an intermittent fault. Because of the limited manifestation duration as well as the unpredictability of their occurrence, the intermittent faults are creating a much more critical situation in maintaining the dependability of a system as compared to the permanent faults, mostly because their effect does not remain permanent. Off-Line testing, for example, will not be able to detect a transient fault the moment it happened (e.g. its effect disappeared) but can successfully identify the permanent faults, mostly because their effect remains constant for a large period of time.

The intermittent faults can affect the correct behavior of a circuit during its normal operation, resulting in a system failure. The effect of intermittent faults is often described as gate-level or transistor-level fault models. In order to make sure that the on-line testing technique used is correctly functioning, the integrity of the circuit is evaluated by inspecting all the inputs and outputs of the CUT and signaling them through an error indicator message or line to the user of the system or to the control unit.

2.3.1. Linear Feedback Shift Register

In this section we will discuss the linear pseudo-random generator called after the young French mathematician Evariste Galois.

In the vast domain of digital electronics, a Linear Feedback Shift Register (LFSR) defines itself as a chain of D Flip-Flops where the output of the last storage element is connected to the input of the first storage element, thus forming an endless cycle that provides a fixed number of test patterns [58]. Therefore, LFSRs represent typical mechanisms for generating test vectors in Built-In Self-Test (BIST) architectures. They are constructed as shift registers with feedback connection, operated by EXOR gates. In computing we can distinguish two types of LFSR architectures: Fibonacci and Galois representation. An example of a common form of Galois LFSR can be seen in Fig.2.9, which describes a typical LFSR structure that is generated by the primitive and irreducible polynomial function x^4+x+1.

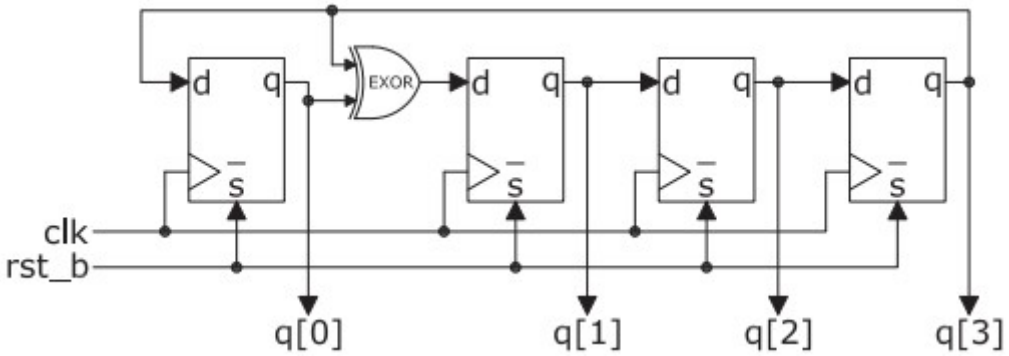


Fig. 2.9. The architecture of a Rank-4 Galois LFSR.

When initialized with a non-zero vector, a LFSR generates at its output, a pseudo-random, periodic sequence.

2.3.2. Signature Registers

Similar to a LFSR implementation, a Single Input Signature Register (SISR) contains also a fixed number of D Flip-Flops, each of them having a clock, a reset and a set input, connected in the same manner as the Rank-4 Galois LFSR register presented earlier in Fig.2.9. However, the SISR contains an additional EXOR gate at the input of the first D Flip-Flop, which is denoted with **A**. Additionally, the set line will load the SISR with an initial vector ***B*** = [0 0 0 0] while the reset line is always connected to a high logic level (**1'd1**), as can be seen in Fig.2.10.

2.3.3. Built-In Self-Test

Integrated circuits (ICs) nowadays are built around an internal logic that takes a set of inputs, applies successive operations on them and generates the expected outputs. In unfortunate cases, these complex circuits can be affected by errors derived from manufacturing processes.

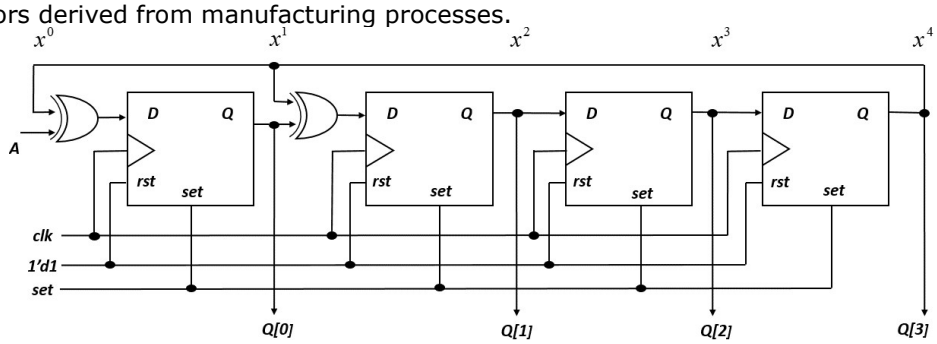


Fig. 2.10. SISR flip-flop design.

Errors are defined with respect to a system's service or in other words its intended functionality. The system's service is represented by a chain of external states and, in this context, an error occurs when at least one of the system's external states deviates from the intended, correct behavior.

A BIST error detection method transforms a design into a self-testable architecture, capable of detecting the presence of errors in an autonomous manner. According to Fig.2.11, the Test Pattern Generator (TPG) provides test vectors to be delivered to Logic Circuit inputs. Here a multiplexer is capable by means of a selection line to choose between the standard data inputs and the delivered generated test vectors. The injected test vectors will find their path through the internal logic of the circuit and eventually will be delivered at the Outputs.

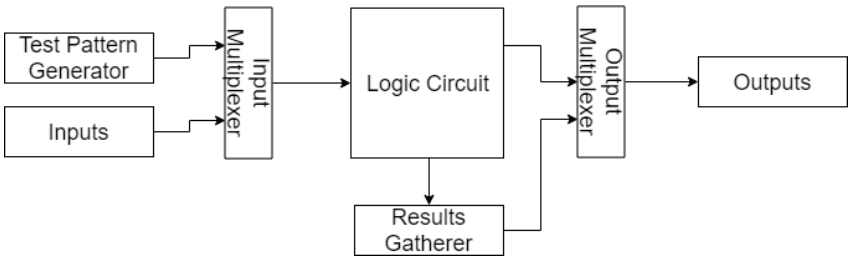


Fig. 2.11. Example of a Built-In Self-Test (BIST).

The Results Gatherer or sometimes called Output Response Analyzer (ORA) will perform data compaction (with loss of information) by processing all CUTs responses while exercised with the test vectors generated by the TPG. At the final stage of the compaction process, the ORA device will provide a signature. The signature is a reduced, fixed-size vector, characterizing the entire set of results. The signature for a CUT is associated equally to a TPG unit as well as an ORA device, generating CUT's input vectors. The gold signature refers to the signature obtained for the correct, fault-free circuit and is usually procured through simulation. The presence of errors in a CUT can be detected by comparing the obtained signature with the gold signature. This is managed by the output multiplexer that is mounted at the end of the BIST architecture. The Results Gatherer can be replaced by a SISR design or a Multiple Input Signature Register (MISR) structure which will be detailed in the next subchapters of this PhD Thesis.

2.3.4. In-Circuit Testing

In-Circuit testing reffers to the domain where ICs are verified for their functionality by using a dedicated test equipment such as bed of nails and flying probe devices, to name only a few.

Due to the cost of the Automated Test Equipment (ATE) necessary for at-speed functional tests, the test development time and effect, no possible options for upgrading to a BIST solution and the lack of fast/accurate fault diagnostics determined companies to drive away from functional testing and make use of Scan-based test strategies which, on the other hand, offer better alternative methods for testing inputs/outputs, more practical ways to detect delay defects, higher coverage for all circuit types and reduced testing time for embedded analog cores and Application Specific Integrated Circuits (ASICS) [59].

On the other hand, Design for Testing (DFT) procedures are intended to introduce engineers to the challenge of making ICs more testable. Integrated filters, Analog to Digital Converters (ADCs) and Digital to Analog Converters (DACs) will be also taken into account as they are today the main analog and mixed-signal cores found in Systems-on-Chip (SoC) devices. In particular, the possibilities offered by techniques using small circuit modifications will be specifically focused as the means to improve the testability of circuits and thus the coverage of faults, while at the most avoiding the degradation of the final electronic system's performance [60].

2.3.5. White-Box Testing

When it comes to software testing, one of the primary objectives is often the security. White-Box testing [61] is summarized in Fig.2.12 and is a software test method in which the tester is familiar with the internal structure, design, and implementation of the test item or Device Under Test (DUT).

The name White-Box testing derives from the fact that the tester is able to „see" inside the white/transparent box which is the software program. Here, the tester selects inputs through the code to exercise paths and to determine appropriate outputs which can result in an improved design, usability, and security.

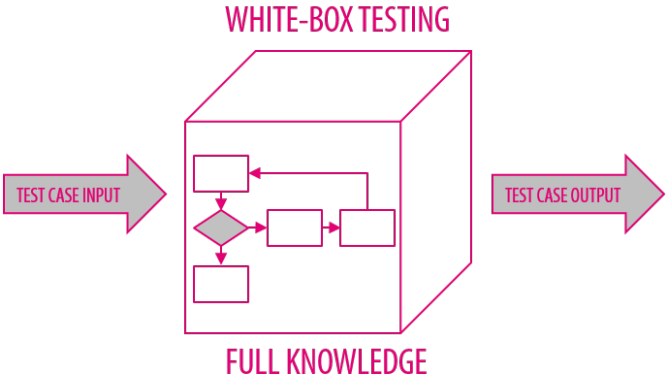


Fig. 2.12. White-Box Testing General Diagram.

Some of the advantages of the White-Box testing method are that the testing can be done at the earlier stage, without the need of a Graphical User Interface (GUI) and also that this testing method is more comprehensive. Some disadvantages are that, in order for the analysis of the internal structure of the system or component to be tested successfully, the White-Box testing method requires that the tester has advanced knowledge of programming and implementation. This is essential, especially in case of an update to the existent code for which the White-Box testing script was written, where the maintenance of such script that should be able to find security issues can be very difficult.

Also, it is important to mention that a White-Box testing method is closely tied to the Application Under Test (AUT), thus it cannot be re-used to every kind of implementation or platform out there. White-Box testing method is mainly applied for testing paths within a unit (Unit Testing), but can also be applied for testing paths between units (Integration Testing) and subsystems (System Testing). Unit Testing is often the first type of testing performed on an application because it helps to identify most bugs early in the lifecycle of software development, making them cheaper and easier to fix.

In our research, we implemented Unit Testing techniques and successfully investigated Communication, Control Flow and Error handling faults by using a White-Box testing strategy applied on a solar tracking device [18].

2.4. Hash Functions

Hash functions represent an important instrument in the secure computing paradigm operating at the core of many of today's most popular cryptographic protocols and services such as Public Key Infrastructure (PKI), TLS and IPSec. Other applications relying on cryptographic hash functions are authenticated access to Virtual Private Networks, file integrity verification and electronic voting systems [62]. A hash function maps a message of arbitrary length to a binary sequence of a fixed length, known as the hash value or message digest, being used to secure the integrity of the original message [63]. The security of hash functions relies on their collision resistance, meaning that given a message, it must be computationally infeasible [57] to find a different one generating the same hash value.

In our research, we implemented several techniques for improving the thoughput and performance of a SHA-256 hardware implementation with the aim to use them for future research regarding the implementation of a more secure medical DL-based system that stores confidential and sensitive data regarding patients health status and is used for training or inference of DL models with the scope of predicting possible diseases such as pneumonia [64] and COVID-19 [65].

2.4.1. SHA-256 Algorithm

The SHA-256 is formally presented in [66] and operates with words on 32 bits. The hash value or message digest of a message is a 256-bit vector. Message processing by SHA-256 involves three stages: padding and parsing, message schedule and hash computation or data compression. The padded message is parsed in blocks of 512 bits, each block being processed individually in order to obtain the final hash. The hash value is a vector of 8 words, defined as $H_j^{(i)}$, with *j* being the index of the hash word, $0 \leq j < 8$, and *i* being a counter for the currently processing 512-bit block. The initial values of the hash words, $H_j^{(0)}$, are given in [66].

The message schedule expands the 16 words of the 512-bit block into 64 words, denoted from W_0 to W_{63}. The first 16 words of the message scheduler are the very input words forming the 512-bit block. The remaining 48 words are generated by the message scheduler by using two dedicated functions, $\sigma_0^{\{256\}}$ and $\sigma_1^{\{256\}}$, and a binary adder modulo 2^{32}. The data compression phase makes use of 8 working variables, *a* to *h*, initialized with the current hash value at the beginning of each 512-bit block processing. This stage involves 64 iterations, each one updating the 8 working variables and using one of the 64 words generated by the message schedule.

The working variables' processing makes use of 4 functions, $\Sigma_0^{\{256\}}$, $\Sigma_1^{\{256\}}$, *Ch* and *Maj*. The variables' processing utilizes a lookup table for storing 64 word constants and generates the result using a modulo 2^{32} adder [66]. Finally, after the 64-th iteration, the hash values, $H_j^{(i)}$, are updated by adding to each of them the content of the corresponding working variables. After processing the last 512-bit block, the 8 hash words are delivered at the output as the message's digest.

2.5. Related Work

In this section, we will present the previous works related to our research. We examine previous attempts related to our DL applications, position optimizations of dual-axis solar trackers, hardware and software testing as well as the deployment of DL models on low-cost embedded platforms such as the Nvidia Jetson TX2. Then, we will continue with works related to the evaluation of DL models and systems as well as to the data collection, cleaning and labeling. Finally, we will review the previous works related to FPT testing as well as the ones related to hardware implementations of SHA-256 algorithms.

2.5.1. Different Deep Learning Applications for Detecting Fraud and Increasing Security

Even though there are some efforts in the direction of making the shopping experience better using mobile applications, they are lacking the ability to check for equivalence between prices at the shelf and in the computer system database. An example is the Carrefour Pay application [67], which gives customers the option to scan the barcodes of products they intend to buy, find out their price and automatically create a shopping list. However, this application lacks the ability to run fully on-device and is cloud-dependent. The work in [68] proposed an Android mobile application that creates an expense list based on the receipt photos taken by the smartphone camera with the help of the Tesseract OCR engine [69]. Their application performs poorly on images that contain noise, a problem which Tesseract and other OCR libraries are known to have [70]. A CNN based solution for the problem related to wrong labeling of use by date in retail food packages is presented in [71] where the authors make use of transfer learning and Maximally Stable Extremal Regions (MSER) algorithm in order to recognize the date within the region of interest (ROI). Their architecture has more than 42 layers, resulting in higher computational time and bigger complexity, which is not justifiable, especially when deploying them on mobile applications [72]. Also, regarding digit recognition, by making use of 2 CNNs and a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM), the work in [73] tackles the problem of detecting and recognizing car license plates from natural scene images but is not efficient for running in real-time. Also regarding license plate detection, this time in real-time and using OpenCV and Tesseract, the authors in [74] proposed an OCR algorithm that extracts characters from an image belonging to a license plate. Similar, another work using Tesseract for recognizing digits is also presented in [75], where the authors created a Thai lottery number reader Android application for blind people, having a processing time of 4 seconds. and a distance of 8 inches between the camera and the object .

In [76], a CNN for Fashion Classification and Object Detection was implemented by using a standard AlexNet architecture that was pre-trained using ImageNet for clothing type classification. In [77] a CNN model for clothes classification is presented, where, in order to evaluate the performance of their model, the authors adopted the classification accuracy as the measure criteria. In [78], a fast and accurate fashion item detection model is proposed by modifying and combining MultiBox and Fast R-CNN detection architectures. In [79], a method for identification of an outfit in an image followed by a classification using CNNs is presented. The authors use the weights from the InceptionV3 GoogLeNet architecture which was trained on the ImageNet dataset. In [80], a CNN approach to the classification of texture of clothing is presented, where the authors show that CNNs outperform seven well known hand-engineered feature extraction methods. In [81], a CNN application for scale analysis of clothes, styles, and fashion was developed by the authors for million images taken by people from all around the world spanning a couple of years.

Despite elaborate research and intensive work towards constructing automated animal recognition systems, only a few approaches resulted in usable tools. Some examples of available applications are iNaturalist [82], a popular app for the automated identification of animals and plants at the species level as well as Merlin Bird ID [83], an app that is aiming to identify 650 most common bird species

in North America based on their images. The authors in [84] present an animal recognition and identification system for automated wildlife monitoring by using CNNs to identify the most common animal classes from images taken by trap cameras in Australia. Their model is trained from scratch as well as by using fine-tuning (having pre-trained weights from the ImageNet dataset) and show that the deepest model, the ResNet architecture with 50 layers, also called ResNet-50, achieves the best performance. The work in [85] presents an animal recognition system based on Support Vector Machines (SVM) and a proposed CNN for animal classification. Their results show that the CNN model outperforms other classical ML methods when it comes to animal face recognition. A drawback of their model is that it can classify images of animals only by their faces and not also their entire body. The work in [86] presents a CNN and multiclass-SVM based method for animal classification from images using the AlexNet architecture. Similarly, the authors in [87] use a pre-trained CNN model on the AlexNet architecture that is combined with a multi-class SVM classifier in order to classify 20 animal classes from video frames, achieving 83.33% accuracy. Another works towards improving the classification work of ecologists are presented in [88], where the authors propose a VGG-16 CNN model that can classify 20 African wildlife species with 87.5% accuracy from images, as well as in [89], where the authors present a CNN model on a ResNet-18 architecture for the classification of wildlife animal species found on camera trap pictures obtained from U.S., Canada, and Tanzania.

We distinguish ourselves from the above mentioned related works by proposing an application that targets the detection of fraud regarding product and receipt prices in a Supermarket, an application that identifies the Romanian traditional motifs as well as an application that identifies the most common animals found in domestic areas of Europe in real-time.

2.5.2. Position Optimization and Testing of Dual-Axis Solar Trackers

Variable weather conditions are a challenge even for professional solar trackers. The work in [90] is bisecting this problem in two stages of direct and indirect sensing of the Sun. By using an Arduino UNO board, DC motors with gearbox, LDR sensor modules, angle sensors, timing circuit, Bluetooth module for wireless operation and a motor driver circuit, the authors obtain an overall power increase of 10 to 40% compared to a fixed-tilted solar panel, regardless of atmospheric conditions. A similar configuration found in [91] which includes an Arduino328, four LDRs and two servomotors, focuses on voltage measurements comparison, showing an average rise of 37% for a dual-axis solar tracker over a static solar panel. A multipurpose dual-axis solar tracker with two tracking strategies (normal and daily adjustment strategies) is implemented in [92], which can be applied to flat PV panels as well as Concentrating Solar Power (CSP) systems. The authors in [93] propose an autonomous solar tracker oriented with the support of light sensors and compare the experimental results of a dual-axis, single-axis and fixed panel. According to their chart, the two-axis model is considerably more efficient in generating an average of 77.58% Watt, followed by the single-axis with 61.75% Watt when compared to the static PV panel on the course of one day cycle. In terms of originality, two works are worthy of mentioning in this section. First is a novel design of a bi-level automatic solar tracker described in [94] where the authors investigate the possibility of rotating the payloads around orthogonal

axes and accomplish a unique structural design formed of four PV panels that allow tilting with only five servo-motors rather than eight used in regular four dual axis trackers. Their proposed method tops static PV panels and proves to be efficient compared to some modern dual-axis solar trackers from other works. Secondly, the work in [95] comes with a genuine idea of implementing an online sensorless dual-axis sun tracker based on the MPPT method. Apart from traditional sensorless solar trackers which operate in the open-loop regime, the proposed system works as a closed-loop device which cumulates the advantages of both sensor-based and sensorless dual-axis sun trackers but lacks all of their disadvantages.

Current advances in fault diagnosis and detection regarding solar harvesting systems can be sustained by applications such as ANNs, Line-to-Line (L-L) fault detection, online fault detection and diagnosis, simultaneous fault detection algorithm and simulation of microgrid systems with distributed generation. The work in [96] proposes an ANN-based model for fault identification and classification towards PV systems. The simulation results show that the method is efficient in detecting and classifying five different types of faults in PV systems. As the number of PV panels increases, L-L faults may remain undetected causing loss of energy and potential fire outbreaks. The authors in [97] resort to a fault detection algorithm based on multi-resolution signal decomposition (MSD) for feature extraction and ML techniques for decision making, demonstrating the accuracy of the adopted approach. Another method of detecting faults is linked to an online fault detection based on wavelet packets [98]. A concurrent fault detection algorithm is proposed in [99] that can successfully identify faults such as faulty PV modules, faulty PV String, faulty Bypass diode and faulty MPPT unit. A certain number of PV panels can be combined together, resulting in a Grid-Connected PV (GCPV) plant. The simulation experiments of the authors in [100] show that they are capable of switching between grid-connected and isolated modes of operation, resulting in the detection of errors that can occur in system behavior.

The interest shown in the literature for testing solar trackers at hardware level is lower compared to the software level. However, BIST routines are part of the industrial practices as well as an important research interest, especially in the last 10 years, where researchers have developed an interest in proving their efficiency in embedded memory testing [101]. In order to gain more flexibility with the memory testing techniques, a generic BIST methodology has been created around a set of March algorithm registers that can quickly adapt to the CUT and its corresponding memory types. With the help of Modelsim simulations, the authors achieved a proper validation, showing high coverage performance for common faults. More recent advances in the testing domain are highlighting the requirements for an efficient test pattern generation, configuration, oscillation techniques and multilayer features of BIST systems. P. Moorthy et al. present a novel TPG for the BIST without extending the length of test sequences [102]. Their proposed method is described as a generation of multiple patterns with single input change (SIC) that reduces the number of transitions during scan shift operations and decreases the switch-mode activity in the DUT. By using a LFSR as a test generator in combination with a MISR as a test compactor to verify a complex Wallace tree circuit, the authors demonstrate that a multiple pattern single input change (MSIC) method saves test power by 7% and reaches a fault coverage greater than 70%.

We distinguish ourselves from the aforementioned related works by firstly focusing on the development of a solar tracking device which rotates the PV panel according to the Cast-Shadow principle. Secondly, by implementing a White-Box

strategy oriented toward the detection of common software errors found in modern microcontroller units. Finally, by implementing an OBIST solution which is composed of an TPG, an ORA, ADCs and DACs and an idle state detector, all of them connected to several switch-batches for testing a solar tracking device that comprises an Optocoupler (LTV-847), an Arduino UNO and two L298N ICs.

2.5.3. Deep Learning Inference using Nvidia Jetson TX2 and Motion Detection

Similar work that evaluates the power efficiency of DL inference for image recognition on embedded GPU systems is presented in [103] where the authors compare different platforms like Nvidia Jetson TX1 and TX2 as well as a Tesla P40 and show that the Nvidia Jetson TX2 board is able to achieve the highest accuracy with the lowest power consumption. The work in [104] make use of an Nvidia Jetson TX2 to test a fully CNN for detecting traffic lights and use a power supply unit (12V) with stabilizer in order to increase the stability of the system, mentioning that the Nvidia Jetson TX2 has a low power consumption of around 10W, which is also confirmed by our experimental results using different CNN architectures. The work in [105] train and test two CNNs for classifying skin cancer images as Benign or Malignant on the Nvidia Jetson TX2, proving that this embedded platform is capable of handling DL computations even for training CNN models, not only for inference. The authors in [106] propose an object detection implementation using MobileNet as the backbone network on an Nvidia Jetson TX2 board, showing a higher fps and reduced model size when compared to other networks. Nvidia Jetson TX2 is used also in [107] where the authors propose a CNN based application that can run onboard a satellite in order to detect and classify boats on the open ocean from satellite imagery. Experimental results show that the Nvidia Jetson TX2 has almost half the power consumption when compared with standard systems designed for a satellite onboard processing. The authors in [108] use Nvidia Jetson TX2 for their proposed methodology regarding a faster and more accurate object detection in unmanned aerial vehicle (UAV) imagery. The work in [109] proposes a vehicle and pedestrian detection system that uses CNNs in order to evaluate traffic violations and which is implemented on an Nvidia Jetson TX2 board. Other related works that use Nvidia Jetson TX2 are regarding real-time ear detection [110], when developing embedded online fish detection and tracking system for ocean observatory network [111], real-time multiple face recognition [112], a streaming cloud platform for real-time video processing [113] and detecting diabetic foot ulcer in real-time [114]. A comparison between different DNN computing platforms, including Nvidia Jetson TX2, is made also by the work in [115].

Regarding motion detection, the authors in [116] present a comparative analysis of motion-based and feature-based algorithms for object detection and tracking and show that Adaptive Gaussian Mixture Model (AGMM) [117] is faster and more robust to illumination (shadows) than Grimson Gaussian Mixture Model (GGMM) [118] when performing on real-time videos. Also, the authors in [119] present a study on preprocessing methods for tracking objects in soccer videos, showing that background subtraction and edge detection are advantageous for detecting moving objects. OpenCV is also using AGMM together with other several algorithms for background subtraction which are presented in the works of [120] and [121] but, in comparison with OpenCV, the algorithms in OpenCV are more

modern, accurate and faster (a reason for this is because they are continuously developed by the OpenCV community).

We distinguish ourselves from the above mentioned related works by making use of a motion detection method implemented with the help of OpenCV in order to lower the power consumption of an Nvidia Jetson TX2 board that runs inference in real-time and which is powered 100% by a dual-axis solar tracking device.

2.5.4. Metrics for Evaluating the Performance of Deep Learning

Awareness regarding the importance of energy consumption can be seen not only in the field literature [118-126] but also in competitions such as the Low-Power Image Recognition Challenge (LPIRC) [127]. The question of energy consumption to be used as a metric when evaluating the performance of DL models or DL-based systems is of high importance for many papers in the literature. An example is the work in [8] where the authors advocate for a simple and compute-efficient metric, suggesting the use of energy efficiency as a metric when evaluating a DL model instead of "Red AI" which refers to the kind of AI research that uses extreme computational power and costs to achieve state-of-the-art results regarding accuracy. In order to measure the efficiency, after concluding that "Red AI" is used today by almost anyone in the literature, the authors suggest that future researchers should report the amount of work required to train a model using the total number of floating-point operations (FPO). Despite several advantages (e.g. agnostic and tied to the energy consumption of a hardware platform that runs a model), FPO has some limitations regarding taking into consideration the memory consumption of a model as well as its implementation, which, in the case of several implementations of the same model, can lead to different amounts of processing work.

A comprehensive analysis of important metrics such as accuracy, inference time, memory footprint, power consumption, parameters, and operations count as well as some combination of them for 14 DNN architectures is made in [128] where the authors did all their experiments using only the Nvidia Jetson TX1 board and show the importance of these metrics when designing efficient DNNs. Regarding power consumption, they show that it is mostly independent with the batch size for all neural network architectures. Similarly to the results in [128], the authors in [129] expand the analysis to over 40 DNN architectures both on Nvidia Jetson TX1 board as well as on an Nvidia Titan X Pascal GPU, highlighting the importance of metrics when evaluating the performance of a neural network, but lacking to provide the energy consumption of the systems the DNN architectures are running on. Also, the authors in [130, 131] contribute to the challenge of estimating the energy consumption in ML by providing useful guidelines and a large selection of the latest software tools for a ML expert who wants to design and estimate energy for future DL systems.

Some arguments against using only TTA as a metric when evaluating DL systems on the MLPerf Benchmark are presented by the work in [132] where the authors propose the Time-To-Multiple-Thresholds (TTMT) curves and Average-Time-To-Multiple-Thresholds (ATTMT) metric. By comparison, their metric targets the training part, without taking into consideration the energy efficiency whereas our metrics target both the training and the inference parts and take into consideration

the energy consumption as well as the energy cost of a DL-based system. Additionally, the TTA and ATTMT metrics are able to compare only different systems, whereas our metrics are able to compare also different models trained and executed in different systems, e.g. to identify on which hardware is better to train a DL model and on which hardware is better to run an inference with the same DL model.

2.5.5. Data Science-Oriented Computer Vision Application

The authors in [133] propose a solution called ImageX for sorting large amounts of unorganized images found in one or multiple folders with the help of a dynamic image graph and which successfully groups together these images based on their visual similarity. They also created many similar applications, e.g. ImageSorter [134], which besides sorting images based on their color similarity, is also able to search, download and sort images from the internet with a built-in Google Image Search option. A drawback of their applications is that the user is able to only visualize similar images, without also having this images automatically cleaned and sorted in their respective class folder with high accuracy. Also, the authors in [135] created an application called Sharkzor that combines user interaction with DL in order to sort large amounts of images that are similar. Their solutions only sort images by grouping them based on how similar they are to each other after a human interacted and sorted these images initially. An on-device option that uses DL capabilities and helps users find similar photos is presented also by Apple in their newest version of Photos app [136]. Regarding the detection of duplicate images, recently, a python package that makes use of hashing algorithms and CNNs that finds exact or near-duplicates in an image collection called Image Deduplicator (Imagededup) was released in [137]. When training DL models, the work in [138] is assessing the feasibility and usefulness of automated DL in medical imaging classification, where physicians with no programming experience can still complete such tasks successfully. The authors in [139] created the Image ATM (Automated Tagging Machine) tool that automatizes the pipeline of training an image classification model (preprocessing, training with model tweaking, evaluation, and deployment).

Our Computer Vision application distinguishes itself from the related works by offering more functionalities that make use of DL inference and by introducing the APC, APEC, TTCAPC and TTCAPEC metrics calculators, all in a user-friendly GUI.

2.5.6. Affordable Flying Probe-Inspired In-Circuit-Tester for Printed Circuit Boards

Placement accuracy is one of the primary issues in modern FPT systems. The authors in [140] are aware of the fact that the growing complexity of PCBs can introduce risks of faults at any stage of the manufacturing process and they propose a hybrid approach based on the combination of a traditional FPT and an Automated Optical Inspection (AOI) device. The second issue regarding modern FPT systems is the probe's navigation time between test nodes and is generally associated with the Traveling Salesman Problem (TSP). The authors in [141] investigate the ordering requirements for the complete amount of sample points and consider it an extension of the above mentioned TSP because they use more than one probe in their research. Test pad localization is the third issue which is concerned in recent FPT

designs and can be solved by applying a clustering method which was eployed by the authors in [142] referring to a two-stage clustering procedure on a 71040-pixel dataset derived from a PCB image with a precision ratio of 93.25%, proving that their method is highly efficient in identifying test pad locations for electronic boards which lack proper documentation. Finally, the test coverage problem is analyzed by Soh Ying Seah et al. [143] in their work which targets test load boards that are used as an interface between Automatic Test Equipment (ATE) and IC during package level testing using a hybrid approach between the ATE and FPT in order to verify four load board categories before and after merging the two methods together. Their experimental results showed a substantial test coverage increase (up to 100%) categories of electronic load boards.

Despite the fact that the previous works focus on the feasibility of combining FPT with other test methods as well as optimizing traveling paths between test nodes, test pad localization and fault coverage, we are implementing a FPICT that integrates the test node localization features of a CMM, resulting in a sensorless solution.

2.5.7. SHA-256 Hardware Implementation Acceleration Techniques

In [62], the authors investigate a number of acceleration techniques that are expected to improve the performance of hash functions, in general, and of SHA-1 and SHA-256 hash operations, in particular. The proposed techniques are related to loop unrolling, spatial precomputation, prefetching of data, design of an iterative architecture, and using a CSA structure for reducing the critical path and can be applied in any combination in order to attain the targeted performance.

In [144], authors investigate additional acceleration techniques applicable to SHA-256 and, by extension, to other functions from the SHA-2 family. Starting from the critical path of the algorithm, the authors first replace all binary adders by CSAs and include a final lookahead addition stage for computation of the sum in non-redundant representation.

In [145], authors construct hardware architectures for SHA-1 and for SHA-512 standards for high throughput. The hash acceleration techniques include loop unrolling and precomputation for part of the values used for generating the next working variables.

In comparison, we improve the throughput and performance of a standard SHA-256 hardware implementation by proposing several acceleration techniques.

3. DIFFERENT DEEP LEARNING-BASED APPLICATIONS FOR DETECTING FRAUD AND INCREASING SECURITY

Following, we will present three different DL-based applications that use several state-of-the-art CNN architectures with the aim to solve three different problems related to fraud and security.

3.1. Mobile Application for Receipt Fraud Detection Based on Optical Character Recognition

Despite the fact that we live in an Internet Era where multi-billion companies from the e-commerce industry like Amazon and Alibaba achieve record profits year after year due to their online sales, many offline grocery stores existed, exist and will continue to exist, recently, the most conventional ones being known under the name of a hypermarket/supermarket like Kaufland and Carrefour, to name only a few. A very common problem that occurs very often in hypermarkets/supermarkets in many countries around the world is that the price of individual products or the total price on the receipt that needs to be paid, don't always reflect the real price seen at the shelf and the real number of products in the customer's shopping cart. This problem exists due to a computer or human error and can happen because various reasons: a product gets scanned more or less than the number of times it was actually present in customer's shopping cart, doesn't get its price scanned correctly or because its price or special offer discount seen at the shelf is wrong, old or was not updated with the new price value. In order to solve this problem, our approach in this chapter is to provide a DL-based solution in the form of an on-device smartphone application that will give a user the options to take photos of the products at the shelf as well as of the paid receipt and automatically have their prices compared with the help of an OCR algorithm [146] based on image processing techniques and two CNNs.

3.1.1. Proposed Receipt Fraud Detection Application

In comparison to a cloud-based solution, the on-device inference not only has advantages regarding the price (cloud solutions can be costly) and latency (in case of poor internet connection) but also regarding the protection of a user's privacy [147]. Despite their smaller size when compared to a personal computer or a laptop, in recent years, smartphones became a platform of choice for DL applications [72] and big names in the industry (e.g. Google, Facebook, etc.) released their mobile versions of DL frameworks [45], [148] with the goal of running inference on the device (e.g. Android) itself.

When it comes to recognizing digits in natural images (e.g. photos taken in the hypermarket/supermarket with our smartphone), there are many problems that can occur, as can be seen in Fig.3.1 and Fig.3.2: the lack of contrast between the

pixels representing the digit and the pixels representing the background as well as the existence of noise like texture and patterns in their regions.

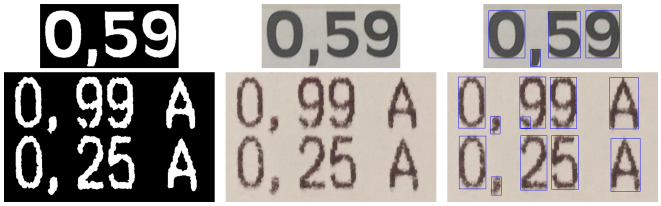


Fig. 3.1. Image Processing Techniques applied on the dataset regarding price images from Products (Top) and Receipts (Bottom): Images of Product and Receipt after thresholding (Left), Manually cropped images of Product and Receipt (Middle) and Contours detected in Product and Receipt images (Right).

Another problem can be the size and alignment of the digits, their font style or optical distortions (photos of these digits that represent the price can be situated under a plastic cover or having different angles at the shelf).

Moreover, because of the artificial lighting conditions in a supermarket, there can also be other obstructions like shadows or even that the lens of the smartphone camera can be defocused. Other problems are related to the design of digit recognition systems including the acquisition of images, their pre-processing, segmentation as well as representation and classification. The main problem in such a system is the segmentation part, which takes a string of digits (number) as input and segments (crops) them into individual, single digits because it requires a high number of hypotheses when extracting features from the contour of images as well as their background or foreground.

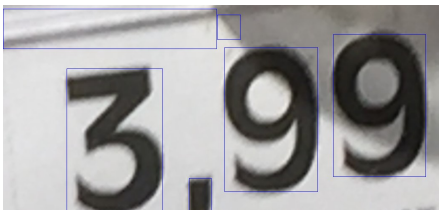


Fig. 3.2. Example of noise (Top-Left side) found in our dataset.

Regarding the proposed receipt fraud detection application summarized in Fig.3.3, there are two main phases considered. Phase 1, where a user takes one photo of the product he/she intends to buy (e.g. in order to be able to prove, in case of fraud detection, that the product at the shelf did had a wrong price tag), then takes a second photo with the price tag and then crops out the single price represented by multiple digits separated by a comma, usually found in a single row in a horizontal position. Phase 2 represents the step where the user is done with the shopping and received the receipt. Here, the user takes only a single picture, in this case, of the receipt and crops out all prices (usually found on the right side of the receipt, in multiple rows in vertical position). The user will be also able, at any point, to review any of the pictures or crops and, if necessary, to edit them.

The reason behind our design decision regarding the cropping of the exact single price (regarding the product) and cropping of multiple digits representing the bought product prices (regarding the receipt) is because of the speed and memory concerns, which in our case, are minimized as much as possible (an average of 1

second per item and 10MB app memory usage). As a result, both CNN models will have to process a much smaller size image, representing very exactly the ROI.

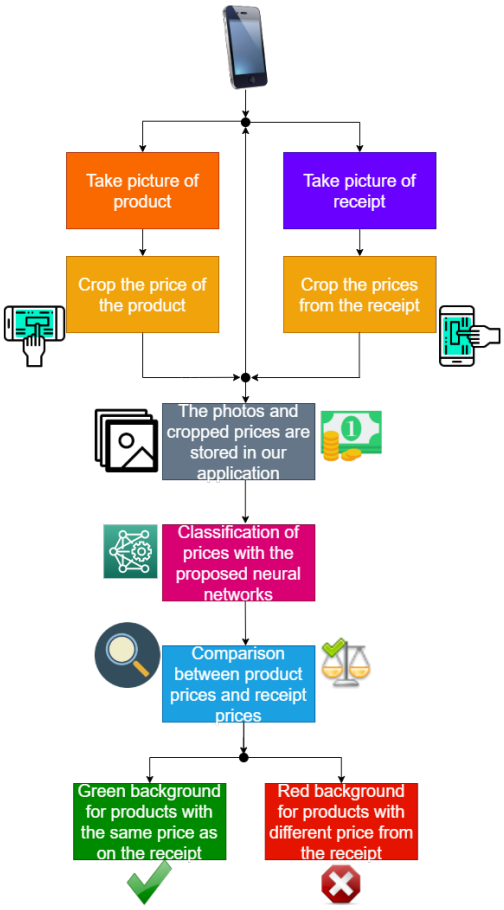


Fig. 3.3. Summary overview of the proposed application for Receipt Fraud Detection.

In order to successfully identify all the multiple digits, comma, and noises that are part of a price, the proposed receipt fraud application is going through 2 main phases, each having two main steps: A. Extracting digits from cropped image and B. Identifying digits, comma, and noise.

3.1.2. Implementation Decisions for Phase 1 (Product Prices)

Following we will describe our implementation decisions regarding the product prices.

Regarding product prices, for extracting digits from cropped images, we defined a function that receives an image as input and returns a list of images of digits, a comma, or noise. Sometimes dark shadows in an image can make the contour finding algorithm detect elements that are neither a digit nor a comma. It is not possible to avoid this problem without introducing another DL algorithm;

however, since we already have a DL algorithm that will receive the output from the contours algorithm, we can add the task to that algorithm in order to identify noise from an actual relevant character. For this function, we didn't try to remove the noise images, but instead, we focused on trying to isolate the actual digits, since it is more efficient to remove the noise images with the next classifier.

This function makes also use of some of the functions from the OpenCV library and finds the images in four steps:

1. Converts the image to grayscale
2. Inverts the color of the image. The contour-finding function assumes that there are objects in the image with a black background. Since in our data the opposite is true, the evident choice is to invert colors
3. Applies the adaptive threshold with the Gaussian method. This is known as a method that works very good regarding contour detection [149]
4. Finds contours

This manages to extract correctly all the digits in 99% of the cropped images. The reason for this is that sometimes a shadow or a dark object may block the image or make it have a very dark appearance, resulting in the impossibility of the algorithm to find contours since all of the darker pixels are connected. In those cases, the algorithm fails. The 1% of the images are the ones that contain very dark regions of pixels of noise, which causes the function to extract all the images as if it were only one digit.

Regarding product prices, for identifying digits, comma, and noise, the objective here is to have a CNN model that can classify each digit, comma or noise with the least amount of processing power used, in order to embed it later in a smartphone application (e.g. Android). To do this, we trained the CNN on augmented data.

With regard to data augmentation, we defined a function that applies four transformations to an image to create a random augmentation of that image:

- The first transformation rotates the image with an angle between -10° and 10°
- The second transformation adds a shadow to the image by randomly choosing a straight line and a side of the line, and darkens the pixels of that side of the line
- The third transformation makes a perspective warp to the image
- The last transformation adds random padding to each side of the image

Using this method of data augmentation we successfully created 12.000 images in our product prices dataset, containing 1.000 images of each digit (0-9), 1.000 images of commas and 1.000 images of noise.

For the image preprocessing, since every input for the CNN must have the same size, we defined a function that resizes and pads an image in order to fit a target size, which in this case we chose 50 width × 100 height, approximately the ratio that most digits in our Products dataset have. Before training, for a better convergence of the weights, all images were normalized.

As can be seen on the left side of Fig.3.4, the proposed model architecture used for identifying the Product Prices contains four layers:

1. Four units of a 2D CONV layer with a kernel size of 10×10, strides of 2×2 and ReLU activation function [26] followed by a dropout [36] rate of 0.5. This reduces the image from 100×50×1 to 46×21×4 and adds 404 trainable parameters
2. A max-pooling layer with a pool size of 2×2. This reduces the image to 23×10×4
3. A dense layer with 12 outputs and ReLU activation function that adds 11052 trainable parameters
4. A dense layer with 12 outputs and Softmax activation function that adds 156 trainable parameters

The reason for this network design approach was to obtain a small architecture, thus a more portable and faster algorithm. The proposed model was obtained by minimizing the number of layers and trainable parameters while preserving the model accuracy, in order to have a compact model that can be deployed on resource-consrained devices (e.g. smartphones).

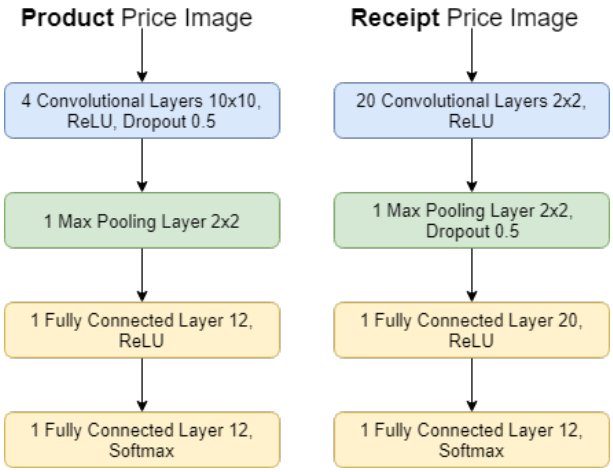


Fig. 3.4. Proposed CNN Architectures for identifying prices from cropped Product (Left) and Receipt (Right) images.

Since the scope of this work aims to classify only one font of characters, the reason for this being the lack of access to real-life product photos from supermarkets [150], [151], it is possible to obtain close to perfect results with a minimalistic model design. We used RMSprop as the optimizer, with a learning rate (LR) of 0.001 and categorical cross-entropy as the loss function.

3.1.3. Implementation Decisions for Phase 2 (Receipt Prices)

Regarding receipt prices, for extracting digits from cropped images, the processing of the receipt images is very similar to the processing of the product prices. For extracting the digits from cropped images we used the same function but changed the parameters of the adaptive threshold, achieving similar results.

Regarding receipt prices, for identifying digits, comma, and noise, we defined a function that applies a random brightness change to the images. Receipts

images are expected to be better aligned and to contain less random shadows, so the expected digit images are only varying in brightness according to the lighting level where the photo is being taken and the camera that takes the picture. For that reason, any other augmentation would add unnecessary complications and noise to the DL model. Using this method of data augmentation, we created 1.200 images in our receipt prices dataset, containing 100 images of each digit (0-9), 100 images of commas and 100 images of noise (anything that isn't a digit or a comma is considered noise, this includes other characters as well). For the image preprocessing, we padded and resized the images to fit 10 width x 20 height, approximately the ratio that most digits in our Receipts dataset have.

As can be seen on the right side of Fig.3.4, the proposed CNN model architecture used for identifying the Receipt Prices is a similar model to the one used for the Product Prices, with the same number of layers but with the following configuration:

1. 20 units of a 2D CONV layer with a kernel size of 2×2, strides of 1×1 and ReLU activation function. This reduces the image from 20×10×1 to 19×9×20 and adds 100 trainable parameters.
2. A max-pooling layer with a pool size of 2×2 followed by a dropout rate of 0.5. This reduces the image to 9×4×20.
3. A dense layer with 20 outputs and a ReLU activation function that adds 14.420 trainable parameters.
4. A dense layer with 12 outputs and a Softmax activation function that adds 252 trainable parameters.

The optimizer, LR and the loss function are the same as the ones described in Phase 1. For both network designs, the procedure was to start with a minimal architecture (1 CONV unit, max pooling and an FC layer) but in both cases (products and receipt) that was not enough to learn the features. We then added a second FC layer with 12 units (the same size as output, which is proven [31] to be able to solve any classification problem if given enough units), but the results were only slightly better, so we added more units to the CONV layer to be able to obtain different local features from the images. After this step, the model overfitted on the training data, which made us add a dropout with a 0.5 rate before the CONV unit (in both cases) to eliminate it. The results regarding product price detector were very good, but not regarding receipt. Since input images are smaller in receipt prices, the CONV layers are 2×2, meaning that the amount of trainable parameters is less; for this reason, the receipt price detector has more CONV and FC units. Also, in the receipt case, the dropout performed better after the max pooling.

3.1.4. Android Application GUI

The proposed receipt fraud application is composed of two views: Products (Items) view" and "Receipt view", each of them having 3 frames (header, body, footer).

We implemented the Android smartphone application using Python programming language. For the image processing, handling the camera and image storage, we used the OpenCV library; for the application development, we used the Kivy Framework [152] and finally, for deploying the application into an Android app, we used the generic Python packager called Buildozer [153].

In the case of products, as can be seen in Fig.3.5, the header frame presents the user a button for adding new products he/she intends to buy, called "Add item" as well as the "Total price" field, which represents the total price of all products added in the shopping cart (sum of all rows from column 4 explained below).

Fig. 3.5. Summarized Android GUI view of the proposed Receipt Fraud application: Products (Items) View (1 and 2); Receipt View (3); Products (Items) View after price comparison between Receipt and Products was made and Price is equivalent (4) or not equivalent (5).

The body frame contains 4 columns. After pressing the "Add item" button from header section, Column 1 presents two buttons called "Add picture", meaning that the user should take a picture of the product he/she intends to buy and "Add price tag", meaning that the user should take a picture of the product price tag. After taking the picture of the price tag, a third button appears called "Crop Price" which gives the user the option to perfectly crop only the price out, representing the price of the product (digits separated by a comma) from the price tag. This is important because, usually, on a price tag there are many other characters and digits representing the name of the product, name of the company, the price/Kg, barcode, etc., and detecting all of them will be out of the scope of this application. All these three pictures (full product, price tag, crop from price tag) will be stored in column 1, all in the same row that belongs to a single product, the user having the possibility to enlarge or edit them at any time. The decision for allowing the user to take a picture of the full product instead of just the price tag is because it can be a very important part of the proof, in case of a receipt fraud. Column 2 shows the unit price which is the identified price by our first CNN model which received the cropped price tag image as input. In case of failure regarding digit extraction, the prices can also be edited. Column 3 gives the user the option to enter the number of times he/she intends to add a product he intends to buy in the shopping cart. By default, every product receives an amount value of "×1" and can be increased (e.g. in case the user wants to buy more times the same product) or decreased to the amount value of "×0" meaning that the product will be removed from the grocery list. Column 4 presents the total price per row for every product added in the shopping cart (e.g. if a product costs 1.50 but the number of times in column 3 is ×2, the total price per row in column 4 will be 3.00). The footer frame shows a "Finish" button meaning that all products intended to be bought are added, cropped and prices correctly identified and the user is now ready to take a photo of the receipt.

In the case of receipts, as can be seen in Fig.3.5, the header frame shows the user a button for adding a receipt photo called "Add receipt" as well as a button

for cropping the multiple prices vertically aligned on the right side of the receipt called "Crop receipt". The body frame contains 3 columns. Column 1 shows the picture of the receipt taken by the user. Column 2 shows the picture of the vertically aligned cropped prices from the receipt photo, which our second CNN will receive as an input. Column 3 presents the identified prices (containing digits and commas) by our second CNN model. In case of failure regarding digit extraction, these prices can be also edited. The footer frame shows a button called "Return to items" which is necessary to be used after detecting the receipt prices, in order to go back and see the price comparison results (entire row of a product that has a different price than that found on the receipt will have a red-colored background, otherwise, if the price is equivalent, it will have a green-colored background) between products and receipt.

It is important to know that after installing the application, the first thing that needs to be done is to allow camera permission from the Android Settings Menu. The smartphone application was compiled using Python 3.6.7, OpenCV 4.0 and Kivy 1.11.0.dev0. We tested the app with a Moto Z Play Android smartphone. The Moto Z Play includes a 5.5-inch 1080p display, an octa-core Qualcomm Snapdragon625 system-on-chip, and 3 GB of RAM. The application takes 3 seconds to open. The average memory use of the appplication is always less than 10MB. After an hour of use, the application consumed 5% of battery (120 milliampere-hour (mAh)).

The app is installed locally and doesn't require a network connection, works smoothly, without lag, except when processing images. When it is detecting a price tag it takes about 1 second to do it. When processing a receipt the time is longer, depending on how many contours the algorithms found in the image. It takes an average of 1 second per item.

The size of the .apk file is around 21MB and when installed on the smartphone device, the size of the smartphone application is 73MB (OpenCV used in the image processing algorithm together with the 2 proposed CNNs).

By implementing an OCR ourselves, we succeeded to adapt it to our specific dataset, resulting in a smartphone-friendly model that can handle noise very well, unlike other OCR methods [70].

3.1.5. Experimental Setup and Results

For the experimental setup, it is important to notice that our two models were trained on a Desktop-PC with the following configuration: on the hardware side, we use an Intel(R) Core(TM) i5-6600 CPU @ 3.30GHz and a GIGABYTE GeForce GTX 1060 WINDFORCE 2 GPU with 6GB GDDR5 memory; on the software side, we use an Ubuntu distribution of Linux, version 18.04, together with Keras 2.2.0 framework using Tensorflow 1.10.

During training, for the first proposed CNN model, regarding the recognition of prices with commas from products, we applied it to the augmented data for 200 epochs, with a batch size of 100 and a validation split of 10%. We obtained 98% training accuracy after 18 epochs and at the end of the training, we obtained 99.84% accuracy, 99.50% validation accuracy, and an overall 99.96% test accuracy, as can be seen in Table 1.

Table 1. Test Accuracy and other metrics of the CNN model regarding Product Prices.

Class	Samples	Test Accuracy	Precision	Recall	F1-Score
0	37	1	1	1	1
1	125	1	1	1	1
2	147	1	1	1	1
3	41	1	1	1	1
4	72	1	1	1	1
5	19	1	1	1	1
6	21	1	1	1	1
7	38	0.99	1	0.97	0.98
8	8	1	1	1	1
9	365	0.99	1	0.99	0.99
Comma	365	0.99	0.99	1	0.99
Noise	41	0.99	0.97	1	0.98
Overall Test Accuracy			**99.96%**		

Additionally, the accuracy and loss results during training and validation of the CNN model regarding Product Prices, are presented in Fig.3.6.

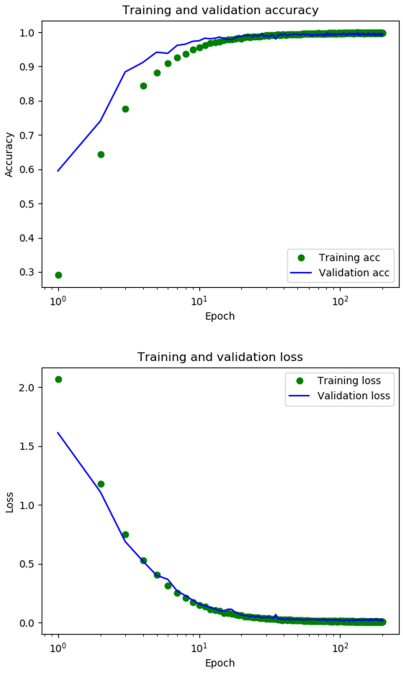


Fig. 3.6. Training and Validation Accuracy (Top) together with Training and Validation Loss (Bottom) for the CNN model regarding Product Prices.

Also, for the second proposed CNN model, regarding the recognition of prices with commas from receipts, during training on the same setup, we applied it to the augmented data for 1.100 epochs, with a batch size of 1200 and a validation

split of 10%. We obtained 99% accuracy after 360 epochs and at the end of the training, we obtained 100% validation accuracy, and an overall 99.35% test accuracy, as can be seen in Table 2.

Table 2. Test Accuracy and other metrics of the CNN model regarding Receipt Prices.

Class	Samples	Test Accuracy	Precision	Recall	F1-Score
0	28	0.99	1	0.96	0.98
1	29	0.97	0.68	1	0.81
2	17	1	1	1	1
3	12	0.97	1	0.57	0.72
4	7	1	1	1	1
5	22	1	1	1	1
6	13	1	1	1	1
7	7	1	1	1	1
8	6	1	1	1	1
9	43	1	1	1	1
Comma	69	0.98	0.98	0.95	0.97
Noise	107	0.98	0.96	0.99	0.97
Overall Test Accuracy			**99.35%**		

Additionally, the accuracy and loss results during training and validation of the CNN model regarding Receipt Prices, are presented in Fig.3.7.

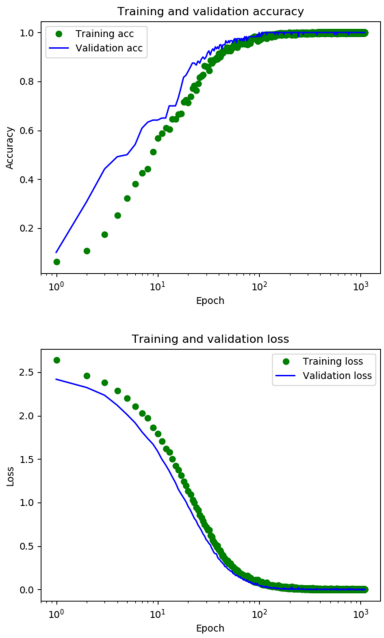


Fig. 3.7. Training and Validation Accuracy (Top) together with Training and Validation Loss (Bottom) for the CNN model regarding Receipt Prices.

More details regarding the number of samples used for the test accuracy of both CNN models as well as other metrics values like Precision, Recall and F1-Score [154], can be seen in Table 1 and Table 2.

Because of the very small size of our model's architecture, especially the one used in Phase 1, it is important to mention that it is possible to run the training even on any other smaller personal hardware device, which is an important advantage.

In order to validate the decision of creating our own OCR implementation with regard to recognition accuracy and speed, we compared the proposed OCR with the Tesseract OCR Engine [69] on the same system configuration and the same number of samples.

As can be seen in Table 3, the proposed OCR outperforms Tesseract by a large margin regarding test accuracy on images with cropped Product and Receipt prices from our dataset.

Table 3. Recognition Accuracy and Speed Comparison between the proposed OCR and Tesseract OCR on images with cropped Product and Receipt prices.

Accuracy Test	**Samples**	**Proposed OCR**	**Tesseract OCR [69]**
Cropped Price tags	254	98.43%	51.18%
Cropped Receipt prices	32	71.87%	3.12%
Speed Test	**Samples**	**Test Time [s]**	**Average Time/Recognition [ms]**
Cropped Price tags proposed CNN	254	6.6	24.2
Cropped Price tags Tesseract		27.34	101.9
Cropped Receipt prices proposed CNN	32	3.51	685.3
Cropped Receipt prices Tesseract		1.02	186.4

The first CNN, regarding price tag recognition, is 47% more accurate and 76% faster than Tesseract, while the second CNN, regarding receipt prices recognition, is 68.57% more accurate than Tesseract, but due to our image preprocessing, Tesseract performs the inference 70% faster than the proposed method.

3.2. Identification of Traditional Motifs using Convolutional Neural Networks

Ancient knowledge was preserved in many places around the globe in many forms (architecture, wood carving, pottery, etc.), one of them being the form of motifs sewn in the textiles. Often, these traditional motifs are found in clothes that are copied without permission or without giving credit by the international clothing design industries [155], [156] as seen in Fig.3.8. Romanian traditional motifs are beautiful patterns that help the Romanian traditional clothing be characterized by unity (such as the composition of the garment, the raw material from which the

pieces of clothing are made, the tailoring, the harmonious fresh colors or by the stitching points) and by continuity (evolution of clothes over the years) [173], [174].

Fig. 3.8. Example of cultural appropriation of traditional clothes by major brands [155].

In order to prevent cultural appropriation and the takeover (or theft) of traditional clothes by major brands [155, 156], this work contributes to proposing a method for the classification of Romanian traditional motifs using CNNs and which, when compared to other traditional manually-designed feature extraction methods, outperforms them by a large margin.

3.2.1. Proposed System Design for Classifying Romanian Traditional Motifs

We propose a system that inherits the advantages of ResNet-50 architecture, the most important ones being to obtain higher accuracy and faster training performance regarding image classification. Our model is trained using the Keras framework, a Tensorflow high-level API written in Python and integrated into the proposed classification system. For training and processing the features detected in the hidden CONV layers, a CPU as well as a high-performance GPU were used. Using a webcam, these detected features (motifs) are identified by the proposed CNN, as shown in Fig.3.9.

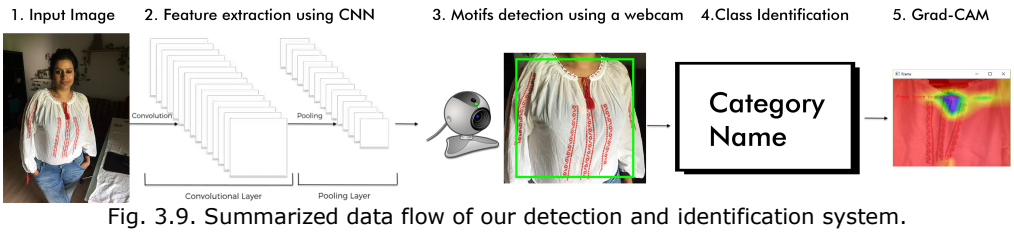


Fig. 3.9. Summarized data flow of our detection and identification system.

We trained the proposed model on the widely known academically dataset called ImageNet using a modified ResNet-50 architecture, as seen in Fig.3.10, which was initially loaded with pre-trained weights.

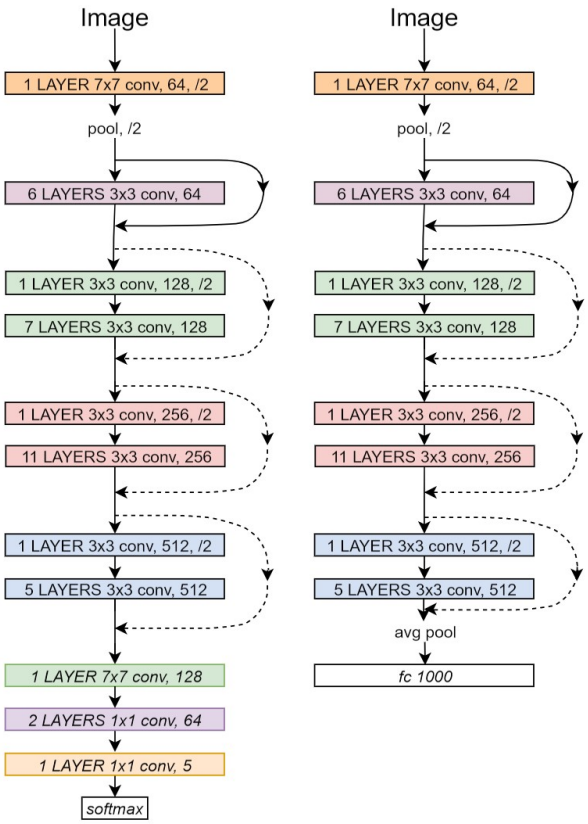


Fig. 3.10. Our proposed network architecture (left) and a typical ResNet (right). The dotted arches represent an increase in dimension.

ResNet-50 is chosen for its favorable properties in transfer learning and also because it achieves better accuracy than VGGNet and GoogLeNet while being computationally more efficient than VGGNet [128], as the experimental results prove. Also, because ResNet-50 is quicker to train than the deeper variants, thus allowing for more hyperparameter tuning.

We modified these weights by training on the images from 5 categories (4 categories for the motif classes: clothes, ceramics, carpets, painted eggs and a fifth category for images not representing any of the learned motifs) without the FC and the previous layer. Instead, we added 3 CONV layers, equivalent to FC layers, each accompanied by a 50% dropout and a BN layer. The neural network architecture is completed by a final flattening layer followed by the output dense layer consisting of 5 neurons corresponding to the 5 classes. In order to reduce the complexity of training the network, consisting of 36.392.834 parameters, we trained on phases, for 56 epochs. In Fig.3.11, the top-left figure shows how fast the accuracy evolved through the 56 epochs used for training.

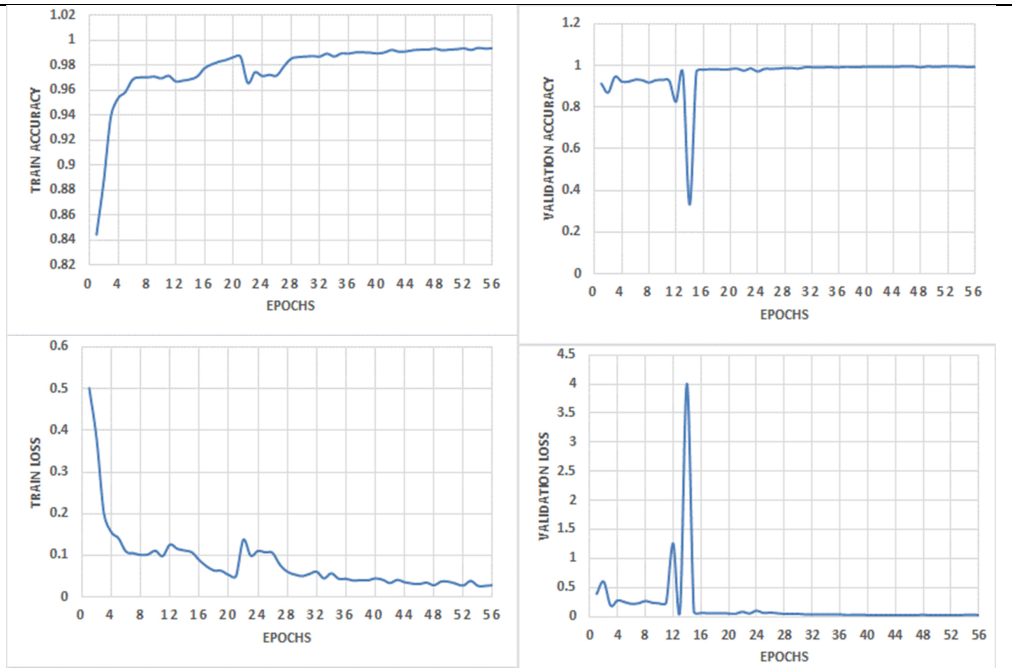


Fig. 3.11. Train and Validation Accuracy (top) as well as Train and Validation Loss (bottom) of the proposed model.

For increasing the performance of the proposed network, before the actual training, we reduced the size of all images belonging to the 5 categories to 256 pixels by keeping the aspect ratio and trained them with a batch size of 32. The need for resizing the images arises due to the network size, available GPU computation power and size of the receptive field. By scaling down the images, our network is able to identify the key features in the initial layers instead of being learned later, resulting in less computation per layer and fewer memory requirements. Additionally, in order to increase the amount of relevant data in our dataset, we apply data augmentation. This helps our CNN model to robustly classify objects that may exist in a variety of conditions, such as different orientation, scale, brightness, location, etc., resulting in our CNN model to gain the invariance attribute. The following data augmentations are applied: horizontal flipping with a probability of 0.5; zoom in the range of 0.8 and 1.2 of the original image and sheer transformation with a shear angle of 0.2. Finally, in order for the network to converge faster, the per-channel mean of the ImageNet dataset is subtracted from the input images. In other words, the mean of red, green and blue (RGB) channels are subtracted. This is a common practice, as even the pre-trained ResNet-50 had the images preprocessed in this way [27].

As expected, the accuracy increased above 99% in the last 16 epochs. However, the train loss graph seen at the bottom left of Fig.3.11 exposes the error on the training dataset where the error was under 4% in the last 13 epochs, hitting the minimum error of 2,67% in the 53rd epoch and then slightly rising. Because the validation accuracy seen in top-right of Fig.3.11 stays above 99% in the last 26 epochs and the validation loss seen in bottom-right of Fig.3.11 was under 3% for the last 17 epochs, we decided to end the training process at the 56th epoch in

order to avoid overfitting the training set. The graphs seen in Fig.3.11 are useful in evaluating the performance of the training and how effectively the proposed CNN architecture was trained. Because ResNet-50 was pre-trained on ImageNet (the size of this dataset is around 150GB, which consists of 1.2 million labeled images and 1000 categories in the training set as well as 50.000 images, 50 per class, in the test set), the first few layers already captured universal features like curves and edges that are relevant to our problem. In consequence, we decided on making the proposed network to focus on learning only the new dataset-specific features (the dataset containing the traditional motifs) in the subsequent layers. For this, first, we trained only the last 3 CONV layers, then the last 1/3 and finally the last 2/3rds. As mentioned earlier, we used one dataset of positive samples with images from the 4 categories containing Romanian traditional motifs and another dataset from ImageNet for the negative samples. These negative samples were drawn randomly from ImageNet in each training epoch, up to the number of positive samples. Since it is much larger than the positive examples dataset, we use only a fraction (2%) of ImageNet. This allows the model not to be overwhelmed by negative examples and to learn new parameters from the positive ones. This 2% portion of ImageNet is randomly selected at the beginning of each epoch. Before training, we resize each input image to 256×256 pixels and take random 224×224 crops out of it.

The modifications to the standard ResNet-50 architecture can be seen in Fig.3.10 presented earlier and are as follows:

1. We remove the last FC layer and the global average pooling layer before it. This is done in order to give the next, newly added layers more fine-grained information.
2. The ResNet-50 architecture applies seven bottleneck (they are called "bottleneck" because it adds the activations of two branches in one) blocks which include CONV layers with a stride of 2 and a single max pooling layer with a stride of 2, thus reducing the size of the input image 32 times. Since we start off with 224×224 pixel images, the activations map that is output by ResNet-50 has a dimensionality of 7×7×2048. More exactly, the bottleneck blocks which include a stride of 2 and the max pooling, are reducing the dimensions 32 times (224/32=7). Thus, we add a CONV layer with a kernel size of 7×7 and 128 filters. This layer is able to capture the knowledge present in the output of the pre-trained ResNet-50 while lowering the dimensionality.
3. We then add two CONV layers with a kernel size of 1×1 and 64 filters. These layers are equivalent to FC layers with 64 units. The reason for making this choice is because traditional CONV networks having FC layers cannot manage different input sizes, whereas fully CONV networks have CONV layers that can do this.
4. All three CONV layers described above use ReLU activations. Each CONV layer in the proposed network architecture is followed by a dropout layer with a rate of 0.5 and a BN layer. We chose a dropout value of 0.5 because it regularizes the network effectively for the purpose of our work. The BN layer gives our CNN model resistance to vanishing gradient during training by decreasing the training time, resulting in a better performance.
5. Finally, a Softmax-activated CONV layer with a kernel size of 1×1 and 5 filters follows. The layer uses a Softmax activation function to classify the input image characteristics generated in different classes based on the training dataset.

For training, since there are no balanced samples in each class (ImageNet = 456.567 images, Carpets = 132 images, Ceramics = 4.688 images, Clothes = 19.549 images, and Painted eggs = 190 images), we oversample the Carpets and the Painted eggs classes 8 times. We do oversampling only on training data in order for our model to generalize better on new data. Examples of random images from the 4 categories (clothes, ceramics, carpets, painted eggs) identified by our CNN model can be seen in Fig.3.12.

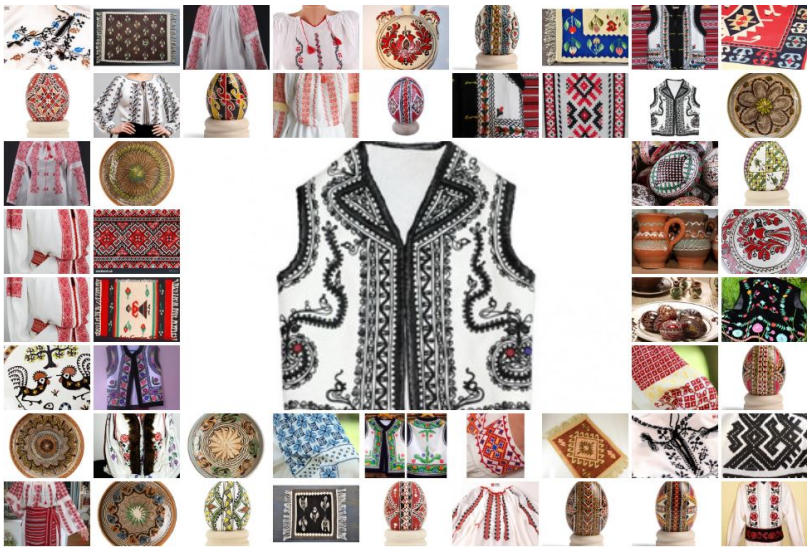


Fig. 3.12. Example of random images from the 4 categories (clothes, ceramics, carpets, painted eggs) identified by our model.

We utilize the following training schedule:

1. First, we freeze the weights and biases on the pre-trained ResNet-50 network and train only our newly added layers for 5 epochs with a LR of 0.1, for the next 3 epochs with LR of 0.01 and for the next 3 epochs with LR of 10^{-3}. This enables these randomly-initialized layers to train without perturbing the earlier layers.
2. Then, the first 10 bottleneck blocks of ResNet-50 are kept frozen, and the rest are trained with an LR of 10^{-4} and 10^{-5} for 5 epoch each. After this, 6 more bottleneck blocks are thawed and trained for with LR of 10^{-4} and 10^{-5} for 5 epoch each.
3. Finally, the whole network is trained for 20 more epochs with an LR of 10^{-5}.

Also, in order to keep track of the accuracy results after every epoch, checkpoints and logs files are automatically generated and saved. This is important not only for keeping the training records but also because the whole dataset doesn't need to be retrained in case of possible errors. As mentioned earlier, we implemented the proposed model in order to detect and identify motifs using a webcam. First, we do inference using the trained model to detect the predominant class in the image. Then, in order to see what part of the identified class contributed the most to the successful classification, we apply the Gradient-weighted Class Activation Mapping (Grad-CAM) algorithm [157]. Grad-CAM approach adds more interpretability (simplicity) [158], transparency and trust [159] in our model. An

example of how two classes (ceramics and clothes) are being detected, can be seen in Fig.3.13.

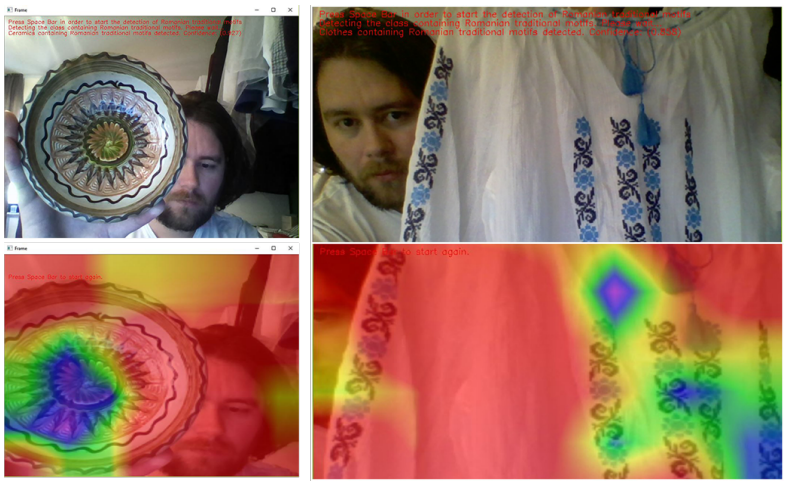


Fig. 3.13. Top Left: Detection of Ceramics class (i.e. Horezu). Top Right: Detection of Clothes class (i.e. IA). Bottom: Grad-CAM heatmap is generated for both classes.

One important aspect to notice here is that after the Grad-CAM technique is applied, the image gets automatically zoomed for better clarity of what part of the detected class contributed the most (where the CNN identified motifs in the image to actually distinguish between the classes) to the prediction accuracy.

3.2.2. Experimental Setup and Results

For the experimental results regarding training and testing of our model, we make use of a Desktop PC system that has the following configuration: on the hardware side, we use an Nvidia GTX 1080 Ti GPU together with an Intel-Core i5-7500 3.4GHz Quad-Core Processor. On the software side, we used an Ubuntu distribution, version 16.04 together with CUDA 9 [160], CuDNN 7 [161] and Tensorflow 1.5 using the Keras framework.

The experimental results of the proposed model's accuracy are summarized in Table 4 and show that our novel CNN model implementation is able to classify the Romanian traditional motifs found in 4 categories (carpets, ceramics, clothes, and painted eggs) with high accuracy and reduced processing time.

Table 4. Test Accuracy together with other metrics values and webcam processing time.

Identified Classes	Test Accuracy [%]	Samples	Precision	Recall	F1-Score	Webcam Processing Time [ms]
Carpets	92.8	14	1.0	0.93	0.96	47.7
Ceramics (e.g. Horezu)	98.4	459	1.0	0.98	0.99	46.8
Clothes (e.g. IA)	99.3	1944	1.0	0.99	1.00	4.8
Painted Eggs	100	20	1.0	1.00	1.00	48.7

ImageNet	99.7	2555	0.99	1.00	0.99	0.04
Overall Accuracy [%]	**99.4**					

However, it is important to notice here that for the webcam detection, identification, and processing time, we use an Asus ROG-GL752VW Laptop with an Intel-Core i7-6700HQ 2.6GHz CPU having an NVIDIA GeForce GTX 960M with 2GB memory. In order to show how well our system performs the classification task of Romanian traditional motifs found on the 4 categories, we also presented the Precision, Recall, and F1-Score metrics values in Table 4.

The model comparison results presented in Table 5, clearly show that the proposed ResNet-50 model outperforms other architectures in classification accuracy and by means of Keras' Grad-CAM technique, our solution can be used with high confidence when it comes to features extraction.

Table 5. Model Classification Comparison Results

Models	**Grad-CAM**	**Accuracy [%]**
SVM [162]	NO	35.0
Random Forests [162]		38.3
Transfer Forests [162]		41.4
Fine-tuned FC Layers CaffeNet [76]		46.0
Fine-tune All Layers CaffeNet [76]		50.2
CNN [163]		61.2
AlexNet [38]		81.8
VGG_S [38]		82.9
Deep CNN [80]		84.5
Inception v3 [79]		98.2
Our proposed model	YES	99.4

3.3. Real-Time Identification of Animals Found in Domestic Areas of Europe

The world's human population is constantly growing and the necessity for shelter and food is pushing our civilization towards exploring new areas and building residential areas there. A consequence of this is that unaware, we are destroying many flora and fauna habitats, thus steps towards preserving biodiversity are of major importance. Regarding animals, in order to track and monitor them, classical animal recognition methodologies like ear tattoos, embedded microchips or transponders in the electronic devices, sensors and radio frequency identification (RFID) [164] were used for many years and are still in use today. These methods are intrusive in their nature and depend heavily on the direct contact between humans and animals (e.g. when tagging them for research purposes). A minimally intrusive and remotely method in monitoring and identifying animals is that of using camera traps (e.g. especially in the case of wild animals), but a common limitation is that it requires spending a huge amount of time to manually label and classify these images (which can reach millions) [11]. This is due to the complexity of the real-life pictures analyzed which can contain perturbations regarding background,

illumination, position, posture, inter-class variations, etc. In order to accelerate the discovery, tracking and monitoring of animal species that are on the verge of extinction, the recent AI algorithms, e.g. DL, are showing promising results. An example in this direction is Microsoft's AI for Earth project [165].

Considering the animals found in domestic areas, in the case of residents or farmers, the need to avoid accidents (e.g. animal-vehicle collisions) or maintain the security of their domestic animals and crops against wild animals is also crucial and show a clear demand for systems that can automatically detect, classify and store information about the identified animal class, regardless of real-life scenarios and challenges.

3.3.1. Proposed Real-Time Animal Class Identification System

The proposed real-time animal class identification system is composed of two main processes called Main Process and Inference Process, as can be seen in Fig.3.14.

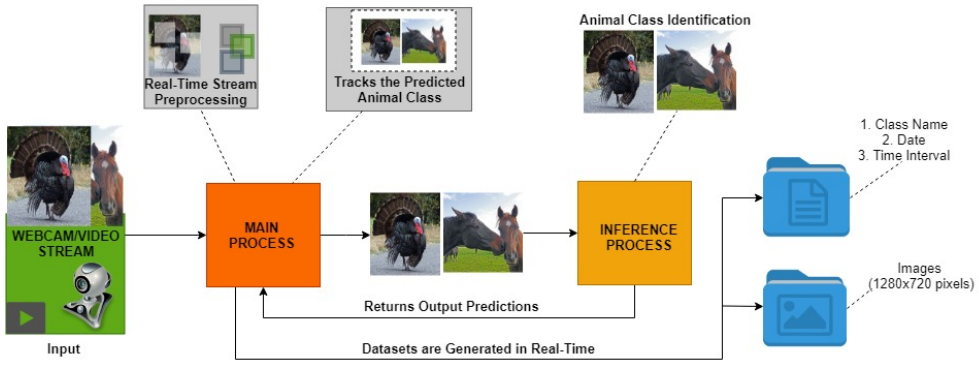


Fig. 3.14. Summarized view of the proposed real-time animal class identification system.

The Main Process is responsible for the model-specific preprocessing (e.g. normalization of the RGB input channels), for the real-time stream preprocessing (e.g. resizing the full-size frame from the webcam/video to 256×256 pixels and doing a center and random crop for a better association of different parts with the corresponding animal class) and for making a certain number of streamed frames available to the Inference Process. The real-time stream preprocessing is realized with the help of the OpenCV library in the Main Process (because of issues running OpenCV in a multiprocess environment). In order to know how many fps our system (the computer used for running the proposed model) can perform inference on, initially, before running the Inference Process, a speed test is performed. This speed test is measuring the inference speed for 1, 2, 4, 8, 16 and 24 frames as well as the inference time for each number of frames. The speed test is relevant because it helps to make available a higher number of images per second (e.g. 4 frames instead of 1 frame) to the Inference Process, increasing the chances of a better prediction. When tracking the predicted animal class, the Main Process is taking the decision of generating the 2 datasets, one containing textual information and the second one containing images, in real-time by verifying which animal class was present the most in the last 3 seconds over other animal classes and "nothing detected" class. This rule will help reduce misdetections.

As mentioned earlier, in order for the proposed real-time animal class identification system to have an increased recognition accuracy, we decided to train four state-of-the-art CNNs for image classification using Keras framework with Tensorflow backend. More exactly, we fine-tuned the VGG-19 [38], InceptionV3 [40] ResNet-50 [27] and MobileNetV2 [43] architectures, each of them having a different number of trainable parameters and prediction accuracy.

The first architecture we trained our model on is called VGGNet [38], which was originally proposed in 2014 when it won first place in the ILSVRC challenge regarding image localization as well as second place regarding image classification. More exactly we make use of the VGG-19 version, composed of 19 weight layers, 16 CONV and 3 FC layers.

As can be seen in Fig.3.15, because the original VGG-19 architecture didn't work well on our dataset, we modified it by a) adding a GlobalAveragePooling layer after the last MaxPooling layer; b) removing first Dense/FC layer; c) modifying the number of units of the second Dense/FC layer from 4096 units to 1024; d) modifying the number of units in the last Dense/FC layer from 1000 units to 34 representing our animal classes. It is important to mention that the proposed VGG-19 architecture has 12.3 million trainable parameters as compared to around 144 million parameters of the original VGG-19 and uses ReLU as the activation function for all layers but the last one, which uses the Softmax activation function.

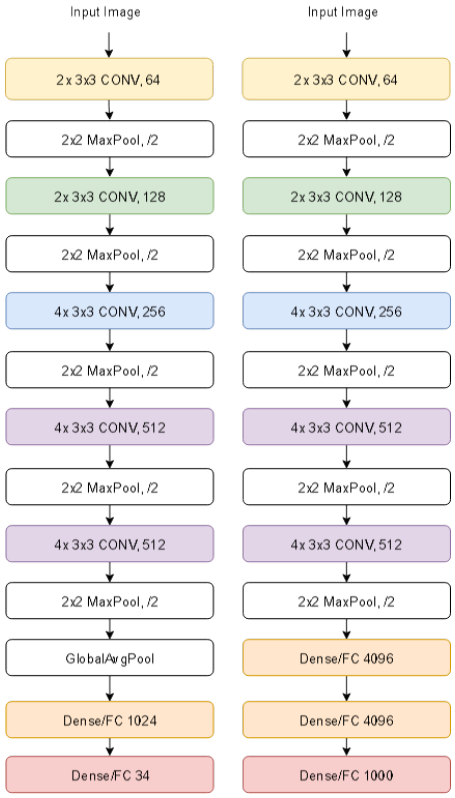


Fig. 3.15. Proposed (left) and Original (right) VGG-19 architecture.

The second architecture we trained is called InceptionV3 [40] which was proposed in 2015 in order to increase the ImageNet classification accuracy. We summarized the proposed InceptionV3 architecture in Fig.3.16 where, in order to present a compacted view, we present a compressed view of it.

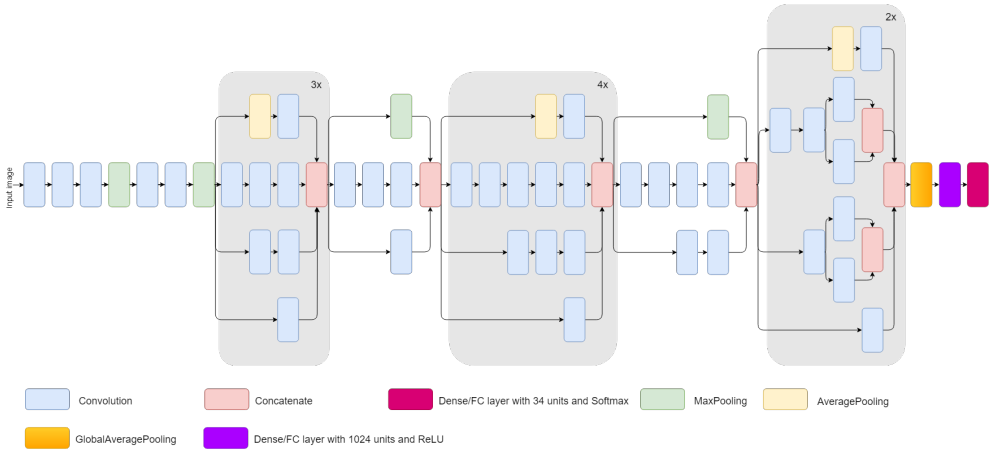


Fig. 3.16. Schematic diagram of the proposed InceptionV3 model architecture (compressed view).

It is important to mention that all CONV layers are followed by a BN layer as well as a ReLU activation function. Due to the GlobalAveragePooling layer, all the channels after the last CONV layer are averaged out, reducing the number of parameters, thus having a smaller weights size than the original VGG and ResNet architectures. More exactly the proposed InceptionV3 architecture has 23.9 million parameters. Regarding the last 2 Dense/FC layers, the first FC is having 1024 units and ReLu as the activation function and the second one is having 34 units representing the animal classes and Softmax as the activation function. We used the SGD optimizer with an initial LR of 0.01, momentum 0.9 and categorical cross-entropy as the loss function.

The third architecture we trained is called ResNet [27]. More exactly, we use a conventional version of ResNet called ResNet-50 which has 25.6 million trainable parameters across 49 CONV layers and 1 FC layer and which we modified by removing the top FC layer with outputs for 1000 target classes, and replacing it with an FC layer with outputs for 34 target classes, as seen in Fig.3.17. Replacing the last ResNet-50 layer with a single layer worked best for our scenario. More expressive replacements (e.g. 3 FC layers of respectively 256, 128, 64 units) were tested but were found to be hard to train and inaccurate, the reason for this being the limited number of images available. An example of an architecture setup that didn't work well was: ResNet-50 -> FC layer with 64 units (ReLU activation) -> BN -> FC layer with 34 units (Softmax activation). It is important to mention that we use ReLU as the activation function not only after the first BN layer but also after all BN layers inside the 16 residual blocks consisting of 4 CONV BLOCKs and 12 IDENTITY BLOCKs (each having 3 CONV layers = 48 CONV layers). Only after the FC layer with 34 units, we use Softmax as the activation function. The identity shortcuts are presented as solid and dotted lines shortcuts; the solid lines are used where the input, as well as the output, have the same dimensions, whereas the dotted lines are used where their dimension is different.

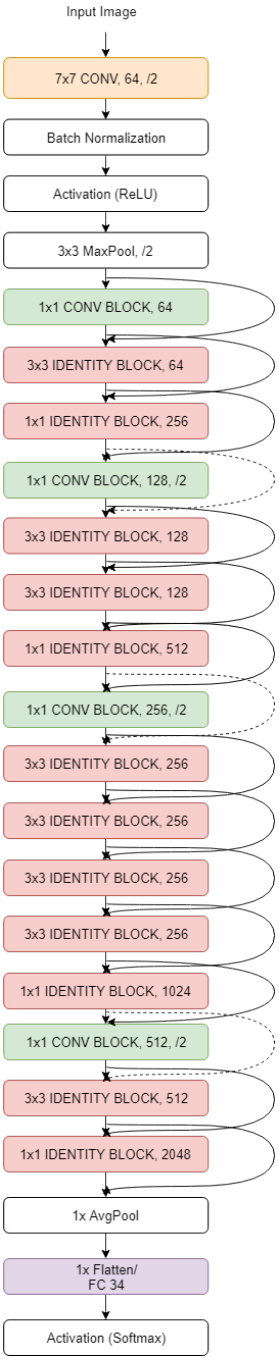


Fig. 3.17. Proposed ResNet-50 architecture with the last FC layer having 34 outputs representing the animal classes. On the right side are presented the identity shortcuts between all residual blocks (solid lines when the input and output have the same dimensions; dotted lines when otherwise).

As mentioned earlier, the ResNet-50 architecture is more computationally efficient than other architectures such as the VGG and GoogLeNet, requiring 0 extra parameters, having considerably fewer operations (e.g. ResNet with only 34 layers requires 18% of the operations compared to a VGG with 19 layers) and achieving better accuracy. We used SGD as the optimizer (initially we experimented with Adam, but SGD proved to give better results) with an initial LR of 0.01, the momentum of 0.9 and categorical cross-entropy as the loss function.

Forth and last architecture we trained our model on is called MobileNetV2 [43] which was released in 2018 at the Conference on Computer Vision and Pattern Recognition (CVPR) and which outperforms all other 3 architectures presented earlier regarding training and testing accuracy on our animal images dataset. Moreover, it is a very light-weight architecture that uses in our case 6.1 million trainable parameters, being much more efficient than all other architectures we trained our model on. MobileNetV2 is suitable for mobile devices because it requires less space, memory, and computation, thus it can run faster (e.g. when running inference in real-time).

The proposed MobileNetV2 architecture can be seen in Fig.3.18 and is composed of 10 blocks of stride 1 and 6 blocks of stride 2, each having ReLU6 as the activation function after the 3×3 depthwise CONV layer. Because each stride block is composed of 3 CONV layers, the entire architecture is composed of a total of 52 CONV layers, 1 GlobalAveragePooling layer as well as 2 Dense/FC layers. It is important to mention that the first Dense/FC layer with 1024 units has ReLU as the activation function and the second (last) Dense/FC layer with 34 units representing the animal classes, has Softmax as the activation function. We used SGD as the optimizer with an initial LR of 0.01, the momentum of 0.9 and categorical cross-entropy as the loss function.

In practice, it is very rare to have a self-made dataset consisting of millions of images and very common to have small datasets consisting of hundreds or a few thousands of images. With the size of a dataset reaching hundreds of thousands of images, the complexity of a neural network also grows, this being the reason why DNNs consisting of millions of parameters are very expensive to train, with most complex models taking weeks (for example, the original ImageNet ILSVRC model was trained on 1.2 million images over the period of 2-3 weeks across multiple GPUs). In order to avoid this problem, we fine-tunned our models. More exactly, we use a pre-trained version of all 4 architectures on the ImageNet dataset which already provides us with the learned features relevant for our animal class identification problem. We apply fine-tuning instead of training from scratch, in order to prevent overfitting, reduce the training time and benefit the environment [8].

We carry out experiments on a home-made dataset containing animal images from personally made pictures (for some of the classes), as well as animal pictures from other online resources, scrapped for educational purposes with the help of a home-made Python script. Because the home-made Python script randomly searched for thousands of images containing the name of this classes all over the Internet (thus the reason we cannot reference the image sources), many of the images found were noisy, meaning that additional manual data filtering was required for removal of invalid images. Because the scope of our work is related to animals found only in domestic areas of Europe, we considered only 34 species.

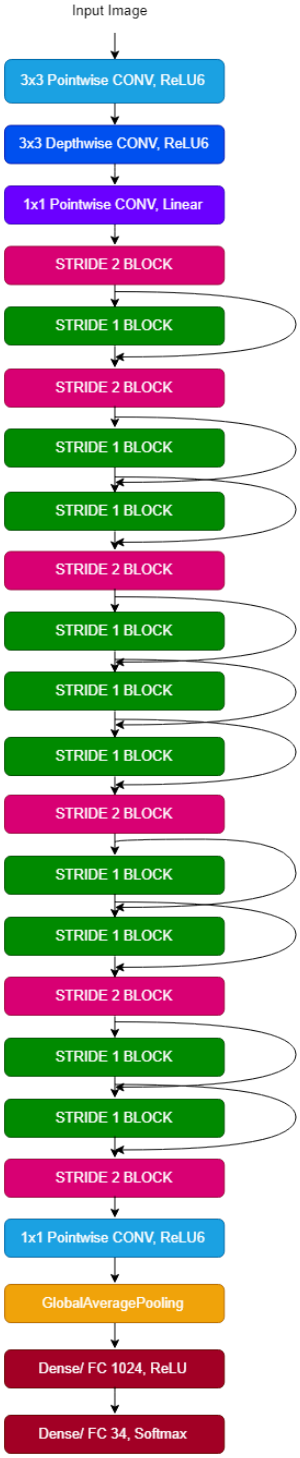


Fig. 3.18. The proposed MobileNetV2 architecture.

More specifically, our training dataset contains, as can be seen in Fig.3.19 and Fig.3.20, a number of 34 classes, each with large variations in scale, lighting and pose: bat, bear, canary, cat, cattle, chicken, deer, dog, donkey, duck, fox, frog, goat, goose, hamster, hedgehog, horse, lizard, magpie, mole, owl, parrot, pig, pigeon, rabbit, raven, sheep, snake, sparrow, squirrel, stork, tortoise, turkey, and woodpecker.

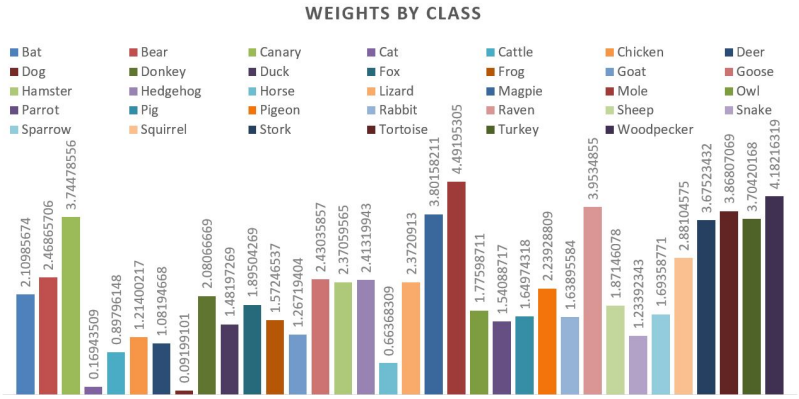


Fig. 3.19. Weights by Class for the considered 34 animal classes.

The training, validation and test sets were having a total number of 90.249 images (72.469 images for training, 8.994 images for validation and 8.786 images for testing).

Before training, we resize each input image to 256 pixels by maintaining the weight by height ratio and take 224×224 pixels random crop out of it. This helps out the network to learn key features in the early layers rather than later, resulting in faster training and less memory used. Also, because pooling layers induce translational invariance, our CNN model is able to robustly classify images of animals that can exist in a variety of conditions, such as location, brightness, orientation, scale, etc. In order to increase the amount of relevant data in our dataset, we apply data augmentation (horizontal flipping with a probability of 0.5; zoom 0.1 of the original image as well as shear transformation with a shear angle of 0.1). The resized version of our dataset (train, validation and test set) consists of a total of 90.249 images.

Following, we will present all 4 architectures (VGG-19, InceptionV3, ResNet-50, and MobileNetV2) our model is trained on in order to determine which is the best architecture that achieves the highest validation accuracy, has the smallest number of trainable parameters and trains the fastest. During training, each class was weighted to give more importance to classes that are underrepresented. For example, as seen earlier in Fig.3.19, dog and cat classes were heavily underweighted (0.09199101 and 0.16943509) because of the large number of training samples in these classes. In order to decrease the training time as well as estimate the error rate of the loss function, we make use of the following callback functions from Keras: EarlyStopping with patience (number of epochs with no improvement after which LR will be reduced) of 10 and ReduceLROnPlateau with a factor (by which the LR will be reduced) of 0.2 and patience of 3.

Fig. 3.20. Random images from our training dataset. A total number of 34 classes representing animals found in domestic areas of Europe (bat, bear, canary, cat, cattle, chicken, deer, dog, donkey, duck, fox, frog, goat, goose, hamster, hedgehog, horse, lizard, magpie, mole, owl, parrot, pig, pigeon, rabbit, raven, sheep, snake, sparrow, squirrel, stork, tortoise, turkey, and woodpecker).

We trained our VGG-19 model consisting of 12.359.202 trainable parameters with a batch size of 64 on a number of 72.469 train images belonging to 34 animal classes for 27 epochs in the following training schedule:

1. First, we trained the first 13 epochs with an LR of 0.0001. This brought the validation loss to 0.33725 and validation accuracy to 89.84%.
2. Next, we trained for 6 more epochs: 3 epochs with a LR of 2e-05 and 3 epochs with a LR of 4e-06. This didn't improve the validation loss but increased the validation accuracy to 90.84%.
3. Finally, we trained for 8 more epochs, 3 epochs with a LR of 8e-07, 3 epochs with a LR of 2e-07 and 2 epochs with a LR of 3e-08. This didn't improve the validation loss and validation accuracy, so we decided to stop the training, leading to convergence at 90.56% overall test accuracy, as can be seen in Fig.3.21 and Table 6.

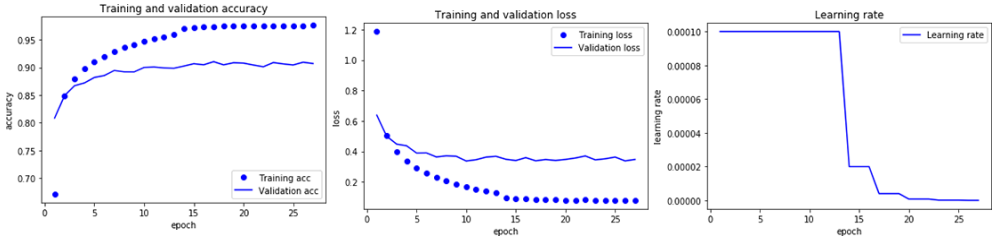


Fig. 3.21. Train and Validation Accuracy (left), Train and Validation Loss (middle) as well as LR (right) of the proposed VGG-19 model.

The total amount of time needed to train the 27 epochs of the proposed VGG-19 model was 20.273 seconds (around 5 hours and 37 minutes).

We trained our InceptionV3 model consisting of 23.901.378 trainable parameters with a batch size of 64 on a number of 72.469 train images belonging to 34 animal classes for 51 epochs in the following training schedule:

1. First, we trained the first 12 epochs with an LR of 0.01. This brought the validation loss to 0.38108 and validation accuracy to 89.16%.
2. Next, we trained for 7 more epochs with an LR of 0.001. This brought the validation loss to 0.25620 and validation accuracy to 94.06%.
3. Next, we trained for 16 more epochs: 3 epochs with a LR of 0.0003, 3 epochs with a LR of 8e-05, 3 epochs with a LR of 2e-05, 3 epochs with a LR of 3e-06 and 4 epochs with a LR of 6e-07. This brought the validation loss to 0.25176 but without improving the validation accuracy.
4. Finally, we trained for 16 more epochs: 3 epochs with a LR of 1e-07, 3 epochs with a LR of 3e-08, 3 epochs with a LR of 5e-09, 3 epochs with a LR of 1e-09, 3 epochs with a LR of 2e-10 and 1 epoch with a LR of 4e-11. This didn't improve the validation loss, but brought the validation accuracy to 94.28%, leading to convergence at 93.41% overall test accuracy, as can be seen in Fig.3.22 and Table 6.

The total amount of time needed to train the 51 epochs of the proposed InceptionV3 model was 38.853 seconds (around 10 hours and 47 minutes).

We trained our ResNet-50 model consisting of 25.583.394 trainable parameters with a batch size of 64 on a number of 72.469 train images belonging to 34 animal classes for 28 epochs in the following training schedule:

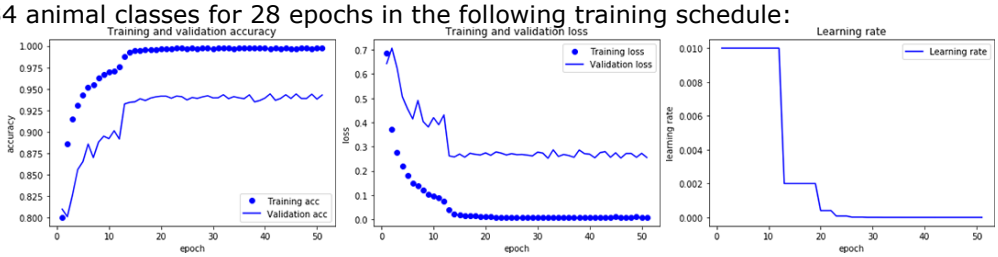


Fig. 3.22. Train and Validation Accuracy (left), Train and Validation Loss (middle) as well as LR (right) of the proposed InceptionV3 model.

1. First, we trained the first 10 epochs with an LR of 0.01. This brought the validation loss to 0.36145 and validation accuracy to 89.37%.
2. Next, we trained for 4 more epochs with an LR of 0.001. This brought the validation loss to 0.26349 and validation accuracy to 93.28%.
3. Next, we trained for 10 more epochs: 3 epochs with an LR of 0.0003 and 7 epochs with an LR of 8e-05. This brought the validation loss to 0.25335 and validation accuracy to 93.62%.
4. Finally, we trained for 4 more epochs: 3 epochs with a LR of 2e-05 and 1 epoch with a LR of 3e-06. This didn't improve the validation loss, but brought the validation accuracy to 93.66%, leading to convergence at 93.49% overall test accuracy, as can be seen in Fig.3.23 and Table 6.

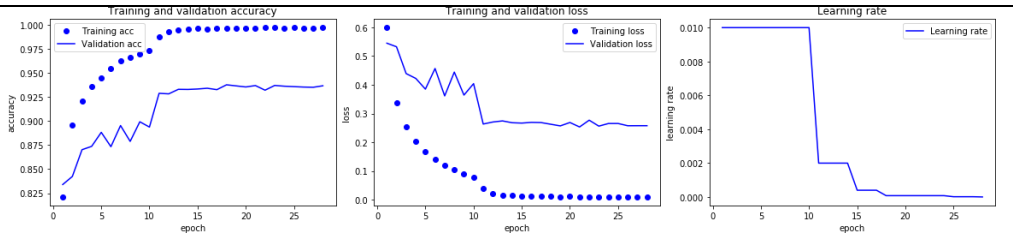


Fig. 3.23. Train and Validation Accuracy (left), Train and Validation Loss (middle) as well as LR (right) of the proposed ResNet-50 model.

The total amount of time needed to train the 28 epochs of the proposed ResNet-50 model was 21.396 seconds (around 5 hours and 56 minutes).

We trained our MobileNetV2 model consisting of 6.186.658 trainable parameters with a batch size of 64 on a number of 72.469 train images belonging to 34 animal classes for 51 epochs in the following training schedule:

1. First, we trained the first 17 epochs with an LR of 0.01. This brought the validation loss to 0.46479 and validation accuracy to 88.75%.
2. Next, we trained for 9 more epochs with an LR of 0.001. This brought the validation loss to 0.22574 and validation accuracy to 94.05%.
3. Next, we trained for 18 more epochs: 4 epochs with a LR of 0.0003, 3 epochs with a LR of 8e-05, 5 epochs with a LR of 2e-05 and 6 epochs with a LR of 3e-06. This brought the validation loss to 0.21380 and validation accuracy to 94.28%.
4. Finally, we trained for 7 more epochs: 3 epochs with a LR of 6e-07, 3 epochs with a LR of 1e-07 and 1 epoch with a LR of 3e-08. This didn't improve the validation loss and validation accuracy, so we decided to stop the training, leading convergence at 94.54% overall test accuracy, as can be seen in Fig.3.24 and Table 6.

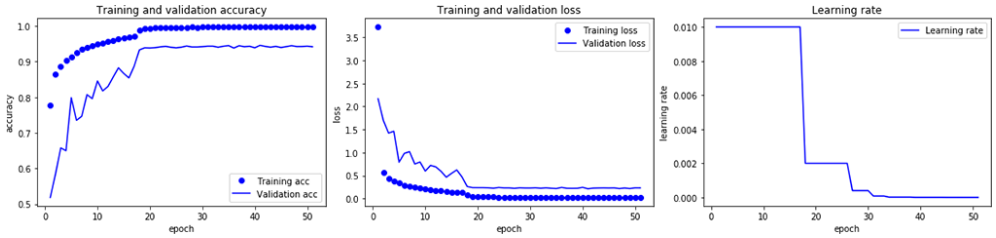


Fig. 3.24. Train and Validation Accuracy (left), Train and Validation Loss (middle) as well as LR (right) of the proposed MobileNetV2 model.

The total amount of time needed to train the 51 epochs of the proposed MobileNetV2 model was 38.847 seconds (around 10 hours and 47 minutes).

3.3.2. Experimental Setup and Results

For the experimental setup and results, our model was trained on a Desktop PC system with the following configuration: on the hardware side, we used a Desktop PC having an Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz (12-Core), ~3.5GHz processor, 32 GB RAM, and an Nvidia GTX 1080 Ti GPU; on the software side, we used Windows 10 together with CUDA 9.0, CuDNN 7.6.0 and Tensorflow

1.10 using the Keras 2.2.4 framework. However, for the real-time identification of animal classes using the webcam, we use an Asus ROG-GL752VW Laptop with an Intel-Core i7-6700HQ 2.6GHz CPU having an NVIDIA GeForce GTX 960M GPU with 2GB of memory. The experimental results are presented in Table 6 and clearly show that all CNN models are able to identify 34 classes representing animals found in domestic areas of Europe with high accuracy.

Table 6. Test Accuracy Report for the proposed models.

Animal Class	**Samples**	**Test Accuracy [%] for VGG-19 (V), InceptionV3 (I), ResNet-50 (R) and MobileNetV2 (M)**			
		V	**I**	**R**	**M**
Bat	236	91.6	95.4	90.3	94.9
Bear	208	88.4	93.8	92.8	92.8
Canary	130	86.9	92.3	93	93.8
Cat	279	91.7	94.9	95.3	95.0
Cattle	552	89.3	91.9	90.2	92.9
Chicken	402	92.5	93.8	95.8	96.5
Deer	401	92.8	93.8	94.3	96.0
Dog	346	85.5	92.8	92.2	93.1
Donkey	224	75.8	87.6	91.5	90.2
Duck	329	85.1	85.4	87.8	90.6
Fox	247	88.3	91.5	93.1	90.3
Frog	312	92.9	92	93.6	95.8
Goat	333	81.1	88	88.9	91.9
Goose	169	86.0	88.9	90.1	92.9
Hamster	197	88.3	94.9	93.4	95.4
Hedgehog	192	97.9	99.5	96.9	97.4
Horse	742	94.7	96.4	93.1	95.8
Lizard	211	90.0	90.5	92.4	91.9
Magpie	115	93.0	94.7	93.9	95.7
Mole	103	93.3	97.1	98.1	97.1
Owl	256	91.8	95.7	95.3	97.7
Parrot	307	94.1	96.1	95.4	94.8
Pig	274	93.1	96	96	97.4
Pigeon	227	89.9	94.7	94.7	97.8
Rabbit	285	93.0	94.4	96.2	97.2
Raven	105	88.8	94.4	94.4	95.2
Sheep	274	88.6	89.7	92.3	89.8
Snake	392	95.9	96.4	95.9	95.7
Sparrow	275	90.2	93.8	94.9	94.5
Squirrel	139	90.6	92.8	92.8	94.2
Stork	126	96.8	98.4	99.2	100.0
Tortoise	123	95.9	91.9	92.6	93.5
Turkey	141	88.0	94.4	97.2	93.6
Woodpecker	116	94.9	96.6	97.4	95.7
Overall Test Accuracy [%]:		**90.56**	**93.41**	**93.49**	**94.54**

In order to show how well our system performs the animal class identification task, we also present the Precision, Recall, and F1-Score metrics values in Table 7 where, for simplicity, because we use the same animal classes and the same number of samples in the same order like in Table 6, we don't include here the first two columns.

Table 7. Confusion matrix values for the proposed VGG-19 (V), InceptionV3 (I) ResNet-50 (R) and MobileNetV2 (M) models.

Precision				**Recall**				**F1-Score**			
V	**I**	**R**	**M**	**V**	**I**	**R**	**M**	**V**	**I**	**R**	**M**
0.93	0.96	0.98	0.97	0.92	0.95	0.9	0.95	0.92	0.96	0.94	0.96
0.9	0.97	0.95	0.95	0.88	0.94	0.93	0.93	0.89	0.95	0.94	0.94
0.88	0.9	0.89	0.9	0.87	0.92	0.93	0.94	0.88	0.91	0.91	0.92
0.91	0.95	0.93	0.94	0.92	0.95	0.95	0.95	0.91	0.95	0.94	0.94
0.88	0.9	0.91	0.91	0.89	0.92	0.9	0.93	0.89	0.91	0.91	0.92
0.89	0.94	0.98	0.96	0.93	0.94	0.96	0.97	0.91	0.94	0.97	0.96
0.92	0.94	0.93	0.93	0.93	0.94	0.94	0.96	0.92	0.94	0.94	0.95
0.88	0.89	0.89	0.88	0.86	0.93	0.92	0.93	0.87	0.91	0.9	0.9
0.91	0.92	0.86	0.95	0.76	0.88	0.92	0.9	0.83	0.9	0.89	0.92
0.89	0.9	0.92	0.94	0.85	0.85	0.88	0.91	0.87	0.88	0.9	0.92
0.93	0.95	0.95	0.97	0.88	0.91	0.93	0.9	0.91	0.93	0.94	0.94
0.9	0.94	0.96	0.93	0.93	0.92	0.94	0.96	0.92	0.93	0.95	0.94
0.82	0.87	0.85	0.88	0.81	0.88	0.89	0.92	0.82	0.88	0.87	0.9
0.84	0.85	0.86	0.9	0.86	0.89	0.9	0.93	0.85	0.87	0.88	0.92
0.94	0.94	0.98	0.96	0.88	0.95	0.93	0.95	0.91	0.95	0.96	0.96
0.99	0.98	0.99	0.99	0.98	0.99	0.97	0.97	0.98	0.99	0.98	0.98
0.87	0.94	0.96	0.95	0.95	0.96	0.93	0.96	0.91	0.95	0.94	0.96
0.95	0.93	0.92	0.95	0.9	0.91	0.92	0.92	0.92	0.92	0.92	0.93
0.94	0.95	0.96	0.96	0.93	0.95	0.94	0.96	0.93	0.95	0.95	0.96
0.9	0.97	0.93	0.97	0.93	0.97	0.98	0.97	0.92	0.97	0.95	0.97
0.94	0.98	0.96	0.97	0.92	0.96	0.95	0.98	0.93	0.97	0.96	0.97
0.95	0.95	0.98	0.98	0.94	0.96	0.95	0.95	0.94	0.95	0.97	0.96
0.93	0.97	0.96	0.97	0.93	0.96	0.96	0.97	0.93	0.97	0.96	0.97
0.91	0.93	0.94	0.97	0.9	0.95	0.95	0.98	0.9	0.94	0.95	0.97
0.95	0.95	0.95	0.97	0.93	0.94	0.96	0.97	0.94	0.95	0.95	0.97
0.89	0.96	0.94	0.93	0.89	0.94	0.94	0.95	0.89	0.95	0.94	0.94
0.93	0.93	0.92	0.96	0.89	0.9	0.92	0.9	0.91	0.92	0.92	0.93
0.94	0.93	0.93	0.96	0.96	0.96	0.96	0.96	0.95	0.95	0.94	0.96
0.93	0.94	0.93	0.97	0.9	0.94	0.95	0.95	0.92	0.94	0.94	0.96
0.83	0.9	0.89	0.92	0.91	0.93	0.93	0.94	0.87	0.91	0.91	0.93
0.98	0.99	0.96	0.99	0.97	0.98	0.99	1	0.98	0.99	0.98	1
0.78	0.93	0.9	0.91	0.96	0.92	0.93	0.93	0.86	0.92	0.91	0.92
0.95	0.98	0.97	0.99	0.88	0.94	0.97	0.94	0.92	0.96	0.97	0.96
0.97	1	0.98	0.96	0.95	0.97	0.97	0.96	0.96	0.98	0.98	0.96

When compared with some of the existing related works seen in Table 8, our solution has some advantages.

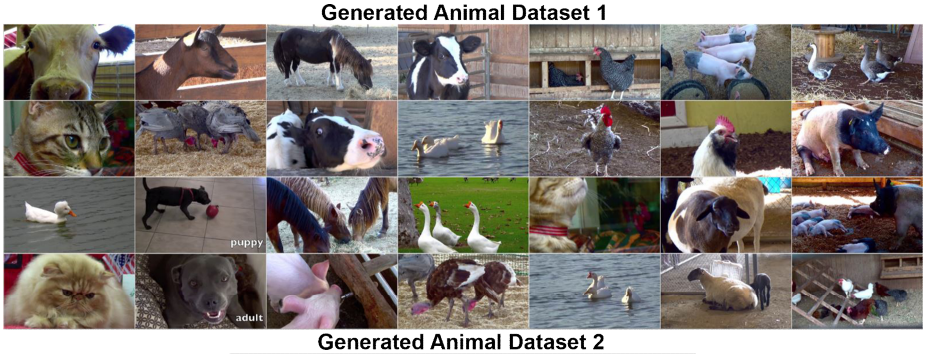
Table 8. Comparison between one of our 4 proposed CNN model architectures (MobileNetV2) and other related works.

Model:	[87]	[88]	[84]	[86]	[85]	One of our 4 proposed CNN	[89]

							model architectures (MobileNetV2)	
Number of Animal Classes:	20	20	3	6	50	20	34	24
Overall Test Accuracy [%]:	83.33	87.5	88.2	84.39	90.2	91.4	94.5	97.6

One of the main advantages is that the proposed system can identify animal classes not only from images but also in real-time from videos or using a webcam. The webcam animal class identification is very important because, even though most of the videos can be in high-definition, a real-life scenario in which the webcam operates can include shadows, dust, fog, and other weather conditions which can make the detection and identification task more difficult. Another advantage is that 2 new datasets are generated in real-time; one dataset containing textual information with the class name, date and time interval when an animal was present in the frame and another dataset containing animal images. An example of these datasets is presented in Fig.3.25.

Generated Animal Dataset 1



Generated Animal Dataset 2

Animal Class:	From:	To:
Cattle	2019-07-21 11:47:11 UTC	2019-07-21 11:48:41 UTC
Chicken	2019-07-21 12:03:37 UTC	2019-07-21 12:04:14 UTC
Duck	2019-07-20 13:07:45 UTC	2019-07-20 13:08:06 UTC
Goose	2019-07-21 14:47:48 UTC	2019-07-21 14:48:12 UTC
Horse	2019-07-20 14:05:37 UTC	2019-07-20 14:07:45 UTC
Pig	2019-07-20 15:32:45 UTC	2019-07-20 15:33:23 UTC
Sheep	2019-07-21 15:55:48 UTC	2019-07-21 15:56:07 UTC
Goat	2019-07-19 16:01:17 UTC	2019-07-19 16:20:52 UTC
Cat	2019-07-20 16:21:17 UTC	2019-07-20 16:21:49 UTC
Turkey	2019-07-21 11:01:17 UTC	2019-07-21 11:02:37 UTC
Dog	2019-07-19 16:01:17 UTC	2019-07-19 16:01:24 UTC

Fig. 3.25. Example of random animal images and their textual information generated in real-time by our proposed DL-based system from videos as well as using a webcam.

It is important to mention that these images were never seen before by our model and that they are automatically generated in real-time from videos or webcam. The size of each image generated in this new image dataset is 1280×720 pixels, meaning that they can be very easily analyzed by anyone later or used for further research, e.g. retraining a much more robust CNN model regarding animal classification and identification. Both datasets can be very useful also in the science of ecology, e.g. in order to monitor what animals are present the most, between what time interval and in which area. Additionally, the proposed solution can be used by ecology and biology scientists, veterinary professionals, animal or bird experts, farmers or anyone else who is interested in protecting their own safety or that of their domestic animals and crops.

4. POWERING A REAL-TIME DEEP LEARNING-BASED SYSTEM USING SOLAR ENERGY

Following we will present three subchapters that will cover the entire construction and testing of our dual-axis solar tracking equipment based on the Cast-Shadow principle that was later also improved and successfully used as part of a self-sufficient solar-powered real-time DL-based system.

4.1. Constructing a Dual-Axis Solar Tracking Device using the Cast-Shadow Principle

In a world affected by continuous resource depletion, the need to appeal to renewable energy solutions has become more and more justified. The Sun is one of the most available sources for harvesting solar energy and can be exploited successfully with the help of PV panels.

Solar tracking systems that rely on the PV effect are constantly expanding, even in geographical areas that do not have a lot of sunlight at their maximum level over a year period. Because of the affordable price and the existence of highly-efficient solar cells [166] like the solar modules based on silicon, which deliver more than 20% performance, PV panels have become an attractive choice, especially to homeowners. The maximum energy output of solar cells with silicon is based on current and voltage monitoring. Modifications in current-voltage charts via solar module heating or variable intensity illumination may be important causes of efficiency loss in solar generators, thus resolute energy cannot be produced when the sunlight is constantly changing. In order to conserve energy savings, electronic circuits became available on the market, being used for MPPT as well as to prevent unproductive temporary modules from interrupting the production of active cells. Solar panels have maximum efficiency under ideal conditions: e.g. when the electrical power is generated by illumination near the Equator on a serene day as well as when a square meter of the Earth's surface receives more than 1kW of power from the energy of the Sun. However, environmental conditions or mismatches of electrical characteristics of PV panels can reduce the overall system efficiency.

4.1.1. Position Optimization Method

Usually, a full daylight cycle is described by a 150-degree rotation of the Sun around the horizon whereas a year period is outlined by a 46 degrees movement of the star from the north to the south direction [167]. Static PV panels are physically not able to capture the maximum potential of solar energy mainly because of their generally fixed 45 degrees installation angle and while the Earth travels around the Sun, certain regions of the solar panel become shaded and thus provide a minor or no contribution to the power output of the solar system. Taking

into account that cells act as photodiodes as well as the obvious effect of the dark areas on the panel, we employed a novel tracking mechanism based on the Cast-Shadow principle for which we used monocrystalline solar cells.

Initially, monocrystalline PV cells were placed on a 4 mm thick polycarbonate plate. The cells were serially linked, 10 units on each row, so that we obtained two groups of 20 photocells which generate 10V. Once tied in a serial manner, the two groups of PV cells have been tethered in a parallel connection, resulting in a double amount of current produced by just one photocell. In the corners of the solar panel, four protective wings were mounted in order to serve as a screen for the PV cells placed at the extremities of the payload. As it is depicted in Fig.4.1, we selected a group of 3 cells from each corner which later will have the vital role of analyzing the light distribution in their respective locations.

Each of the 40 PV cells is capable of providing a voltage of 0.5V and a power output of 0.3W. According to the formula in (4.1), the current value can be effortlessly determined.

$$I = P / V \tag{4.1}$$

When irradiated by sunlight, the associated current which traverses a solar cell is I=0.6A. Shaded PV cells will drastically reduce energy production if they are not effectively managed. Shading less than 3% of the solar panel surface may reduce output efficiency by more than 15%, according to US National Renewable Energy Laboratory [168]. The efficiency of a solar cell decreases with temperature increase: 0.15-0.25% / 1° C for amorphous silicon, 0.35/1° C for monocrystalline silicon and 0.5% / 1° C for polycrystalline silicon. The temperature of a polycrystalline solar panel for instance, during summer, in plain areas reaches easily 50° C, resulting in a 12.5% reduction in power output compared to 25° C. This reduction in efficiency is important; the conclusion is that high amounts of sunlighting during summer does not produce the maximum current, except in cold areas. We considered this research direction an opportunity to investigate how heat generation from the sun rays can alter the efficiency of our solar cells.

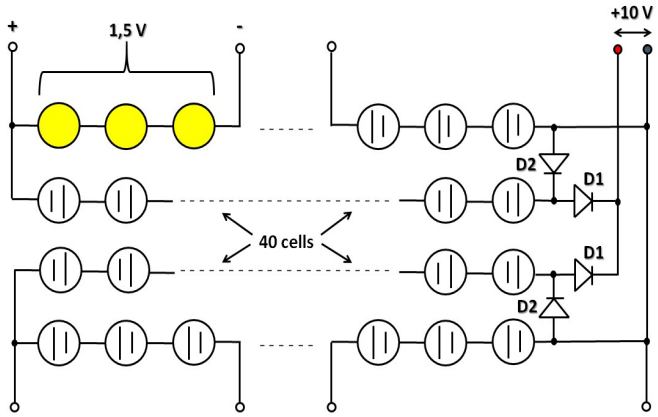


Fig. 4.1. Electrical Connection Scheme for PV modules with the added bypass diodes.

With the help of an infrared thermometer and a multimeter, we measured the voltage-temperature relation for monocrystalline solar cells and represented it on the diagram in Fig.4.2.

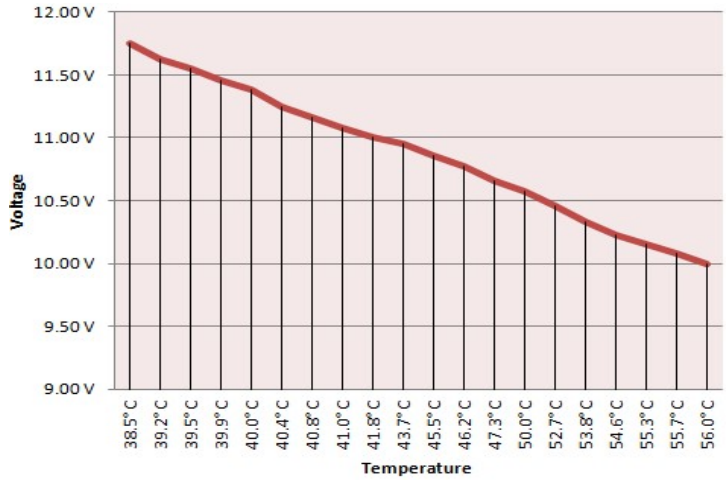


Fig. 4.2. Heating Effect on Monocrystalline Solar Cells.

A number of 20 measurements have been carried over a sampling time t = 3 minutes to establish the average temperature of the solar panel for each data point. Starting from an initial value of 11.75V at 38.5° C, considerable voltage drops have been monitored until the temperature has reached a peak point of 56° C where the multimeter has registered 10V. According to the degradation coefficient of PV modules, the average efficiency loss is 10.85% for monocrystalline, respectively 15.5% for polycrystalline cells. Due to rising temperatures resulting directly from sunrays, overheating issues affect unavoidably the solar cells by causing continuous voltage drops. Consequently, future investment in a hybrid PV system given by the combination of a solar panel and a water-cooling mechanism could prove favorable in enhancing the performance of PV cells by almost 50% [169].

However, in the absence of any cooling solutions, shaded or defective cells are circumvented by the current of illuminated solar cells. To avoid overheating issues and ensure that PV modules operate reliably, bypass diodes, denoted with D2 earlier in Fig.4.1, can be supplemented. The function of this diode is to protect PV cells if the light on the surface of a module is not uniform. On the other hand, the blocking diodes, indicated with D1, are responsible for protecting each PV cell string of reverse current from other PV cell strings located on the solar panel, usually caused by shading on only one PV cells row. Bypass diodes are typically placed on sub-strings of 20 PV cells. Since the PV cells are connected in series, power differences cause also voltage differences. If the conduction of high current is initiated by a shadowed cell, its voltage is actually negative. Instead of producing energy, this solar cell will only consume it, thus becoming a reverse polarized diode that dissipates power and which will cause itself to heat up. If the area, the structure, and the environmental conditions do not allow proper heat dissipation, a critical power point is reached called hot-spot, which interrupts the row of solar cells. The exact point at which a PV cell becomes a consumer instead of being an energy producer differs by types of cells and diodes. Despite successfully fulfilling the PV cell protection function, bypass diodes are not effective in reducing temperature increases in PV cells.

4.1.2. Performance Evaluation of Electrical Equipment

With reference to commercial solar panels, the key component that ensures efficiency is the inverter that generates power compatible with the AC power grid. In the case of solar panel networks, inverters are complex systems, which normally have three functions: DC / DC conversion, DC / AC conversion, and Anti-Island control. The inverter must be adjusted to the changing conditions in the solar cell matrix. This is generally done using the MPPT algorithm, in other words, maintaining the product voltage × the current at maximum values. Using MPPT, inverter circuits can use the optimal combination of voltage and current, delivering power to a load in an effective way. In this perspective, our new Cast-Shadow concept helps to maintain the solar panel perpendicular to the Sun streams without using the MPPT method.

The solar tracking device, as seen in Fig.4.3, is materialized from an Arduino UNO board, two Stepper motors, a pair of specialized L298N circuits and an Optocoupler, which will be further detailed in this section.

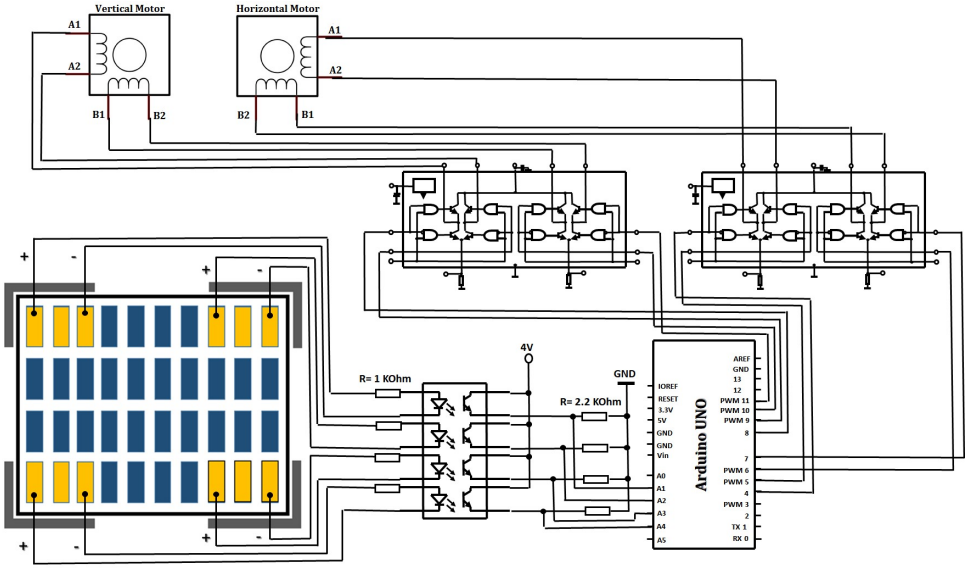


Fig. 4.3. Schematic Overview of the Solar Tracking System.

In order to achieve a low-cost solution, we aimed for the Arduino UNO, a development board based on the architecture of the Atmega328 microcontroller. Also, in order to design a mobile variant of a solar panel, we made use of 4 analog inputs (A1-A4) for collecting sensor information from the solar cells, 8 digital outputs (D4-D11) for commanding sequentially the Dual H-Bridges, 4 digital pins, from which 2 of them (D2-D3) were assigned as inputs for the upper and lower switch and the other pair (D12-D13) for a blocking circuit to reduce the power consumption of the Stepper motors.

The Arduino microcontroller can be powered in two ways, either via a USB connection from the computer or via a battery to which the positive terminal will be connected to the V_{in} input and the minus terminal will be linked to the ground. The board was designed to operate between 6V and 20V, but the recommended voltage

range is between 7-12V. In our automation project, the Arduino UNO drains 350mA during standby phases and 380mA while sending commands to the output circuits, resulting in average power consumption of 0.15 Watt-hour (Wh)/day.

Stepper motors are brushless DC electric motors capable of divaricating a 360-degree rotation into a number of identical steps. In our work, we used a unipolar EM-61 23LM-C352 stepper motor for horizontal operation and a bipolar 103G771-0240 stepper motor for executing vertical movements, recycled from old printer models. The unipolar motor differs from the bipolar model by having a common center tap per phase, which will be linked to the positive valence of the H-Bridge circuit. In most cases, given one phase, the central tap for each winding has the following arrangement: 3× phase-conductors and 6× conductors for a regular two-phase stepper motor. These types of stepper motors offer a cheap solution for precise angular movements. Bipolar stepper motors, on the other hand, have only one winding per phase. Generally, Dual H-bridges are the circuits of choice to change the current in the winding, which in turn reverts the magnetic pole causing the stepper motor to move in one direction or another. Further technical information for both stepper motors can be seen in Table 9.

In many of the studied related works, stepper motors are not the primary engine of choice in order to move the solar panel in different directions. This is mainly because other actuators such as DC motors can fulfill the same task for less power consumption [170]. We followed this premise and carried out a series of measurements on our stepper motors. In order to establish the current drain for just one stepper motor, we disconnected the vertical motor from the solar panel installation.

Table 9. Technical Data for Stepper Motors.

Parameter	**Horizontal Stepper Value**	**Vertical Stepper Value**
Nominal Voltage	4 V	1.53 V
Current Intensity	1.5 A	3 A
Resistance	3 Ω	1 Ω
No. of terminals	6	4
No. of steps/revolution	200	200
Weight	450 g	552 g
Angular resolution	1.8 °	1.8 °

After careful measurements, we determined that a maximum value of 3A was drained in the standby phase while the stepper motor is resting and keeps the solar panel tightly fixed at the given position. At this point, we could only reduce the supply voltage to 4V to obtain 2A while the stepper remained fully functional. The average current of 1.5A is only visible during its duty cycle, therefore steppers consume less current when they rotate the panel in a certain direction. Consequently, the used stepper motors have a much higher power specification of 5W compared to DC motors with the 2.4W power rating, which results in an overall efficiency decrease of the employed system. Despite this fact, we managed to cover these disadvantages by improving both engine accuracy and energy consumption.

First of all, as can be seen in Fig.4.4, we constructed a gear train by fixing gears on a frame so that the teeth of the wheels interact with each other.

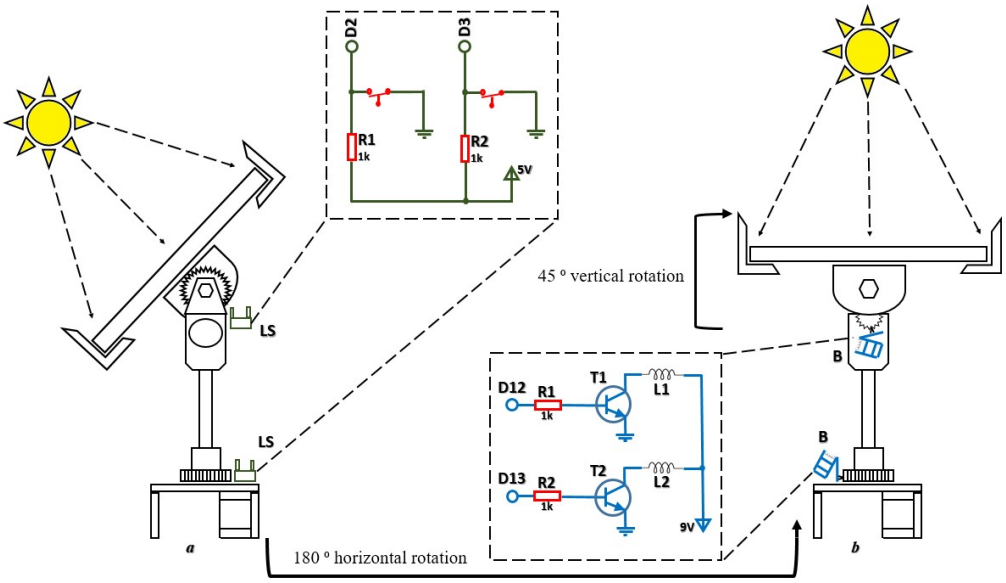


Fig. 4.4. Mechanical System of PV Panel with enhanced Stepper Motors.

The mechanical advantage for a gear wheel is often defined as the ratio of the number of driving gear teeth divided by the number of driven gear teeth (load). That means the gear ratio known also as speed ratio is inversely proportional to the pitch circle radius and the input gear's number of teeth. In mathematical language the formula can be written as in (4.2):

$$R = Nb / Na \tag{4.2}$$

where *Nb* represents the number of teeth of the input cogwheel and *Na* signifies the number of teeth from the output gear.

According to Table 9 seen earlier, both stepper motors have a standard angular resolution of 1.8 degrees/step. We counted the number of teeth for the larger and smaller-sized cogwheel and obtained *Na*=85, respectively *Nb*=21. By using the above formula in (2), the resulted mechanical advantage was *R*=1/4, which led us to an improved angular resolution of 1.8/4=0.45 degrees/step. In respect to the vertical motor, by applying the same mathematical relation for the values *Na*=85, *Nb*=25 we obtained *R*=1/3.4, thus resulting in a new angular resolution of 1.8/3.4=0.52 degrees/step.

Secondly, our Stepper motors are improved to rest outside of their duty cycle, hence remaining competitive in terms of reliability and power consumption. Steppers are known for two major drawbacks: the lack of position feedback and high energy drain during stationary periods. To compensate for these disadvantages, as seen in Fig.4.4, we added four extra elements to the mechanical structure of the solar panel: two switches (denoted with LS) and a pair of blocking elements (denoted with B). The upper switch which engages with the vertical motor has the role to restrict the elevation movements, while the lower switch restrains the azimuth rotations of the horizontal motor. The lower switch also initiates the rotating command of the solar panel in the original state it was in. From the electric diagram presented earlier in Fig.4.3, we can see that the circuit is equipped with

two resistors with a value of 1kΩ. If one of the two inputs (D2 and D3 from Fig.4.4) activates one limiter, its value will change to HIGH (under-voltage) and thus will stop one of the two-axis movements. If none of the limiters are validated, the two inputs will have the value LOW, both being linked in this case to the ground point.

To initiate the braking on the toothed wheels of the stepper motors, a circuit consisting of two resistors, two transistors, and two coils was used. While the 1kΩ resistors are intended to limit the current to the transistors whenever a voltage is applied, the electric junction opens, thus activating the coil which will release the locking element on the cogwheel. In the opposite scenario, when the coil is not under voltage, the stepper motor will be deactivated with the *digital.Write()* function of the Arduino, which implements LOW values on all inputs of the Dual-H bridge driver. In this manner, the stepper motor will maintain its current position due to the blocking element feature and will not consume any power during the stationary regime.

The two stepper motors are driven by the L298N circuits by sequentially commanding the diagonal of the two bridges. This type of circuit encompasses a double bridge of transistors, of which a bridge feeds a motor winding and the other bridge, the second winding. The Dual-H Bridge is provided with Schottky diode outputs, designed to protect the IC from the auto-inductive voltages that can occur at engine windings. The role of the onboard capacitors is for additional filtering of voltages. When we measured the current draw from the L298N board, we obtained 400mA during standby phases and a maximum of 450mA while transmitting impulses to the stepper motors. The overall power consumption of these components is rated at 0.17Wh/day.

Regarding the Optocoupler, the IC is called LTV 847 and has in its structure a number of 4 Optocouplers, which in turn are made up of 2 components: a Light Emitting Diode (LED) with the opening voltage of 1.2V and a phototransistor that opens when it receives light from its corresponding LED. The role of the Optocoupler is to galvanically isolate the input source from the output source. The collected voltages from the 4 corner groups of PV cells don't have a common connection point, therefore we resorted to the Optocoupler. This is because we were able to connect all 4 resistors of the phototransistors to the ground point of the PCB. According to measurements, each of these 4 groups has a current consumption of 0.6mA. Therefore, the Optocoupler that is connected to the selected array of cells shows a negligible power consumption, more exactly 0.95 mW, compared to the total power discharged by the solar panel.

The average current that is drained by the Optocoupler inputs is equal to 0.5mA as shown in Table 10, resulting in average power consumption of 0.02W, while the outputs, which are rated at 2.2mA develop an average energy drain of 0.088W.

Table 10. Optocoupler – Arduino Pin Connections.

Crt. Nr.	**Optocoupler - Arduino – Connections and Values**			
	Inputs	**Measured values [V]**	**Outputs**	**Measured values [V]**
1	A1-K1	0.567	A1	1.506
2	A2-K2	0.568	A2	1.332
3	A3-K3	0.561	A3	1.510
4	A4-K4	0.550	A4	1.384

To limit the current through the diodes, 1kΩ resistors have been mounted at the input of the Optocoupler. In order to adjust the voltage values that are sent to

the analog inputs of the Arduino (A1-A4), variable resistors have been fixed in the output circuit of the LTV 847.

In respect to the Fig.4.5, we can distinguish a high power drain range of 4.55 – 9.40W for active stepper motors during the entire day, and relatively low power consumption, between 0.22 and 2.39W for inactive steppers.

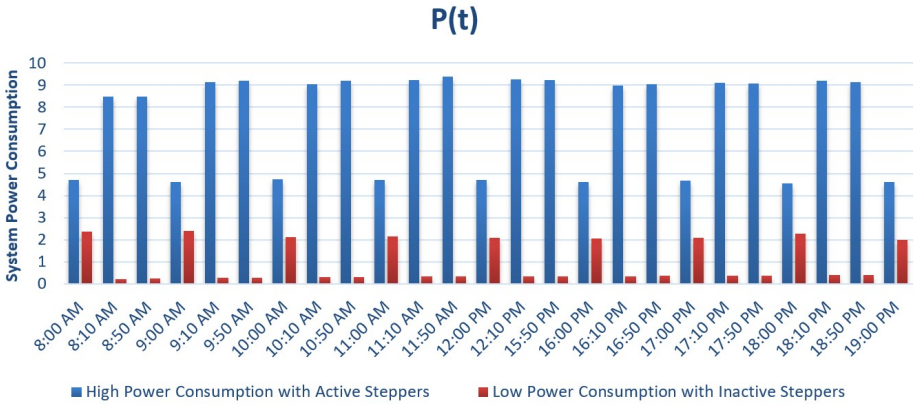


Fig. 4.5. System Power Consumption of the Solar Tracking Device.

In other words, the solar tracking device consumes an average of 7.50 Wh/day with stepper motors under voltage and 0.98W while keeping the steppers turned off between exact hour times. In this manner, we were able to decrease the global power consumption of the portable solar panel by 86.93% per day. This reduction in energy consumption was only possible by mounting an additional 2.4Ω close to the V_{in} of the L298N Dual-H bridge and based on the careful observation that the solar tracker only needs to update its position every hour per day. While there may be gaps between the data points of the graph, usually electrical components maintain a standby phase power consumption, as it is depicted in Fig.4.6.

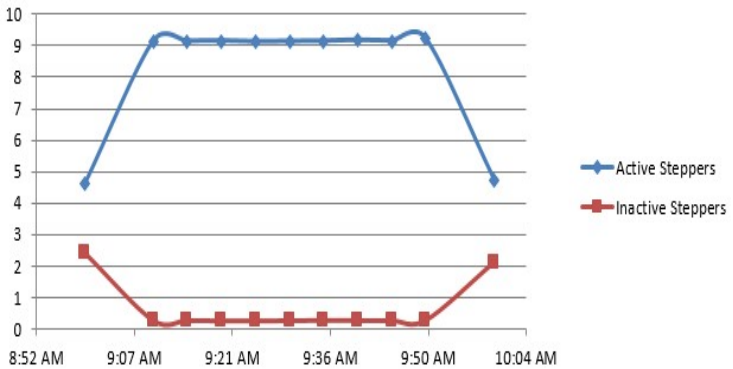


Fig. 4.6. Solar System Standby Power Consumption between Hour Times.

In this scenario, experiments have been carried out over an hour time with a 5 minutes space between each measurement to demonstrate that deactivating both stepper motors can drastically reduce the power consumption. In combination

with the braking method, steppers have to be programmed adequately to be under voltage before the blocking element releases the cogwheel. This is an important aspect of the approach because we avoid any risk of losing the current position of the Stepper.

4.1.3. Algorithm Testing

The software code we developed on the Arduino platform eliminates the requirement of sophisticated mathematical approaches presented in [92] where the authors rely on geometrical formulas to calculate the theoretical altitude and azimuth angles of the Sun's position. A simplified model of the pseudocode which was the basis for the implemented automation program can be visualized in Fig.4.7.

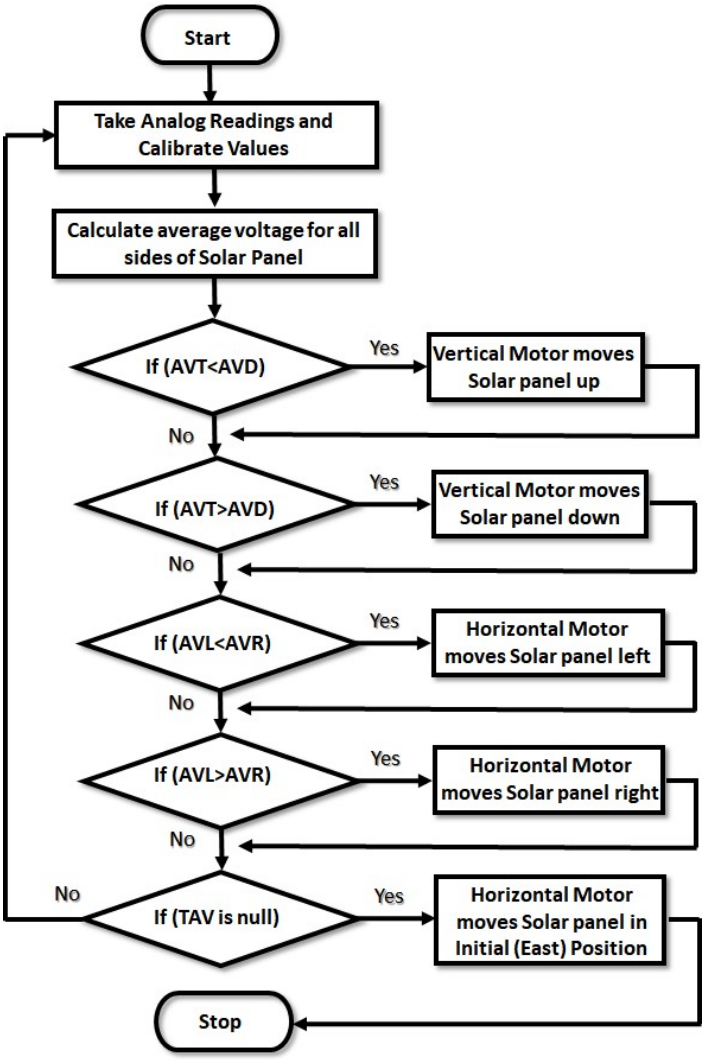


Fig. 4.7. Logical Flowchart for PV Panel Algorithm.

In order to gain a better understanding of the working principle, each solar cell group from the 4 corners will be designated as TR (Top Right), TL (Top Left), BR (Bottom Right) and BL (Bottom Left) in linkage with their locations. We will also consider the average voltages for each side as AVR (Average Voltage Right), AVL (Average Voltage Left), AVT (Average Voltage Top) and AVD (Average Voltage Down), while the total average value will be denoted as TAV.

The solar tracker will generally have an initial eastward position before the algorithm enters its normal routine. After the sun rises, incoming voltage values received from the Optocoupler will be temporarily stored in newly declared variables that require additional in-program calibration. At this point, with the support of formulas from the *math.h* library of the Arduino IDE, average values will be calculated for all sides of the solar panel. Although it is not mentioned, before the algorithm checks each *if* condition to rotate the solar panel in the correct direction, it usually verifies if the total average voltage of the 4 corners is equal or less than a predefined number (for instance 8) and it also checks if the limit switches have the required range. This process is repeated for both stepper motors and guarantees that analog readings are always up-to-date. The direction in which the solar panel moves is generally determined by the presence of shadow on one of the pairs of cell groups. Whenever the algorithm detects a major difference between voltage averages, it will rotate the panel towards the direction of the shaded area. At the end of the day, when all voltages on the 4 corners become null, the solar tracker will drive the engine horizontally to its initial position waiting for the Sun to rise once again the next morning. However, altered values that may be delivered to the Arduino microcontroller can modify the paths of the algorithm and by default disrupt the orientation of the PV panel. In such circumstances, an error handling software is the appropriate solution for preventing communication and calculation errors to appear in the system.

A White-box testing strategy, which we developed in [18] counteracts the intrusive behavior of these common types of software errors by injecting random values that simulate gathered voltage readings from each group of 3 solar cells. The designed testing algorithm implements different testing techniques to evaluate the software functionality and features. Each part of the code will be tackled individually and we attempt to show all the possible breakpoints as well as try to detect possible fault errors using White-box testing techniques. The testing is based on the AUnit [171] Arduino library which is a port of ArduinoUnit and Google Test programs.

The White-box testing routine will follow the program structure of the algorithm under test as seen in Fig.4.8 and will implement a set of continuous tests to check regularly on error handling errors. From this perspective, the algorithm will require the user to choose the desired testing path. If the device is set in Field mode we can read control data received from analog sensors. In the opposite case, Test mode, we will only be able to receive control instructions from our Message Queuing Telemetry Transport (MQTT) server [172]. This selection process is mainly necessary in order to ensure that the data being used to control the stepper motors is solely from one source. From here, the testing algorithm will fetch the incoming injected inputs from an ESP8266 Wi-Fi module which sends data wirelessly, and further proceeding with the calibration of the volatile entries. The first test that will run in a continuous loop will verify each of the analog sensor readings in order to make sure that they are within the specified range. Usually, large data packages, which are a source of calculation errors, can be detected successfully by the test program and isolated for further analysis.

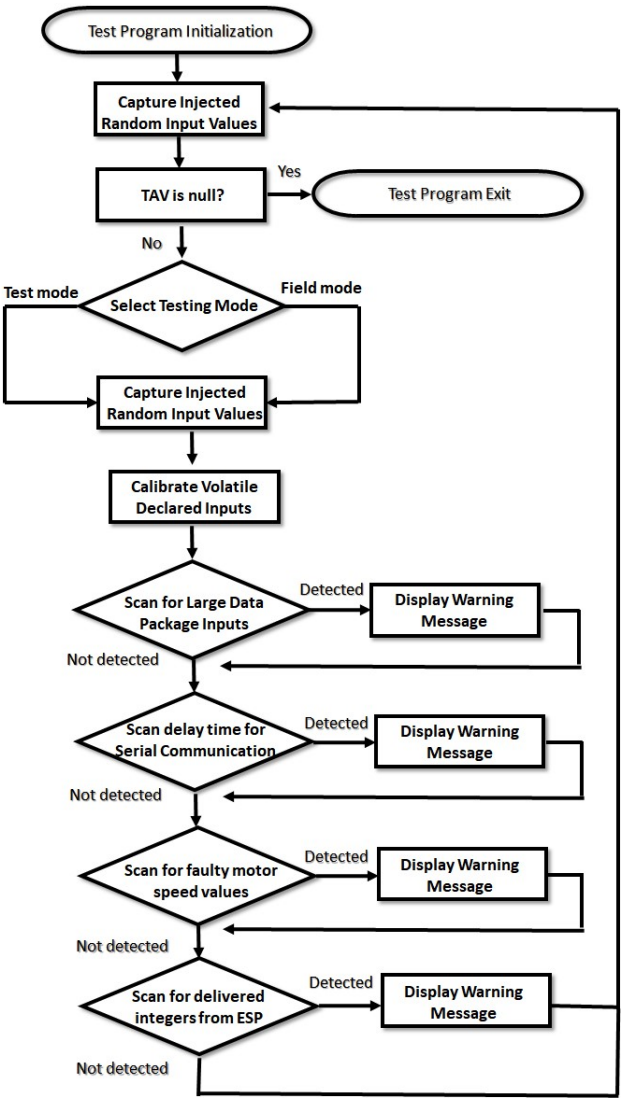


Fig. 4.8. Logical Flowchart of Algorithm based on the White-box testing approach.

Although the values for the initial speed of the stepper motors are hard-coded and there is little room for errors, in a setup where those values are selected dynamically, they can cause an increased number of steps which leads to improper displacement of the solar tracker. In some cases, serial communication may not start fast enough. This will lead to data corruption and communication errors whenever we are trying to receive or send data to the server. A continuous scan on serial communication may warn the user if the connection to the server has failed.

There is also the possibility of declaring a long variable that holds the last time we sent analog data to the server. We can use this variable to control the frequency of data transmission to the server. Sending data continuously can lead to over tasking while the server script that processed the received data can also cause

framing errors in the Serial Communication. In any other case, a last possible breakpoint in the program is given by the function which receives data from the ESP device. When using Serial Communication at higher speeds we have to run a test that will ensure that the received values are valid integers and we can also check for the length of the received data, since we are expecting a specific number of characters from the ESP.

4.1.4. Experimental Results Regarding Position Optimization

The efficiency of solar energy conversion is the percentage of solar energy that is converted into electricity. This is calculated by the ratio between the maximum power (*Pm* - Maximum Power in W) at the outlet and the input light (*E*, in W / m^2) and the solar cell surface (*Ac*, in m^2) as seen in formula (4.3):

$$E = Pm / (E \times Ac) \quad (4.3)$$

By convention, the efficiency of a solar cell is measured under standard test conditions (STC), i.e. at a temperature of 25 ° C and an irradiance of 1000W / m^2 with an air mass AM 1.5 spectra [173], which defines the solar radiation that traversed the atmosphere. These conditions correspond to a sunny day with sunlight on a 37 ° inclined surface facing the sun and when the sun is at an angle of 41.81 ° above the horizon. This is equivalent to sunlight at noon close to spring and autumn equinoxes in the continental area of the United States [174], with the surface of the cell directly targeted by the sun. Under these test conditions, a solar cell with an efficiency of 20% and a surface area of 100 cm^2 (0.01 m^2) would produce a power of 2W.

While our measurements for power consumption were made in August 2018 and the power generation of the solar panel was overlapping between 12:00 PM and 15:00 PM, in order to save energy, we decided to temporarily switch-off the solar tracking device. However, September 2018 was the auspicious month for us because air temperature would drop down to 25-28° C and solar irradiance would be much closer to standard values. Measurements were focused mainly on voltage, current, and power monitoring during a full-day cycle, with a clear atmosphere and an average temperature of 27° C.

The last 6 columns from Table 11 illustrate the voltage and current values registered by the multifunction tester in the following order: VSP (Voltage Static Panel), VAP (Voltage Automated Panel), VSPC (Voltage Static Panel with Consumer), VAPC (Voltage Automated Panel with Consumer), CSP (Current Static Panel with consumer) and CAP (Current Automated Panel with consumer).

The columns regarding VSP and VAP from the Table 11 are not pertinent for the efficiency of the solar tracking device as the average open-circuit voltage increase is located between 8.5% - 9% or above, depending on the environmental conditions. However, after we mounted a 10Ω resistor in the output circuit to serve as an energy consumer we could determine the following parameters: VSPC, VAPC, CSP, and CAP.

According to the collected values, the solar panel generated an average of 5.35 Wh/day in a static position and 8.22 Wh/day while it was tracking the Sun's trajectory resulting in a 53.64% power increase/day. To demonstrate the relevance of our experimental research we continued the measurements for another 6 days to reach a full week.

Table 11. Voltage and Current Monitoring for Static and Automated PV Panel.

Time (Hour)	VSP [V]	VAP [V]	VSPC [V]	VAPC [V]	CSP [A]	CAP [A]
8:00	9.30	11.50	2.00	7.20	0.130	0.835
9:00	10.80	11.60	4.74	8.58	0.497	0.998
10:00	11.12	11.80	6.12	9.15	0.785	1.107
11:00	11.15	11.77	8.35	9.80	0.940	1.090
12:00	11.33	11.70	9.05	9.23	1.065	1.190
13:00	11.46	11.75	8.82	9.31	1.076	1.190
14:00	11.40	11.68	8.63	9.10	0.880	0.918
15:00	11.15	11.45	8.31	9.00	0.830	0.910
16:00	10.67	11.38	8.23	8.70	0.780	0.890
17:00	10.54	11.10	6.60	8.10	0.650	0.830
18:00	10.20	10.84	5.27	7.38	0.580	0.740
19:00	7.35	10.10	4.30	6.95	0.360	0.680

As can be seen in Table 12, the third day was the most productive in terms of generating voltage, current, and power for our energy production. We mention that the Power of the Static panel with Consumer (PSC) and the Power of the Automated panel with Consumer (PAC) were subsequently calculated.

Table 12. Voltage, Current and Power Monitoring for Static and Automated PV Panel (over one week).

Test Schedule	VSPC [V]	VAPC [V]	CSP [A]	CAP [A]	PSC [W]	PAC [W]
Day 1	5.93	7.97	0.561	0.746	3.32	5.94
Day 2	6.41	8.70	0.578	0.824	3.70	7.16
Day 3	6.70	8.54	0.714	0.948	4.78	8.09
Day 4	6.63	8.55	0.649	0.875	4.30	7.48
Day 5	4.66	7.72	0.479	0.834	2.32	6.43
Day 6	4.26	7.88	0.493	0.845	2.10	6.65
Day 7	4.45	7.55	0.552	0.895	2.45	6.75

With reference to Fig.4.9, the performance gain of our solar tracking solution covers voltage, with an average of 45.77%, current, with a mean value of 48.21% and lastly 53.62% more power generation compared to the fixed-tilted solar panel during one week period.

A summarized analysis regarding system power consumption and energy gains of our proposed dual-axis solar tracking device based on Cast-Shadow principle versus state-of-the-art automated PV panels is illustrated in Table 13. We present a comparison with other related works regarding the Wh/day consumption before (red color) and after (orange color) optimization as well as the daily and monthly energy improvements given by current (blue color), voltage (violet color) and power (green color). According to the experimental data from Table 13, our proposed solar tracking solution based on the Cast-Shadow principle outperforms the works in [92-95, 170, 175, 176, 178] with regard to the power increase, the work in [91] regarding the overall voltage increase as well as the work in [90] for all targeted energy areas.

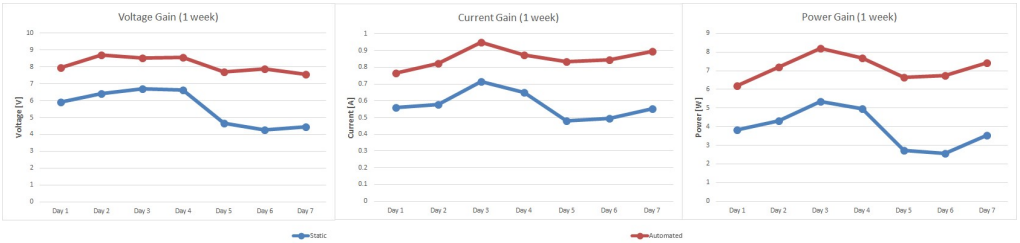


Fig. 4.9. Voltage, Current and Power Gain of Automated Panel (red color) over Static Variant (blue color).

Additionally, it is important to mention that our solution outperforms also the work in [177] regarding power gains for the dual-axis implementation, but due to the multidirectional approach, their solution is more efficient.

Table 13. Energy Gain Analysis for Solar Tracking Devices.

<table>
<tr><th rowspan="2">Crt. No.</th><th rowspan="2">System Components</th><th colspan="2">System Power Consumption (Wh/day) per Day [D]</th><th colspan="3">System Energy Gains (%) per Day [D] and Month [M]</th></tr>
<tr><th>Current</th><th>Voltage</th><th colspan="2">Power</th></tr>
<tr><td rowspan="2">[17]</td><td rowspan="2">Arduino UNO + Optocoupler + L298N + Steppers (Unipolar+Bipolar)</td><td>Before Optimization</td><td>After Optimization</td><td rowspan="2">48,21 [D]</td><td rowspan="2">45,77 [D]</td><td colspan="2" rowspan="2">53,62 [D]</td></tr>
<tr><td>9 [D]</td><td>2 [D]</td></tr>
<tr><td rowspan="2">[92]</td><td rowspan="2">PC + Central Processing Module + Sensor Modules + Motor Driver circuits</td><td>Normal Tracking</td><td>Daily Adjustment</td><td colspan="2" rowspan="2">Not specified</td><td>Normal Tracking</td><td>Daily Adjustment</td></tr>
<tr><td><= 52,8 [D]</td><td><= 1,2 [D]</td><td>>= 23,6 [D]</td><td>>= 31,8 [D]</td></tr>
<tr><td>[94]</td><td>Arduino UNO + LDR’s + Motor Drivers + Servo Motors</td><td colspan="2" rowspan="4">Not specified</td><td>53,35 [D]</td><td>Not specified</td><td colspan="2">52,69 [D]</td></tr>
<tr><td>[170]</td><td>Arduino UNO Microcontroller + Servomotors + LDR’s</td><td colspan="2">Not specified</td><td colspan="2">13,44 [D]</td></tr>
<tr><td>[91]</td><td>Arduino UNO + Servomotors + LDR’s + SD card + Battery</td><td>Not specified</td><td>36,30 [D]</td><td colspan="2">Not specified</td></tr>
<tr><td>[93]</td><td>Atmega328P + LDR’s + Servomotors + Panel Carrier</td><td colspan="2">Not specified</td><td colspan="2">42,81 [D]</td></tr>
</table>

[90]	Arduino UNO + LDR's + DC motors		16,59 [D]	40,66 [D]	35 [D]	
[95]	MC68HC11A8 + INA168 + OPTO-DIAC + AMIS -30543 Motor Drivers + NEMA 23 Steppers		**Spring**		33,6 [M]	
			Summer		43,6 [M]	
			Autumn		38,3 [M]	
			Winter		28,8 [M]	
[175]	MATLAB Simulation of Solar Tracking Design Model		Not specified		33,37 % [D]	
[176]	Solar Panel + LDR Sensors + DC motors + H-Bridges + Regulator	~ 4,07 [D]			>= 24,78 % [D]	
[177]	Sensor Matrix Array + Dual Comparator + Inverter + Microcontroller + Driver Circuit + Stepper Motors + Monocrystalline Solar cells	Not specified			**Dual Axis Solar Tracker**	32,34 % [D]
					Multidirectional Solar Tracker	63,96 % [D]
[178]	Klipp & Zonen Pyrheliometer + NI My RIO + MEMS dual-axis tilt sensor + DC motor Driver + Linear motors for altitude and azimuth				28 % [D]	

A more critical discussion between the works is difficult to achieve due to the following facts: a) each solar tracker is unique in regards to the system components that can be mechanical parts (cogwheels dimensions, gearbox arrangement) which influence the precision of the device, as well as electrical components (ranging from microcontrollers, FPGA's, ASICS to more complex systems such as CPU architectures) that determine the accuracy and speed of the implemented algorithms; b) many of the recent works primarily focus on the energy efficiency without mentioning the power consumption of the implemented devices. The effect of tracking on the PV performance can be measured by analyzing the energy produced by the fixed and automated panel as well as the energy consumption necessary for the tracking; c) the tracking strategies do not hold the highest impact over power generation. A more critical factor is the quality of the used PV cells, as certain categories are rated to provide more energy efficiency. In

our case, we focused on constructing a low-cost solar tracking device. Future investment in a more professional equipment may lead to even greater performance.

4.2. Software and Hardware Testing of a Dual-Axis Solar Tracking Device

Having described the **construction** of the dual-axis solar tracking equipment, this subchapter focuses on presenting **software and hardware testing methods** applied to it. Despite there being, to the best of our knowledge, no software or hardware testing methods applied to a solar tracking equipment before in the literature, the detection of possible software or hardware errors present during its operation was very important because: a) we later use the dual-axis solar tracker in order to power a real-time DL-based system, as will be detailed later in the next subchapter of this Ph.D. thesis and b) we anticipate an increase in the availability as well as in the importance of the solar tracking equipments (e.g. when powering DL-based systems).

4.2.1. White-Box Testing Applied to our Dual-Axis Solar Tracker

With entry into the digital age, the need to use software testing strategies for devices from the renewable energy field has become equally important as hardware testing facilities. In order to obtain maximum coverage of possible occurring defects, it is required to evaluate the functionality of the software code that runs on a digital device. As mentioned earlier, solar tracking devices are programmed PV panels which are able to direct the payload towards the Sun for optimal solar radiation exposure and automated PV panels appear in two forms: single-axis and dual-axis solar tracking devices [179], the second form being more advantageous from the accuracy point of view. However, the complexity of the algorithm which commands the direction of the Sun tracking device grants accesses to intrusive errors that can hinder maximum solar energy gathering, thus affecting the overall efficiency of the solar installation. Therefore, software-testing strategies are a feasible and low-cost solution to this problem, allowing the programmer to design test cases for the written code and verify it's functionality in more critical scenarios. Testing of software and hardware solutions are in high demand, as the need for highly secure applications and systems is increasing. Being the most challenging and dominating activity in the industry, the purpose of testing is to provide quality assurance, verification, and validation, in order to ensure software quality. White-box testing [180], [181], contrary to the Black-box testing which relies on testing from an external or end-user type perspective, involves the testing of internal coding and infrastructure of a software application by focusing mainly on strengthening the security and the flow of inputs and outputs through it, resulting in an improved design and usability of the AUT. White-box testing requires the tester to have very good knowledge of the programming language in which the AUT was written on as well as knowledge of how the system is implemented. This helps in minimizing the costs by reducing testing time and also to minimize the errors.

Our White-box testing strategy covers multiple types of errors that can occur on the control board, which is implemented using an Arduino Uno, as well as to the components around it. Here, we can distinguish four main aspects of the testing phases: first, we are injecting virtual random integer values to capture analog faults of the solar tracking equipment; secondly, we are gaining direct control of the stepper motors to receive feedback analog readings from each of them; third, we are using the White-box testing strategy to verify the internal structure of the ATmega328 microcontroller on the Arduino UNO, with regard to memory errors and buffer overflows. As it is depicted in Fig.4.10, the secondary microcontroller, which is called ESP8266, serves as a middleman between the Arduino Uno platform and the Cloud setup because the DUT lacks inbuilt Wi-Fi capabilities.

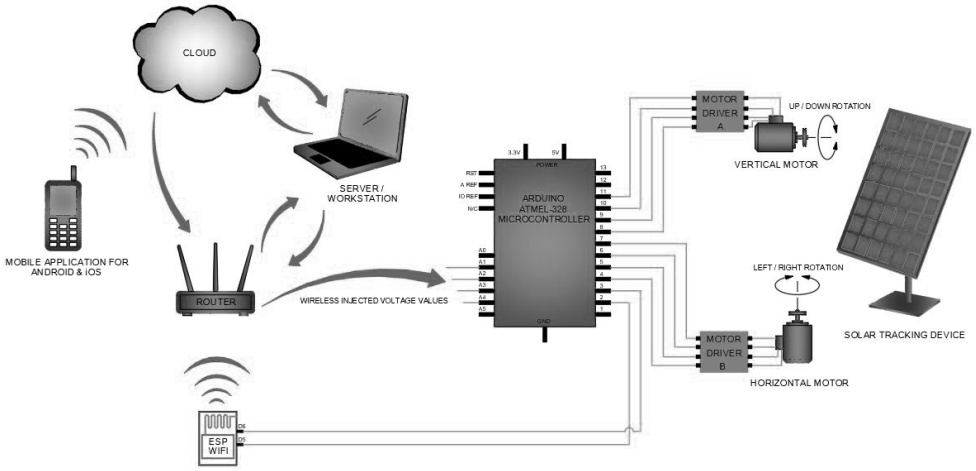


Fig. 4.10. Overview of our Implemented System and White-Box Testing Equipment for a Solar Tracker Device.

The Cloud Server implementation is based on a MQTT [172] broker, which together with the Node-RED [182] interface, is able to test the input and output values flow. We are applying White-box testing in order to analyze the program structure and attempt to find any bugs or programming errors. Some of the common errors encountered during software programming of our solar tracking device include functionality errors, communication faults, syntax errors, error handling defects, control flow errors, and calculation errors. In this direction, we minimized the use of buffers and arrays and instead opted to use individual objects to reduce memory fragmentation and avoid buffer overflow errors. We have also minimized the use of strings, which are notorious [183] for bad memory management on the Arduino platform. The Arduino bootloader checks the firmware while it is being uploaded using checksums for each data segment uploaded, thus eliminating or minimizing the possibility of flash errors. Flash errors are detected while uploading and as a result, the user is prompted to attempt the upload process again. We have also implemented a functionality that enables the Arduino microcontroller to report live analog values from the solar PV cells to the MQTT broker. This data gives the user real-time feedback on the results regarding the movement of the stepper motors and it can be gathered by the end-user for storage visualization or further analysis.

4.2.2. White-Box Testing System Overview

Regarding the hardware implementation on which White-Box testing is applied, the main control board for the project is the Arduino Uno, which controls the position of the solar panel using stepper motors. It reads analog voltage values from the solar panel and is able to detect which direction the Sun is moving in by calculating the corner of the solar panel that is producing the highest amount of electricity. This system works autonomously, but, in order to test all the possible conditions, we had to implement another system because we could not test the device effectively using only the Sun's movement which is very slow and does not vary enough over short periods of time. For this, we introduced a secondary microcontroller called NodeMCU ESP 12, which imitates the sensor values read from the solar panel. Using the NodeMCU ESP 12, we can vary the sensor values within a given range and observe the movement of the solar tracking device. This microcontroller is linked to a web interface and an app, from which, simulated sensor values can be easily sent, enabling us to achieve fast development and test cycles. NodeMCU ESP 12 is a well-known Wi-Fi enabled microcontroller. Its main function is to serve as a middleman between the Arduino Uno and the Internet because the Arduino Uno lacks inbuilt Wi-Fi capabilities. We have chosen NodeMCU ESP 12 because it is readily available and also compatible with our already existing Arduino microcontroller (ideally it can act as both the main and secondary microcontroller, eliminating the need for the Arduino Uno in future iterations of this work). Data and control in our setup flows from the user (Web Interface/App) to the Secondary microcontroller using the MQTT protocol [172]. The NodeMCU ESP 12 receives data from the web interface through the MQTT, processes this data and sends it to the Arduino for execution. It communicates with the Arduino over serial communication and a Universal Asynchronous Receiver Transmitter (UART), which is a convenient choice for short distances and high-speed communications.

MQTT is a machine-to-machine (M2M)/"IoT" lightweight transport protocol that is using the network bandwidth in an efficient way (with a 2 byte fixed header), assuring the delivery of the message from the nodes to the server. It was introduced by IBM in the year 1999 and recently standardized by the Organization for the Advancement of Structured Information Standards (OASIS) [184]. Because it is a message-oriented information exchange protocol based on publish/subscribe, the connections usually involve two types of agents: an MQTT client and an MQTT server, also known as a public broker. An MQTT client is considered to be any device (e.g. sensors, mobiles) that exchange application messages through the MQTT and can be either publisher (publishes application messages) or subscriber (requests for the application messages), both of them being isolated (they do not have to be aware of each other's existence or application). The public broker (Server), being a device or program that interconnects the MQTT clients, it accepts and transmits this application messages between them, being responsible for collecting and organizing the data. MQTT is designed with all complexities in mind to simplify a client's implementation.

For the web interface, we have implemented the Node-RED [182] IoT platform, which is specially tailored towards IoT and embedded systems applications. It is built on Node.js and has support for integrating numerous hardware devices, APIs and online services. We chose Node-RED because it has inbuilt support for our MQTT protocol, it is lightweight and easy to implement and

because it provides a convenient data flow editor. The Node-RED Flow is configured to send MQTT messages to our NodeMCU ESP 12 on a specified topic. The values to be sent are controlled by sliders, as can be seen in Fig.4.11. It has also been configured to receive a confirmation message from the NodeMCU ESP 12 once the command has been executed.

The mobile application was developed using Ionic Framework, which is a framework that enables hybrid app development, meaning one can build apps for Android, iOS and Windows devices using the same code platform. Our app is a wrapper around the Node-RED and acts as a browser, providing us access to the Node-RED Dashboard. We developed our mobile application in two versions: an iOS app as well as an Android app.

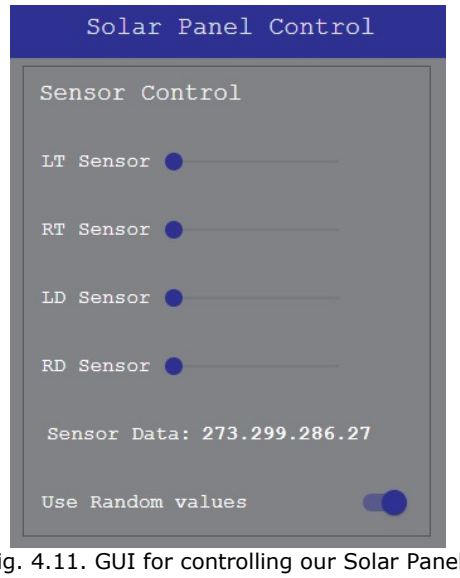


Fig. 4.11. GUI for controlling our Solar Panel.

A pivotal part of the project is the Cloud Server, which has a variety of functions. It functions as the MQTT broker, which means it handles the communication between different clients on the MQTT network and it also acts as the host for our Node-RED flow (we start the MQTT broker, followed by the import of the MQTT flows). The MQTT server was configured by installing Mosquitto, an open-source message broker for MQTT that uses a publish and subscribe model in order to be able to test the input and output values flow. Important to notice here is that Node-RED has both publish (mqtt out) and subscribe (mqtt in) nodes. Arduino IDE, a C++ type language, is used to program the nodes.

4.2.3. Wireless-Based Software Technique

In this section, we focus on the logical diagram description derived from the White-box software code and the testing environment, which represents the interaction between the Cloud Server Layer and Software Layer, illustrated in Fig.4.12. The White-box testing algorithm includes the main program functions as well as custom implemented testing functions.

Regarding the main program functions, our WBST approach is making use of the Node-RED interface, which can be seen as a group of nodes through which data flows. These nodes can modify, display or send data depending on their type.

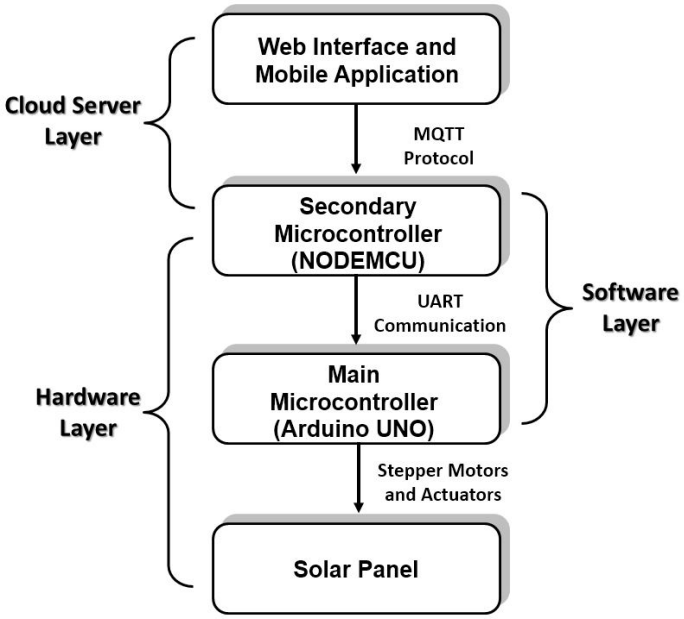


Fig. 4.12. The flow of Data and Control.

The first 4 nodes in Fig.4.13, **Left-Top (LT) Sensor, Right-Top (RT) Sensor, Left-Down (LD) Sensor,** and **Right-Down (RD) Sensor** are input nodes. They are sliders that allow the user to input the value to be sent to the microcontroller in order to test the cases in which the solar tracking device might engage in a failure.

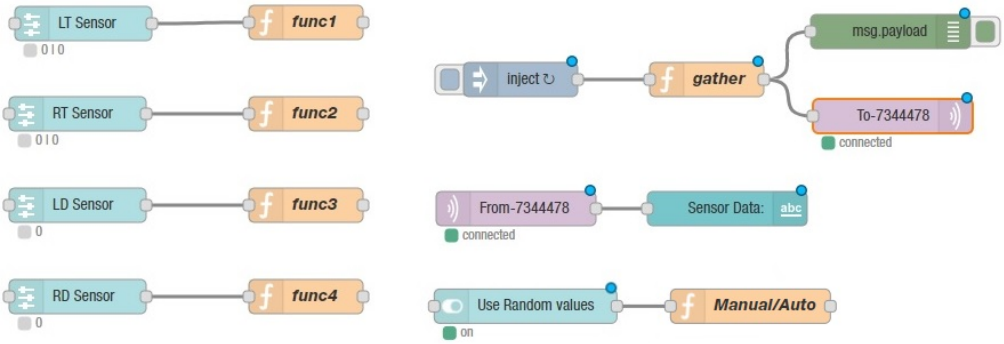


Fig. 4.13. Group of Nodes in the Node-RED Interface for our Solar Tracking Device.

The function nodes **func1, func2, func3,** and **func4** are functions that set input values to the global variables that are sent to the microcontroller. Normally, these variables are collected voltage values from the solar cells which are further

processed on the Arduino UNO board. The **gather** function collects all the inputs from the 4 global variables that were set by the 4 input sliders and combines them into a single string that will be published to the MQTT broker.

The topic to which it publishes is "**To-7344478**". This string is derived from the Unique identification number of the NodeMCU ESP 12, which is shown as "7344478". There exists another communication node called "**From-7344478**" which shows the connection status of the ESP8266 Wi-Fi module. The "**Sensor Data:**" shows values from the LT, RT, LD and RD positions. The "**Use random values**" node is relevant for realizing the experimental results because it provides the option to insert input data either manually or automatically.

The control flow of the White-box testing strategy is presented in Fig.4.14.

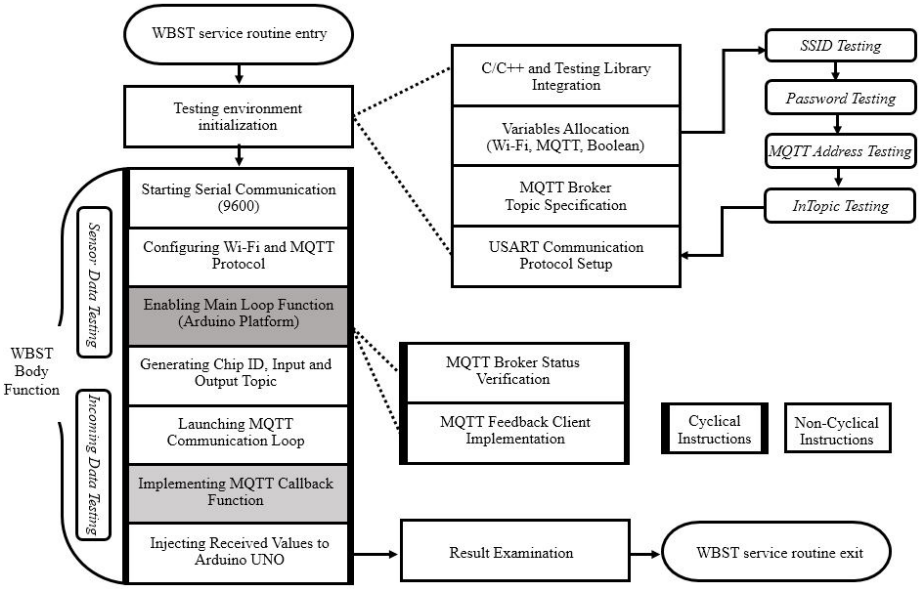


Fig. 4.14. White-Box Testing Strategy Execution Flow.

The test strategy starts with a test environment initialization phase. Here, we included the necessary libraries that provide useful functions and APIs that are needed for our application. The Variables Allocation block is responsible for specifying the password and name of our Wi-Fi connection. Another important aspect is the IP address allocation constant, which has a stable value. Prior to upgrading to a Cloud-based server solution, the field of the IP address required constant rewriting in the software code of the ESP8266 board and the Node-RED interface, each time we attempted to create experimental testing cases for the solar tracking device. Also, important to mention is that in this step, we specify the address of the MQTT broker. In our case, this will be our server, which acts as the MQTT broker. The next block underneath the Variables Allocation specifies the topic for MQTT communication. On an MQTT network, topics are used to decide who receives a message. When a client is sending a message, it specifies the topic on the broker to which that message will be published. When this message is published successfully on the broker, only clients that are subscribed to that message will be able to get the message. Our server has already been configured to listen to the messages published by NodeMCU ESP 12.

The last block of the Testing environment initialization specifies the serial communication details. On the Arduino platform, any 2 pins can be configured to be used for Universal Synchronous/Asynchronous Receiver Transmitter (USART) communication. In our application, we have specified D5 and D6 to be used as serial communication pins. As a general observation, on the Arduino platform, the setup function is used for all the one-time initialization that needs to be made at the start of our code like Pin configurations, Serial communication baud rate, Wi-Fi Initialization, etc. The next block in the logical diagram designates a cyclical instruction set, in other words, a programming structure that repeats itself periodically. The considered block launches our serial communication at a baud rate of 9600. The first serial communication (serial) is used to print debug messages to the Arduino serial terminal. The second serial communication channel is used for data exchange with the NodeMCU ESP 12. From here we are able to call a function that configures our Wi-Fi. It provides the Wi-Fi Service Set Identifier (SSID) and password to the chip so that it can connect to the specified Wi-Fi network. The second block from the Starting Serial Communication also calls a function that initializes the MQTT protocol by specifying the address of the broker, the topics to subscribe to as well as the callback function that should be called when a message is received from the MQTT broker.

The most important step in our software testing technique is enabling the main loop function of the Arduino platform. The void loop function on the Arduino Uno is the infinite loop which, keeps running as long as there is power. In the loop, first, we check if our MQTT connection still exists. One example is the situation when the connection to the broker is dropped because of network issues or the client may have been rejected by the broker. If our connection still exists, we continue with the current process, otherwise, we try to reconnect to the broker. This cyclical structure contains two more entities, described as follows: a) an MQTT Broker Status Verification which implements a small function in order to publish a message periodically to the broker, informing the broker of its state. This enables us to know if our commands sent from the Server were relayed to the Arduino successfully; b) an MQTT Feedback Client Implementation (client loop), which waits for the messages from the broker and passes received messages to our application.

We use *MQTT_Connect* to connect or reconnect to the MQTT broker. At the very beginning of the *MQTT_Connect* function, we specify the address of the MQTT broker and the port to be used for the connection. In this area, we also specify the MQTT callback function, which is the function that we want to call every time we receive a message from the broker. In our application, we want to forward every message we receive from the broker to the Arduino through serial communication. This callback function will, therefore, count the number of attempts to connect to the broker.

Lastly, the logical diagram ends with the MQTT Callback Function Implementation, which basically parses the data received from the broker into integers and then calls a function *UNO_Send()* to transmit this data to the Arduino over serial communication, together with publishing an acknowledgment message to the broker. The received values on the Arduino Uno are therefore injected in the device and processed in order to determine the respective outputs of the platform. Reaching this point, the service routine will exit the flow execution process.

Regarding the testing functions, White-box testing is very efficient in finding hidden errors and optimizing code base, but one disadvantage is that it does not help us find unimplemented or missing issues. White-box testing can be used in Unit testing [171], Integration testing and Regression testing. Some important types of

White-box testing include Control Flow Testing, Branch Testing, Data Flow Testing, Basis Path Testing and Loop Testing [185].

As mentioned earlier, we are applying White-box testing to analyze the program structure and attempt to find any bugs or programming errors. Some of the common errors encountered during software programming include Functionality errors, Communication errors, Syntax errors, Error Handling errors, Control Flow and Calculation errors. We will not be exploring Syntax errors because we are working with a compiled language (C++) and the programmer was informed of all the Syntax errors at compile time. We will pay more attention to Communication, Control Flow and Error Handling errors.

Communication errors are a type of errors that occur in communication between software and end-user as well as within the software. We will give particular attention to Communication errors within the software because we have 2 communication links: the first is between the Server and secondary microcontroller (MQTT over TCP/IP), and the second is between the primary microcontroller (Arduino Uno) and the secondary microcontroller (ESP8266). Error handling defects arise when there is no proper structure in place to handle unexpected values or actions in the program which could lead to hardware faults, circuit noise, etc.

Control Flow in a software decides what it will execute next or what action it will take under certain conditions. Errors in Control Flow can lead to a buffer overflow, unpredictable system states and a host of other problems. In our firmware, we have identified the possible error points in the system using our internal knowledge of the firmware structure and tested it using a unit testing library for the Arduino platform called AUnit [171]. AUnit is a unit testing framework that draws inspiration from the Google Test and Arduino Unit APIs. The following presentation represents a brief description of the implementation of the testing techniques on the ESP8266 firmware, which acts as a middleman between the Arduino Uno and the MQTT server: the first test verifies that the SSID, password and server addresses provided are valid. This test checks if the values are in the alphanumeric range. A wrong SSID, password, or MQTT server address by the user will lead to connection failure as the device will not be able to connect to the Wi-Fi or to the MQTT broker address.

Another possible error point which we have tested is the MQTT Topic array. This is the topic to which the MQTT client subscribes in order to receive messages from the MQTT broker. This topic is unique for each instance of the device, therefore, we have to generate them dynamically using the unique identification number of the microcontroller. We read this unique identification number and use it to generate a string which is then the topic of our incoming message; if there is an error in this topic, we will not be able to receive any incoming messages from the broker and our device will fail. Therefore knowing the expected length of the unique identification number, we run a test that will ensure that the topic of the incoming message is not longer than this length and that all the characters have alphanumeric value. The most error-prone part here, are the incoming data values. These are the values that are received from the MQTT broker and forwarded to the Arduino Uno. The values could be corrupted as a result of communication errors or unexpected user input. If we successfully forward a wrong value from the Arduino Uno, this can lead to unexpected or random behavior of the stepper motors. Therefore, we test to make sure that the values received are within a specified range. It is not uncommon for internet communication to drop unexpectedly, or for MQTT clients to disconnect from the broker. In a situation like this, the device needs to be able to detect that the connection has been lost and attempt to reconnect. In

this case, we implement a control flow test to check the MQTT connection and attempt to reconnect if the connection has been lost. Different routers allow connections at different speeds, some take a longer time to authenticate before internet access is allowed and therefore it is necessary to test that the device has been successfully connected to the router before we proceed with execution of the program. Failure to do this will lead to data loss as the device may attempt to send data when a connection has not been established.

Additionally, the main test on the firmware for the Arduino Uno code is to ensure that the data received via serial connection is valid. Serial communication is prone to different types of errors, especially at high speeds (e.g. 115200 baud rate). If we fail to verify the integrity of the received data, we may feed wrong values to the stepper motor controls, which can lead to unpredictable or unintended movements. Since these are unsigned analog values, our main test is to ensure that they are not above 1024. For a more safety-critical system, we can apply checksums to further validate the integrity of the firmware.

4.2.4. Experimental Setup and Results for White-Box Testing

In this section, we will present the experimental results regarding the basic movements of the solar tracker as well as the coverage percentage of a certain category of memory errors. Our testing equipment, which is depicted in Fig.4.10, can be divided into two major parts: a) the infrastructure composed of physical components such as the Arduino Uno, Dual H-Bridges circuits, stepper motors and the solar panel; b) the abstract layer, given by the software testing code and cloud server setup which were developed on the workstation platform and the specialized network development board.

With regard to the basic movements of the solar tracker, we continue with the description of the Cloud Server setup to finally shape the testing environment of the proposed method. By isolating the Arduino Uno from the rest of the components, we can construct a low-cost solution in order to collect the output signals from the board. Our method strategy relies on the mounting of LEDs at each digital output of the Arduino device. Pulse Width Modulation (PWM) signals usually translate themselves in HIGH and LOW digital values (1's and 0's). However, this testing artifice depends heavily on the observability factor of the test engineer, therefore it is more convenient resorting on a professional device such as an Oscilloscope to ease the signal gathering process. In this direction, we used a Hantek6022BE PC-Oscilloscope. The only necessary requirement before beginning with the actual experiments was the initial calibration of the two probes which came equipped with a clipper. The experimental results presented in Table 14 are showing the injected random unassigned integer values from the Cloud Server which are further processed in the software code of the Arduino board.

Table 14. Input values flow obtained from simulating environmental solar changes induced by artificial light.

Crt. Nr.	**Manually Injected Random Integer Values**				**Oscilloscope Connections**			
	A1	***A2***	***A3***	***A4***	***Out1***	***Out2***	***Out3***	***Out4***
1	1564	987	1703	983	Clipper	Probe	Clipper	Probe
2	756	1588	984	2157	Clipper	Probe	Clipper	Probe
3	1570	1574	984	986	Probe	Clipper	Probe	Clipper
4	980	988	1588	2170	Clipper	Probe	Clipper	Probe

This is resulting in sequential commands received by the motor drivers in order to control the horizontal and vertical motor, in one direction or another. While many of these output values proved to be similar for each motor driver, we decided to reduce the number of possible scenarios to 6 test cases.

The first test case involves injecting only one high value to a random corner of the solar panel. The following values are collected from the solar tracking device while it was tested during cloudy weather with normal light distribution. The injected random unassigned integer values are adjusted accordantly to fit these real-life scenarios. The inputs of the Optocoupler device require additional calibration in order to obtain almost similar voltages (inputs of the Arduino board can be seen in [186]). As seen in Table 14, the A1, A2, A3, and A4 represent the injected random unassigned integer values in the system to test the behavior of the solar tracker, thus the movement of the solar panel only begins at a value not lower than 100, meaning that the average difference between the corners should be approximately 50. Because the Oscilloscope only offered two channels for signal viewing, we had to split the testing into multiple subcases given by the manner in which we connected the probes to the outputs of the motor drivers. Both motor drivers in Fig.4.10 are identical L298N circuits like the ones from our previous work described earlier [17]. Regarding the stepper motors, there is one considerable difference involving the horizontal motor having a median point in its windings. Each pair of outputs Out1, Out2, respectively Out3 and Out4 are separated into two coils. This is why, in the Output Configuration, we connected each probe at one winding to receive the digital signals, and the two Clippers were linked to the same median point (positive value) of the coil.

There are multiple methods of driving a stepper motor. To elaborate, we can assume that each coil is activated once at a time. However, if we overlap coil activation, we can get a more finely tuned step. So if we have 1 activated, then 1 and 2 activated, then 2 activated, we can see that the internal gear will have a step to the left direction between coil 1 and coil 2. This is shown in Fig.4.15.

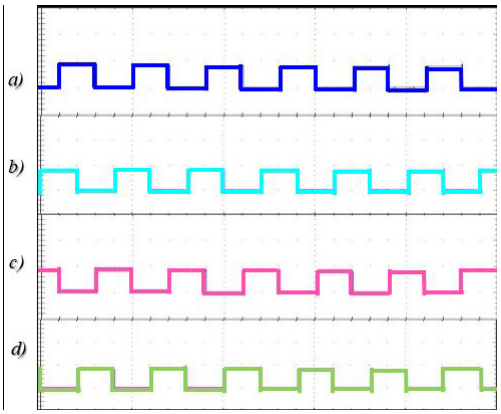


Fig. 4.15. Solar Panel Left and Right Rotation generated by pairing terminals from both windings.

Because the exact test scenario can be obtained by injecting a certain value for the bottom left corner, we will move on to the next test case. **The second test case** implies injecting only one value for the Right Top Sensor to demonstrate the

rotation to the left of the solar panel. **The third test case** combines two injected values and proves that no matter how many values we add to the first test scenario, the rotation will still remain in the right direction. **The fourth test case** implies inserting specific voltage values to the opposite side of the solar panel, in order to verify its rotation action. With the first four test scenarios completed, we finished the entire testing phase of the horizontal motor. **The next two test cases are dedicated to the vertical motor**, which is also worthy to mention that when compared to the horizontal stepper it doesn't contain a common wire in each coil. In Fig.4.16, we can observe a short phase delay between the two collected signals, thus, this can be understood, since the vertical motor uses fewer steps for lowering the solar panel down.

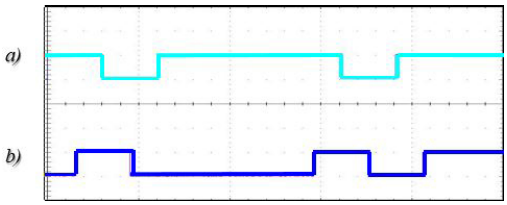


Fig. 4.16. Solar Panel Left and Right Rotation generated by pairing terminals from both windings.

The resulted measurements show an inverted state of the second signal compared to the previous case thus translating these impulses in a lifting operation of the solar panel.

Control Flow errors are related to In Topic and Out Topic values, as seen in Fig. 4.14. The Control Flow errors can be detected by using the assertion function of the AUnit library [171] which is applied to the input strings that enter the Arduino board. Usually, whenever there is a large data package exceeding 10 characters, the test program will capture the altered string along with the program structure. Calculation Errors may occur in real-time scenarios when negative voltages arise at the analog inputs of the solar tracking device. Our WBST solves this issue by implementing three assertion functions for testing if the input sensor string is a value that fits in the required range from 0 to 1024. At whatever time, a series of input data is outside of the specified range, the test program will detect the altered information on the serial monitor. To avoid the intrusion of voltages with inverted polarity, it is important to additionally check if the input string is an unsigned integer number. Error handling faults generally arise from large data packages exceeding a reference value of 10.000.

If three of these data packets are detected, they will be captured by the test program. To demonstrate the efficiency of the presented White-box testing strategy, we resorted to 4 batches of test cases, as can be seen in Table 15.

Table 15. Coverage and Speed of Execution for our WBST.

Test Cases		**Detected Entries**				**Coverage [%]**	**Runtime Execution [m]**
Name	**Total Number**	**Control Flow Errors**	**Communication Errors**	**Calculation Errors**	**Error Handling Faults**		
Batch 1	2277	395	2	568	736	74.70	240
Batch 2	1087	175	1	242	299	65.96	131

Batch 3	840	118	1	166	214	59.40	110
Batch 4	130	78	0	2	42	93.84	17

For these test cases, in order to trigger possible errors/faults, large amounts of input data were injected in an automated manner. For the first batch consisting of 2.277 test cases, we achieved a fault coverage of 17.35% for Control Flow errors, 24.95% coverage for Calculation errors and 32.32% for Error handling faults; this resulted in a total coverage percentage of 74.62% for all the considered error types. Important to notice here is the improved coverage we were able to achieve (i.e. 93.84%) for a much smaller number of test cases (i.e. 130) in batch 4, which resulted in a reduced total runtime execution for the considered tests. The average speed of execution per test cycle was estimated between 4 and 5 seconds. It is important to mention here that Batch 2 contained 174 error-free data entries while Batch 3 included 137 data packages without faults. As expected, Communication errors were rarely encountered due to the robust implementation of the proposed WBST.

4.2.5. Online Built-In Self-Test Architecture for Automated Testing of a Solar Tracking Equipment

Typically, a dual-axis solar tracker is composed of many electronic and mechanical components such as simple IC, e.g. L298N, and complex ICs, such as Microcontroller Units (MCUs), e.g. Arduino UNO, as well as DC-motors and Stepper motors. These components, especially the electronic ones, are prone to errors during their operation due to the fact that usually, they are functioning under unfavorable weather conditions which may cause them to malfunction. By comparison, this is not the case for software related errors found in a solar tracker. The hardware malfunction can be caused by several types of errors such as single bit-flip errors, stuck-at faults, delay faults, and bridging faults, to name only a few.

Considering these aspects, we aim to minimize the operation costs of our dual-axis solar tracker [17], which in case of malfunction (e.g. L298N IC overheating issues, stepper motors don't receive proper stimuli) can result in high financial losses by proposing an OBIST architecture. More specifically, we target the automation components of our solar tracking system that is composed of an Optocoupler, an Arduino UNO and two L298N ICs and which rely on an Idle State Detector and two switch batches in order to enable or disable the online testing procedure.

Regarding the proposed fault injection strategy and according to the traditional fault injection methodology [187], we concluded that the most suitable technique for our solar tracking device is the so-called physical-based fault injection or Hardware Implemented Fault Injection (HWIFI).

The fault injection strategy is presented in Fig.4.17 and satisfies two objectives in our test scenarios: first, we apply a variety of voltage values to the inputs of the Optocoupler and the Arduino UNO and monitor the outputs of each device with a multimeter; secondly, we move on to the parasitic signal injection to check the output signals of the L298N circuits on the PC Oscilloscope.

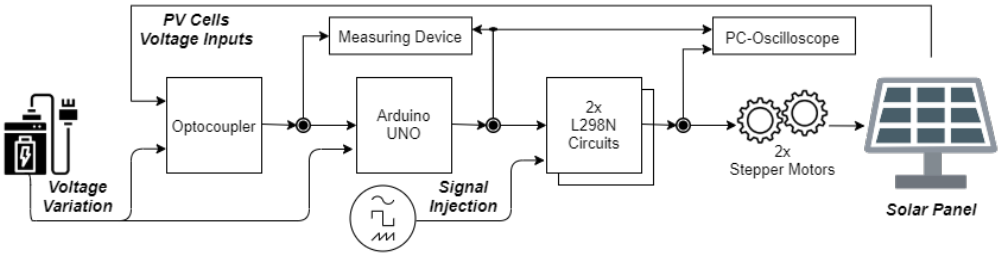


Fig. 4.17. Fault Injection Strategy General Model applied to our Solar Tracker.

According to our previous study in [18] where we applied a White-box testing procedure, we concluded that voltage variation does not affect the system's stability but rather establishes the verge at which the solar tracker starts to function normally, usually at 2.1V. Eventually, the parasitic signal injection may be overlapped with voltage variation to compensate stability testing of the proposed solar tracking system. HWIFI is achieved best via contact (active probes) in order to test the dependability of the proposed Sun tracking system and is effective at any level of the hardware design, which includes the circuit chain: *Optocoupler → Arduino UNO → L298N*. The HWIFI is a top-bottom approach, being extremely useful to mark off the fault coverage domain, from regular parasitic signal insertion to more complex scenarios such as the test vector injection. HWIFI ensures a very easy encoding procedure that will establish the main pattern of a correct signal, as shown in case A) from Fig.4.18.

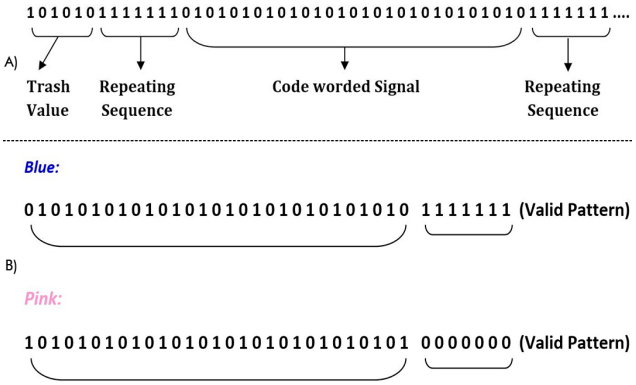


Fig. 4.18. Case A: Arduino Output Signals 4 and 5. Case B: Code worded Signal (Blue) and Cycle Time (Pink).

If we discard the initial trash value (last 6 bits from a previous Code worded Signal) and the repeating sequence, we realize that the valid pattern is structured from the Code worded Signal and the last Repeating Sequence composed of binary 1's. Together they form a so-called cyclic sequence that is generated continuously in order to inject stimulus to the L298N circuit coils. However, one motor driver circuit receives four inputs and generates the same number of outputs. The general rule is that each 2-signal pairs are equivalent in shape, so it is sufficient to analyze only two signal channels on the Oscilloscope. While at first glance, the two generated patterns might look the same, there is one major difference between them, as seen in Fig.4.19 wherein the dual-channel, we can observe that the first output signal (blue color) has the same pattern as the one in Fig.4.20.

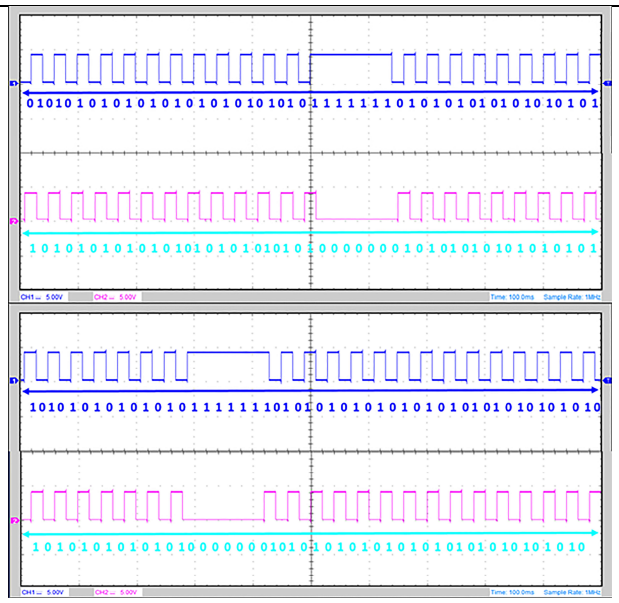


Fig. 4.19. Top: L298N Output Signals 1 and 4. Bottom: L298N Output Signals 2 and 3.

If we look closer, we can see that the second signal (pink color) behaves in a slightly different way, meaning that it generates the inverted pattern of the first output. The rewritten data bits by using only the Code worded fragment and the Repeating Sequence in both cases is illustrated in case B) from Fig.4.18.

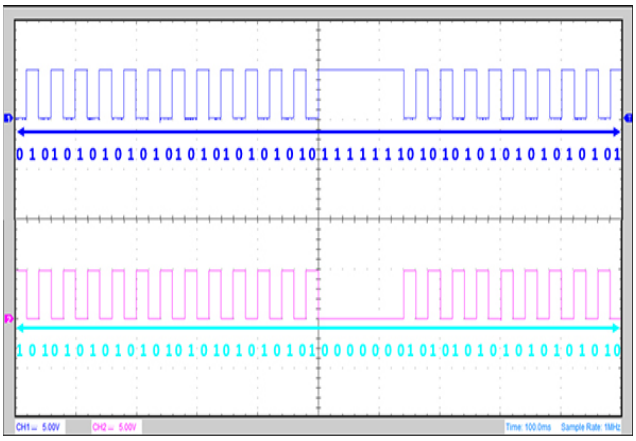


Fig. 4.20. Arduino UNO Output Signals.

Regarding the outputs of the DUT, each pair of signals represent the outputs of a singular L298N circuit from the two available in our solar tracking device. For simplicity, we considered a scenario where we energize the horizontal motor that executes a left rotation according to the output seen in Fig.4.19. In order to inject parasitic signals over the correct waveforms, we used a Precision Waveform Generator, also known as Voltage Controlled Oscillator (ICL8038), which is able to generate three distinct output waveforms, namely sinusoidal, triangle and square

waveform signals. In order to obtain an improved performance model, which is even more customizable, we started from the basic model design and modified its structure. The basic model consists of the variable resistance VR_1-potentiometer, the R_1-resistance, and the C_1-capacitance which are involved in the following formula (4.4) that determines the frequency output (f) of the circuit:

$$f = \frac{0.15}{(VR_1 + R_1)\mathrm{C}_1} \tag{4.4}$$

The first prototype being cheaper also proved less efficient during testing, this being the reason for which we switched to an improved version where we used more components in order to improve the signal-adjusting feature. By replacing the C_1 component with a variable capacitator, we are able to change the C_1 slot by any capacitator value that withstands a voltage of 25V, thus providing a variety of frequency generation (1 Hz – 100 Hz for 1 μF; 100 Hz – 1 kHz for 0.1 μF; 1 kHz – 10 kHz for 0.01 μF; 10 kHz – 100 kHz for 0.001 μF). With a proper selector switch, these capacitors do not require constant replacement. By choosing different frequency ranges, we were able to detect faulty output signals of our two CUTs as seen in Fig.4.21.

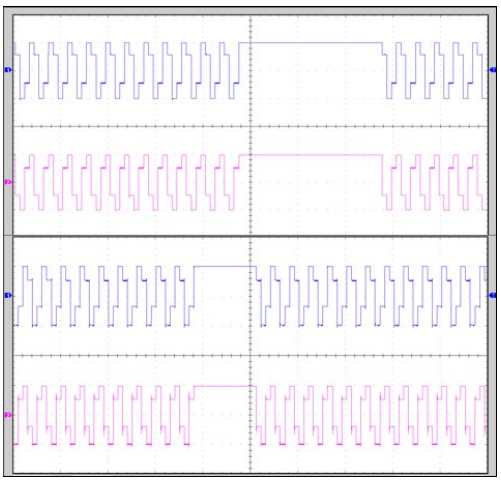


Fig. 4.21. Faulty Output Signals resulted from the Injection Process (Top: Arduino UNO and Bottom: L298N Dual H Bridge).

4.2.6. Hardware BIST Components

As can be seen in Fig. 4.22, the proposed BIST architecture linked to our circuit chain presented earlier in Fig.4.17, is divided into four layers involving: a) the construction of a TPG unit which is an LFSR; b) 4 CUTs defined by the Optocoupler, Arduino UNO, and two L298N ICs; c) a Results Gatherer which is a MISR; d) with the help of a set of ADCs and DACs as well as a dedicated idle state detector, we implemented the BIST architecture that provides an online testing service.

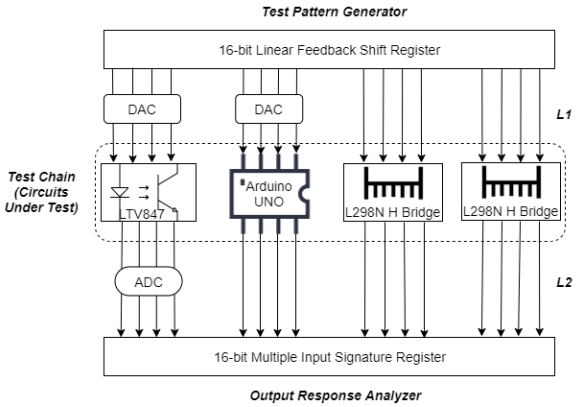


Fig. 4.22. Proposed BIST Architecture.

Regarding the TPG (LFSR) implementation, because we target single bit-flips and single stuck-at-faults, we will randomly inject a set of test vectors that will be constructed from all generated cyclical sequences of an LFSR. The primitive polynomial function $1+x+x^3+x^{12}+x^{16}$ [188] will ensure that the LFSR will be statically deployed as seen in Fig.4.23.

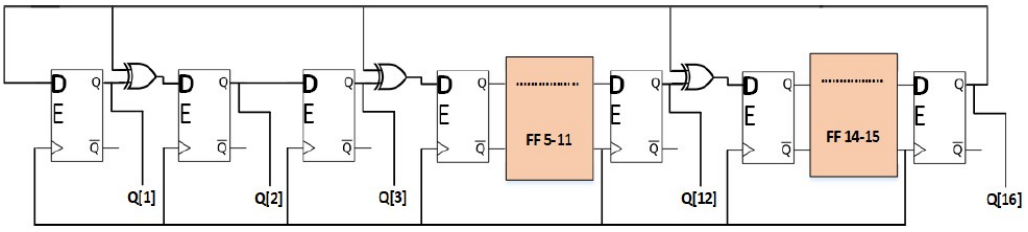


Fig. 4.23. Proposed LFSR Configuration.

We chose an LFSR with an internal EXOR gate because usually external gates introduce an additional delay to the circuit, which in the case of a fast generation of pseudo-random values is not desirable. A 16-rank LFSR will generate all patterns equivalent to the numbers between 1 and 65.535 in a pseudo-random order. We exclude an initial seed value of 0 0 0…0 because this particular case will always provide a cyclical value of zero.

As a general rule Q[1] is X^1, Q[2] is X^2, Q[3] is X^3, Q[4] is X^4 and so forth. The shifting of values is given by the following Boolean rules: ***Q[16] ← Q[15], Q[15] ← Q[14], Q[14] ← Q[13] XOR Q[16], … , Q[4] ← Q[3] XOR Q[16], … , Q[2] ← Q[1] XOR Q[16] and Q[1] ← Q[16].***

The MISR presented at the bottom part of Fig.4.22 is the most important element for our testing objectives. The MISR can be considered the Results Gatherer or the ORA for the DUT. This modified version of an LFSR will collect all stimulus responses containing faulty signal injections from the CUTs and will store them in the 16-bit register for future analysis.

ADCs are mainly used to transform an analog signal (e.g. sinusoidal waveform) in a fixed-point binary number while DACs usually execute the reverse function. Since the I/O pins of the Optocoupler and the inputs of the Arduino UNO require analog signals, we used one LM741 amplifier together with multiple resistors in order to process the information in purely binary or analog values.

As can be seen in Fig. 4.24, the four digital pins are connected at the input of the resistors and by changing the input values, the operational amplifier, which is powered at +12V and -12V, will convert the analog signals to binary digits. The Optocoupler's opposite side will connect the ADC to the MISR's four digital pins. Our ADC encapsulates four comparator circuits that are using LM741 operational amplifiers.

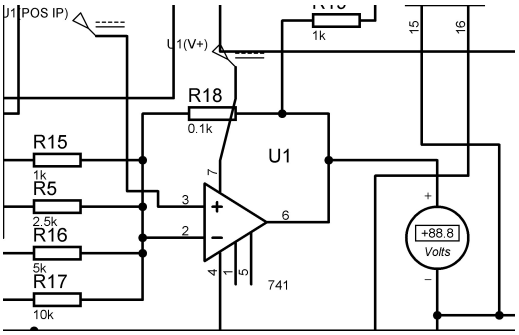


Fig. 4.24. Example of an LM741 Operational Amplifier.

One of the circuits that requires signal translation is the Optocoupler since it receives analog values from the solar panel lines L1 and output lines L2. Consequently, the Arduino UNO will require a DAC on input lines L1, as depicted earlier in Fig.4.22.

The idle state detector is presented in Fig.4.25 and is connected to the switches control panel, as depicted in Fig.4.26. Its hardware implementation contains 2 EXOR gates, one OR gate, and one Inverter. Whenever the sensor readings are all 0's or 1's, the idle state detector will automatically identify that the system is idle, meaning that switches batch A will be deactivated and switches batch B will be turned on. In any other case, when the values are completely different, the system will be active, which leads to the deactivation of the switches batch B and the returning to the normal operation data path (given by switches batch A).

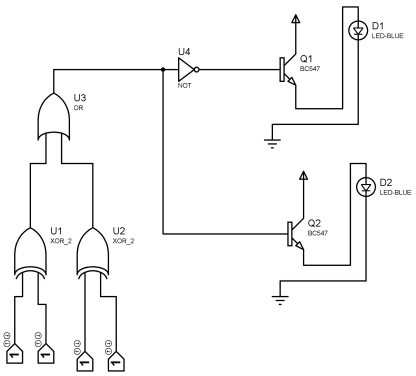


Fig. 4.25. Idle State Detector Hardware Implementation.

As seen in Fig. 4.25, the BC547 transistor is connected to the LED in series and the circuit output is connected to the transistor's base. The transistor's collector

and emitter are involved in switching the Batches A and B on or off in the control panel.

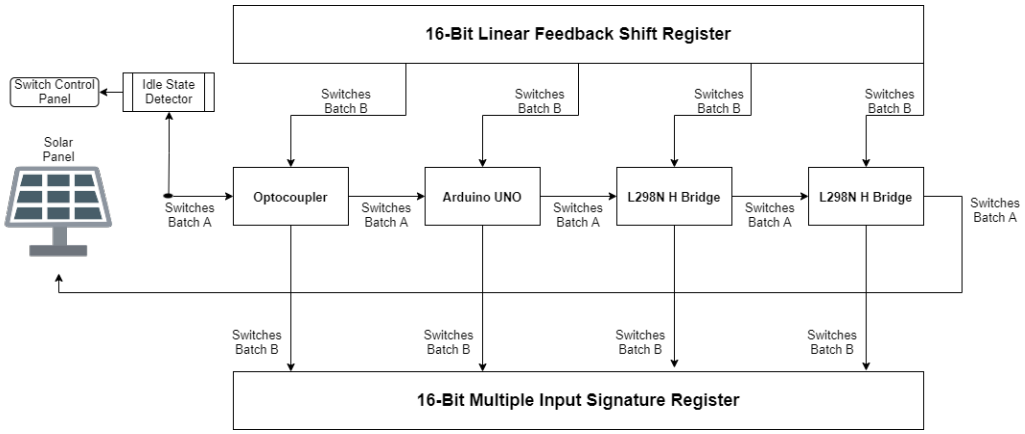


Fig. 4.26. Configurable OBIST Architecture Block Diagram.

The final integration of the above-mentioned hardware components is presented in the next paragraphs.

4.2.7. Proposed OBIST Architecture

The proposed OBIST model represents the joining of the LFSR, CUTs and MISR components into one compact architecture, as can be seen in Fig.4.26. This architecture is divided into two stages: first, we construct a hardware Very Large Scale Integration (VLSI) design in the Proteus environment; secondly, we develop a C++ application that generates signatures based on input test vectors in an exhaustive manner in order to validate the results obtained in the first stage of the implementation.

The first part of our implementation details the hardware setup that was utilized to validate the results obtained with the developed software code. We used Proteus 8.6 Professional Edition as our testing environment because it allows easy deployment of physical components (e.g. flip-flops, EXOR gates, transistors, etc.) and facilitates the systematic monitoring of the experimental results. The initial logical modules were constructed in the software implementation (described in the next section) and together with their inherited properties, they are converted into a hardware VLSI design model.

The proposed software code implementation adopts a layered approach where we declare for each virtual component a variable that defines the shifting process of the test vectors. For instance, LFSR is an object that will inherit an input, the 3 CUTs will depend on the input as well as the Enable line values and the MISR will contain the input and the expected signature output. Each Enable line has the role to authorize the shifting of the input signal directly to the output depending on its present value. In the main body of the program we will declare a LFSR register with an initial seed value of Q[16] = [1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1] while the MISR register will have an initial value of M[16] = [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0] so that it can store new signatures for each test cycle. The first

layer of the application is given by the Logical Module of the LFSR that generates the next pattern according to the current value of the register Q[1].

Regarding the DUT layer, a logical module implementation is attached to the original CUTs, which contain the circuit chain Optocoupler, Arduino UNO and two L298N ICs. The microcontroller and motor drivers depend exclusively on the 4 pin input variables In0, In1, In2, and In3 together with the Enable pins High or Low and will return the value of the output as an array of Out0, Out1, Out2, and Out3. For each test cycle, the CUTs will generate all test vectors that will enter the final layer of the implementation.

The operation mode of the LTV-847 Optocoupler was summarized in Table 16 where the DAC and ADC perform binary-to-analog conversions as well as the reverse function.

Table 16. LTV847 Optocoupler combined with DAC and ADC Components.

Optocoupler combined with DAC					**Optocoupler combined with ADC**				
Binary Input Configuration				***Output Values***	***Input Values***	***Binary Output Configuration***			
In[0]	***In[1]***	***In[2]***	***In[3]***	***Analog [V]***	***Analog [V]***	***Out[0]***	***Out[1]***	***Out[2]***	***Out[3]***
0	0	0	0	0.1	0.1	*d*	*d*	*d*	*d*
0	0	0	1	0.3	0.3	*d*	*d*	*d*	*d*
0	0	1	0	0.6	0.6	*d*	*d*	*d*	*d*
0	0	1	1	0.9	0.9	0	1	0	0
0	1	0	0	1.2	1.2	0	1	0	1
0	1	0	1	1.5	1.5	0	1	1	0
0	1	1	0	1.8	1.8	0	1	1	1
0	1	1	1	2.1	2.1	1	0	0	0
1	0	0	0	2.4	2.4	1	0	0	1
1	0	0	1	2.7	2.7	1	0	1	0
1	0	1	0	3.0	3.0	1	0	1	1
1	0	1	1	3.3	3.3	1	1	0	0
1	1	0	0	3.6	3.6	1	1	0	1
1	1	0	1	3.9	3.9	1	1	1	0
1	1	1	0	4.2	4.2	1	1	1	1
1	1	1	1	4.5	4.5	*d*	*d*	*d*	*d*

It is important to mention that our solar tracker starts functioning only when it receives a voltage value higher than 0.6V at the input of the Arduino UNO, this being the reason why the output of the Optocoupler will be only taken into consideration for the analog values greater than 0.6V and lower than 4.5V, the reason for which we denoted the lower and higher voltage values with "*d*" in Table 16. The MISR logical module takes the current value, calculates a new vector based on the CUTs outputs and stores it back in the MISR register, as can be seen in Table 17.

Table 17. MISR Output Signal Generation.

Crt. Nr.	**MISR Inputs**	**MISR Output Sequence**
1	In[15]	In[15] = In[14] ^ Out[15]
2	In[14]	In[14] = In[13] ^ Out[14]
3	In[13]	In[13] = In[14] ^ Out[13]
4	In[12]	In[12] = In[14] ^ Out[11]
5	In[11]	In[11] = In[14] ^ Out[11]

6	In[10]	In[10] = In[14] ^ Out[10]
7	In[9]	In[9] = In[14] ^ Out[9]
8	In[8]	In[8] = In[14] ^ Out[8]
9	In[7]	In[7] = In[14] ^ Out[7]
10	In[6]	In[6] = In[14] ^ Out[6]
11	In[5]	In[5] = In[14] ^ Out[5]
12	In[4]	In[4] = In[14] ^ Out[4]
13	In[3]	In[3] = In[14] ^ Out[3]
14	In[2]	In[2] = In[14] ^ Out[2]
15	In[1]	In[1] = In[14] ^ Out[1]
16	In[0]	In[0] = (In[0] ^ tmp) ^ Out[0]

The inputs of the MISR register are associated each with the outputs of the DUT circuit: Out1, Out2, Out3, ..., Out16. Similarly to the LFSR logical module, we will store the Q[15] in a temporal variable and will cycle through all the 65.535 iterations until exhausting all test vectors. The last two components, namely the Arduino UNO and L298N motor drivers were implemented according to Table 18.

Table 18. Arduino UNO and L298N equations translated in C++ language.

L298N Integrated Circuits				**Arduino UNO**	
Motor Driver 1		***Motor Driver 2***		***Microcontroller Unit***	
Inputs	***Outputs***	***Inputs***	***Outputs***	***Inputs***	***Outputs***
In[4]	Out[4] = In[4] && EnA	In[8]	Out[8] = In[8] && EnA	In[12]	Out[12] = in[12] ^ in[13]
In[5]	Out[5] = In[5] && EnA	In[9]	Out[9] = In[9] && EnA	In[13]	Out[13] = in[13] ^ in[14]
In[6]	Out[6] = In[6] && EnB	In[10]	Out[10] = In[10] && EnB	In[14]	Out[14] = in[14] ^ in[15]
In[7]	Out[7] = In[7] && EnB	In[11]	Out[11] = In[11] && EnB	In[15]	Out[15] = in[15] ^ in[12]

4.2.8. Experimental Setup and Results for OBIST

In order to validate the robustness of our proposed OBIST implementation, we cloned the initial CUTs chain in our software simulation to obtain two devices, one that generates correct patterns and another one that provides faulty responses. The purpose of the proposed architecture is to compare the pseudo-random MISR output signatures with the valid generated MISR signatures inside a dedicated block called Signal/Signature Analyzer and which is depicted in Fig.4.27.
During the execution of the software implementation, we have evaluated the fault coverage for two types of faults: random singular bit-flips as well as single stuck-at-faults. The test chain of the CUTs contains a total of 16 bits, but in our experiments, we are interested in targeting only 12 bits. These 12 bits are associated with the following 3 CUTs: Optocoupler, Arduino UNO and one L298N circuit. In the case of 12 bits, the possible number of faults is 4.096. Thus, the entire chain of 16 bits can be divided into two parts: one is the least significant part and the other is the most significant part. We are able to evaluate the fault coverage by analyzing these two parts.

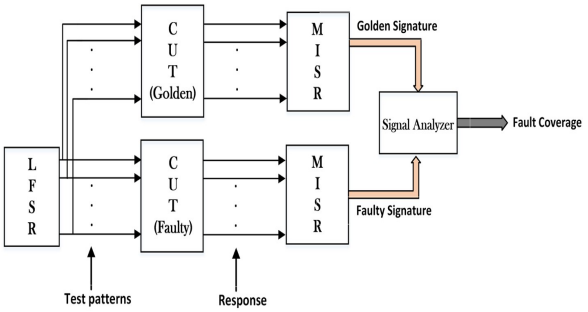


Fig. 4.27. Test Mode Architecture for the proposed OBIST strategy.

Single bit-flip errors can be easily detected by applying the single parity checking method. The single parity checking method is applied in our software implementation by firstly performing an EXOR operation between all the 16 bits of the test chain as well as by adding the extra parity bit in the least or most significant position of the bits chain, as seen earlier in Fig.4.18. Secondly, in order to determine if a single bit-flip has occurred during data transmission, we check the Code worded Signal again by calculating the parity bit in the same manner as mentioned earlier and comparing it to the initial extra parity bit.

The fault coverage for the single bit-flips, stuck-at-faults and the global coverage are presented in Table 19 where FC_{BF} represents the fault coverage for single bit-flip errors detected, FC_{SaF} represents the fault coverage for stuck-at-faults detected and FC_G represents the global fault coverage for both the single bit-flips and single stuck-at-faults. Regarding single bit-flip errors, in our software simulation, we applied the single parity checking method and succeeded to detect 93.93% of the targeted errors from a total number of 65.535 injected test patterns, each with a different initial seed value during 5 test cases, as seen in Table 19. It is known that stuck-at-faults can be easily detected, as they mostly occur due to damaged logical gates, transistors or permanent circuit damage. The five test cases regarding stuck-at-faults seen in Table 19 are performed by injecting 8, 12 and 16 bits in the test chain and based on the analysis of the Signal/Signature Analyzer, we determined that the fault coverage is 100%.

Table 19. Fault Analysis of single bit-flip errors as well as single bit stuck-at-faults.

Crt. Nr.	**Initial Seed (HEX)**	**FC_{BF}**	**FC_{SaF}**	**FC_G**
		Last 8 bits (Mutant)		***Random 12 bits***
1	FFFF	93.95%	100%	96.96%
2	8FFF	93.93%		
3	8CFF	93.92%		
4	8C9F	93.91%		
5	8C94	93.93%		

In order to obtain the global fault coverage of our test chain, we extended the initial test cases from 8 bits (the least significant and most significant part of the test chain) to a total number of 12 bits representing the entire DUT. Thus, the global fault coverage was evaluated for the total number of 12 bits and was determined at 96.96%, proving that our OBIST solution is capable of detecting all targeted errors regardless of the initial seed value.

Because the aliasing usually happens when the flawed device's signature is exactly the same as the perfect device's signature, the probability of aliasing occurrence is calculated with the relation $2^{-16} = 0.0001$ and results in the conclusion that in our experiments aliasing appears in very rare cases.

4.3. Efficient Implementation of a Self-Sufficient Solar-Powered Real-Time Deep Learning-Based System

As mentioned earlier, recent advancements in the field of AI, especially DL, are happening especially thanks to the availability of huge amounts of data and powerful hardware. During training and inference of a DNN, usually expensive and power-hungry GPUs are used, resulting in a proportional growth of computational and environmental costs, with some NLP models even increasing the carbon footprint nearly five times the lifetime emissions of a car [9].

Because climate change is a very relevant problem in our society [10] and considering goal number 7 (affordable and clean energy) and goal number 13 (climate action) of UN's Sustainable Development Goals [189], efforts to develop and use low-power embedded devices are made by many companies, an example, in this case, being Nvidia's Jetson TX2 embedded platform [190]. Consequently, in order to reduce the carbon footprint and the electricity bills, efforts towards renewable energy are made [48], with many researchers building solar tracking systems [17, 191] in order to capture the sun's energy with maximum efficiency.

Considering that the two domains of AI and renewable energy are of major importance for the development of our society, our work introduces a self-sufficient solar-powered real-time DL-based system that makes use of solar energy from the sun with the help of an updated version of our solar tracker based on the Cast-Shadow principle [17] and an Nvidia Jetson TX2 board that runs our real-time animal class identification model [15] on videos or using a webcam and also generates additional datasets containing images and textual information about the animals present in front of the frame, in real-time. In order to justify our decision for choosing an embedded platform instead of a laptop, in our experimental results, we present a comparison between the two platforms, mainly in terms of power consumption. Additionally, we also improve the energy efficiency of the proposed real-time DL-based system by implementing a motion detection method based on background subtraction with the help of Python and OpenCV [192].

A summarized view of the proposed solar-powered DL-based system can be seen in Fig.4.28. It consists of our dual-axis solar tracker based on the Cast-Shadow principle [17], a solar charge controller, an Ultra Cell accumulator with 12V 9 ampere-hour (Ah) acid plumb battery, two DC-to-DC inverters (first DC-to-DC inverter converts 12V to around 5V necessary for the solar tracker to become autonomous regarding energy needs and the second DC-to-DC inverter converts 12V to around 19V necessary for the Nvidia Jetson TX2 to run only on solar energy) and an Nvidia Jetson TX2 embedded platform that uses and powers an external Logitech C920 HD Pro webcam in order to identify animal classes in real-time [15].

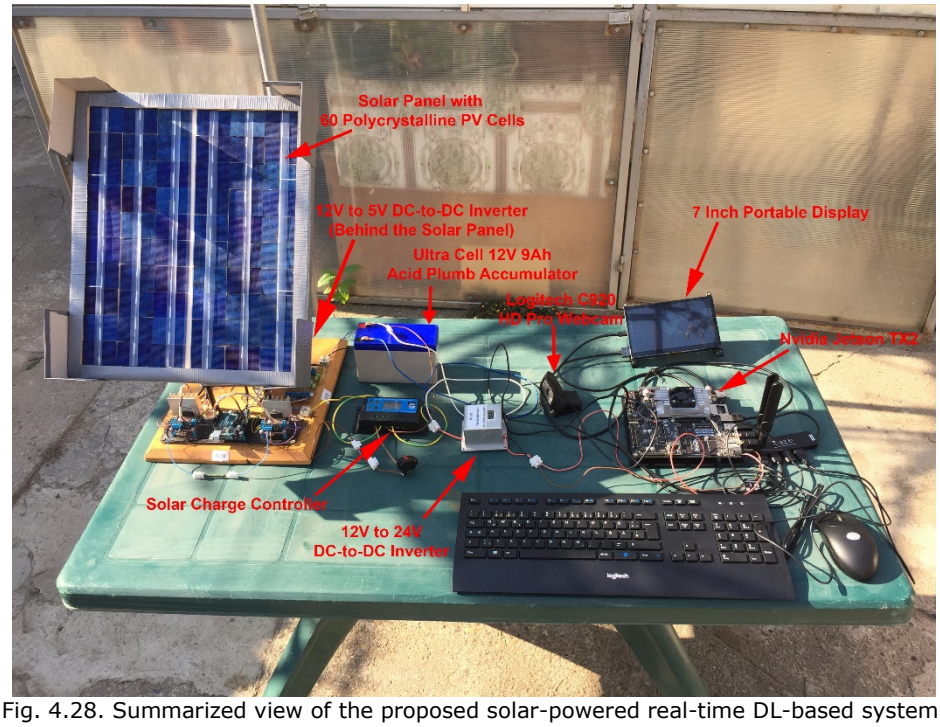


Fig. 4.28. Summarized view of the proposed solar-powered real-time DL-based system.

4.3.1. Solar Panel Improvements

As seen earlier, our old dual-axis solar tracker based on the Cast-Shadow principle [17] used a solar panel with 40 PV monocrystalline cells instead of LDRs which are usually found in the literature. Twelve of these PV cells are used to control 4 low-cost circuits, namely 1× Optocoupler LTV 847, 1× Arduino328 Microcontroller, 2× L298N Dual-H Bridge circuits and 2× stepper motors which are used for the dual-axis positioning of the solar tracker. Our solar panel makes use of 3 PV cells from each corner to analyze light distribution, 2 bypass diodes to protect PV cells in case of a sudden increase or decrease in voltage that may occur due to variable light and 2 blocking diodes to protect solar panel's PV cells from reverse current (i.e. voltage from the load such as the Optocoupler or Arduino UNO).

In order for the real-time DL-based system to run inference completely on solar energy, we considered using an updated version of our earlier proposed dual-axis solar tracker that uses the Cast-Shadow principle [17] in order to optimize its position for a more efficient solar energy gain, without the need of sensors. The important changes that we made to update our solar tracker in order to use it for the experimental results in this paper are:

1) The effective surface area of the panel used in the above described solar tracker was increased from $L_1 \times l_1 = 36 \times 35 = 1260$ cm^2 to an area of $L_2 \times l_2 = 43 \times 36 = 1548$ cm^2 to accommodate 60 polycrystalline PV cells.
2) The method used to produce silicon polycrystalline solar cells is easier to implement and less expensive as compared to monocrystalline counterparts, resulting in a more cost-effective investment. Additionally, polycrystalline

solar panels tend to be somewhat less tolerant of heat than monocrystalline solar panels [193]. Due to their higher temperature coefficient, the overall heat output will be less significant compared to monocrystalline solar modules. As a consequence, our old monocrystalline PV solar cells were replaced by PV polycrystalline cells that generate a maximum voltage of 0.55V and a maximum current of 0.60A per unit, resulting in a total voltage of 17V and 1.5A generated by the improved solar panel.

4.3.2. Deep Learning Models used for Inference

In order to prove the efficiency of our solar-powered real-time DL-based system, we decided to use our earlier proposed implementation regarding real-time identification of animals found in domestic areas of Europe [15] which can also generate 2 new datasets in real-time, one dataset containing textual information (i.e. animal class name, date and time interval when the animal was present in front of the webcam) and one dataset containing images of the animal classes present and identified in videos or in front of a webcam. These newly generated datasets are very useful, as they can provide insights about what animal classes are present at a given date and time in a certain area and how they look like.

As mentioned earlier, our original DL models presented in [15] were trained and tested on a home-made dataset with a total size of 4.06 GB consisting of 90.249 animal images (72.469 images for training, 8.994 images for validation and 8.786 images for testing) belonging to 34 classes on 4 state-of-the-art modified CNN architectures (VGG-19, InceptionV3, ResNet-50, and MobileNetV2) using Keras with Tensorflow backend, achieving high overall test accuracy (90.56% for the proposed VGG-19 model, 93.41% for the proposed InceptionV3 model, 93.49% for the proposed ResNet-50 model and 94.54% for the proposed MobileNetV2 model).

In order to successfully implement and test our Python implementation on the Nvidia Jetson TX2 board, we needed to make some adjustments and improvements in our initial code from [15] as follows:

1) First, because Keras saves its weights in a hierarchical data format (.hdf5) file which slows the loading of the model on the Nvidia Jetson TX2, we have created an optimized (converted it to a frozen graph base model; here the value of all variables are embedded in the graph itself thus the protocol buffers (.pb) file cannot be retrained) frozen file of our Keras model based on Tensorflow. Keras does not include by itself any means to export a TensorFlow graph as a .pb file, but we could do it using regular Tensorflow utilities.
2) Second, because by default Tensorflow pre-allocates the whole GPU memory [194] (which can cause "out of memory" errors) and because the Nvidia Jetson TX2 GPU doesn't have dedicated RAM and cannot use its full 8 GB processing RAM (the reason for this is because Linux and other processes are using most of the available RAM), we implemented a code to control the GPU memory allocation and to define (choose a GPU memory ratio allocation from 0 to 1) the processing memory percentage usage of the GPU at running time. By doing this, we can now control how much data we want to transfer to the GPU that processes it and avoid any possible "out of memory" kind of problems which otherwise would appear on the Nvidia Jetson TX2 due to its lack of dedicated GPU memory. We can now choose a ratio from 0 to 1 to

decide the GPU usage from low to high by passing the arguments in the command line.

3) Third, we implemented a code for testing a certain number of batch frames in one shot. For this, we make use of a Numpy array. Numpy array is a fast method for data manipulation that saves a matrix of numbers in stacks (e.g. we can observe that when we read a frame in OpenCV it becomes numbers in the Numpy array, meaning that if we have 900 frames, Numpy array size will be 900, 224, 224). Then, we transferred that batch of frames to our model for prediction by changing the number of frames (e.g. we can send 30 frames one time like this: 30, 224, 224, so we have a remaining 870, 224, 224 arrays). We have tested it on the frozen graph. The frames are passed from 1 to 60. Fps batch testing is very useful in our tests because it helps find the optimal number of fps our model can efficiently run inference on, e.g. when all 900 frames from a 31-sec video are predicted in 31 sec or less than that, it means that we can run it in real-time (when the identified class name is predicted and shown on the webcam frame without being affected by latency).

Forth, because we wanted to increase the security of animals and humans in domestic areas, we also implemented an automated SMS alert system based on Twilio API [195], as can be seen in Fig.4.29.

Fig. 4.29. Example of an automated SMS alert using Twilio API.

This is very helpful, especially in the cases when a wild animal is detected on private property such as a house or a farming area (e.g. when, because of hunger, a bear is coming near a flock of sheep or a fox is coming near a chicken coop) because it generates and sends an SMS alert to the phone number of the owner, informing him what animal class is detected in real-time through the webcam and thus helping him to take the necessary actions to maintain security. In

order to save the SMS costs and not send an SMS alert every time (e.g. every second) a wild animal is detected in the webcam frame, we wrote a function that sends the SMS only if the wild animal is present in front of the frame for at least 3 seconds (to make sure that there are no SMS alerts sent by mistake due to some 1-second short animal class misdetections in the webcam frame). Additionally, in case the same wild animal class was detected multiple times in the last 5 minutes, this SMS alert is sent only one time every 5 minutes (e.g. if a Bear is detected continuously in the front of the webcam for 10 minutes, the SMS alert will be sent to owner's phone only two times).

4.3.3. Motion Detection

Because we wanted to lower the power consumption on both platforms when running the DL models as much as possible, instead of buying a costly motion detection sensor, we implemented a software motion detection method based on the difference between pixel intensity of the frames, as seen in Fig.4.30.

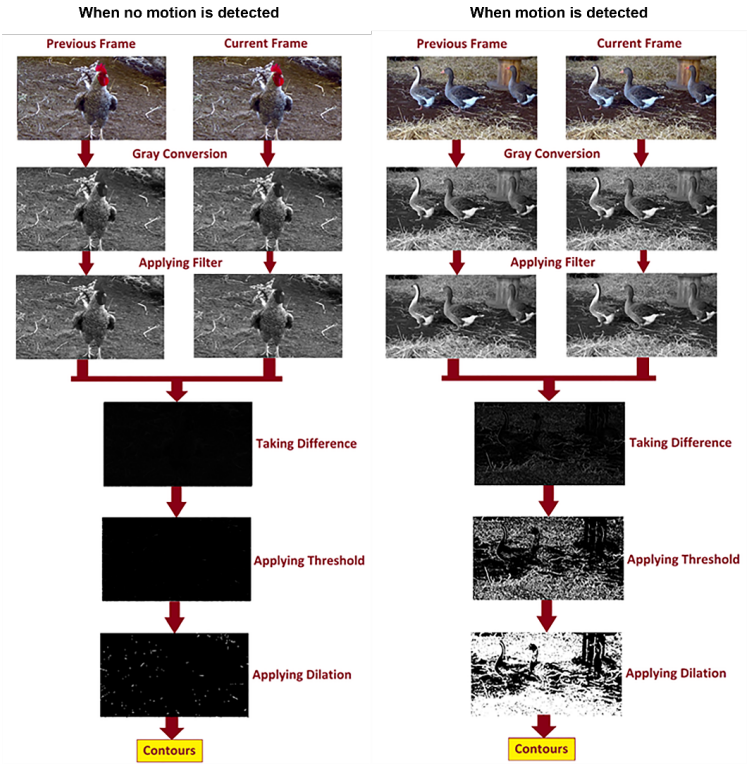


Fig. 4.30. Summarized view of the proposed motion detection.

For our motion detection method seen in Fig.4.30, in order to speed-up the inference processing time of a video or webcam frame, we reduced its size to 224×244 pixels and used several computer vision techniques using OpenCV as follows:

1) We converted the frame color image to grayscale so that we can avoid some effects of illumines. This also results in faster processing.
2) We applied the Gaussian Blur filter to remove any possible noise in the frame image.
3) We computed the absolute difference (subtraction) between the current frame (foreground) and the first frame (modeled background) in order to calculate if their pixel values were close to zero, meaning that motion was not detected, otherwise when pixel values are higher, motion is detected.
4) We applied a frame delta threshold (i.e. with a value of 25), resulting in a black and white image (no other gray light color and mid-value in the image; just black and white).
5) We applied dilation (morphological transformation) to the threshold frame in order to join the broken parts of an object in the image.
6) After that, we applied contour detection on the image and measured the available contour size. If the contour size is smaller then the given threshold, then the pixels in the frame are very similar (as seen on the left side of Fig.4.30) and frame image will not be passed to the inference process. If the contour size is higher than the threshold, then it means that the current frame is quite different from the previous frame (as seen on the right side of Fig.4.30) and the image will be passed to our model for prediction.

There are some advantages of this vision-based motion approach over motion sensors, e.g. regular motion sensors are having some drawback regarding range and time and require extra acquisition costs whereas this vision-based motion approach checks the difference between the previous and present frame in software and if something changed in the image, then it takes it as motion and sends the frame to the inference process. Another advantage is that, even though the program will run all the time having the GPU at running state, the GPU memory transfer will be zero because GPU is not computing anything when there is no significant change in the present frame compared with the previous one; in this way we protect the GPU to heat-up as well.

4.3.4. Experimental Setup and Results

We considered implementing the 4 DL model architectures both on an Acer Predator Helios 300 PH317-51-78SZ laptop with an Intel Core i7-7700HQ, 16GB DDR4 RAM memory and the Nvidia GTX 1060 GPU with 6GB GDDR5/X frame buffer, 8 Gbps memory speed and 1708 boost clock (MHz) as well as on a Nvidia Jetson TX2 board [190] having the following configuration on the hardware side: CPU: ARM Cortex-A57 (quad-core) @ 2GHz + Nvidia Denver2 (dual-core) @ 2GHz, GPU: 256-core Pascal @ 1300MHz, Memory: 8GB 128-bit LPDDR4 @ 1866Mhz | 59.7 GB/s, and Storage: 32GB eMMC 5.1. On the software side, on the Nvidia Jetson TX2 board, we used Nvidia JetPack SDK [196] with Linux Ubuntu18.04 LTS and Tensorflow 1.14.0 (Keras is used from within the tensorflow.keras) for both platforms. For the experimental results using webcam and motion detection, in the case of the laptop, we use its internal webcam, whereas, in the case of the Nvidia Jetson TX2 board, we used an external Logitech C920 HD Pro webcam. It is important to mention that with the help of the command line interface nvpmodel tool, we run all our Nvidia Jetson TX2 tests on the Max-P Core-All mode.

Following, we will show a comparison between the laptop containing the Nvidia GTX 1060 GPU and Nvidia Jetson TX2 regarding inference speed testing and also explain why frames batch testing is important when trying to run a DL model in real-time on both platforms. Finally, we will present a power usage comparison with and without the proposed motion detection on both platforms and motivate our decision for why the Nvidia Jetson TX2 is our platform of choice when designing the solar-powered real-time DL-based system.

The inference speed testing results for the Nvidia GTX 1060 GPU and Nvidia Jetson TX2 are presented in Table 20 where a different number of frames were tested on both platforms in order to evaluate the time it takes for each of the 4 DL models to classify a certain number of frames in under a second, both on a video as well as using a webcam.

Table 20. Inference Speed Testing between Nvidia GTX 1060 GPU and Nvidia Jetson TX2 on a video as well as using a webcam for VGG-19 (V), InceptionV3 (I), ResNet-50 (R) and MobileNetV2 (M) model architectures.

Number of Frames	**Nvidia GTX 1060 GPU Inference Time (Seconds)**				**Nvidia Jetson TX2 Inference Time (Seconds)**			
	V	**I**	**R**	**M**	**V**	**I**	**R**	**M**
1	0.020	0.033	0.027	0.021	0.135	0.114	0.083	0.047
2	0.029	0.030	0.066	0.021	0.223	0.145	0.115	0.062
4	0.054	0.043	0.056	0.044	0.368	0.190	0.187	0.305
8	0.106	0.065	0.080	0.056	0.503	0.289	0.332	0.385
16	0.190	0.106	0.142	0.109	0.682	0.478	0.599	0.525
24	0.304	0.158	0.227	0.177	1.107	0.682	0.898	1.059

Because the results were similar, we presented their average values only once. As can be noticed in Table 20, the inference time of the Nvidia GTX 1060 GPU is always under 1 second for all 4 DL model architectures, even with 24 fps (we also tested the GTX 1060 GPU on up to 60 fps, but it is out of scope to present these results). In comparison, when running the VGG-19 and MobileNetV2 DL models on the Nvidia Jetson TX2 platform with 24 fps, we discovered that the inference time takes more than 1 second, so we decided to run all of our Nvidia Jetson TX2 experiments presented in this paper with 16 fps for all DL architectures.

Regarding frames batch testing, we tested the effect of batch size on computing time by forwarding not just one frame but an *n* number of frame batches to our model for prediction. The frames batch testing is very important because it helps choose the fps parameter that finishes the task in the shortest amount of time with the highest number of frames (the higher the number of frames, the better the prediction) and lowest energy consumption without worries of service interruption when deploying later in a real-life scenario. Because of its 6GB dedicated RAM, we found out that the Nvidia GTX 1060 GPU can make use of 100% GPU memory utilization when running the InceptionV3 and ResNet-50 model architectures but only of 80% GPU memory utilization (higher value than this resulted in "out of memory" errors) when running the VGG-19 and MobileNetV2 model architectures in real-time. Nevertheless, the laptop containing the Nvidia GTX 1060 GPU can help all DL model architectures run the fastest prediction (when there is no latency between the present frame and predicted animal class name on the frame) at different (higher) fps and faster time values (fewer seconds) as compared with the Nvidia

Jetson TX2 which uses more than half of its memory for running the Linux framework, and which is able to run the fps batch testing at only maximum 30% of its memory utilization. The reason for this limitation is because with other ratio values it resulted in "out of memory" related errors.

In order to show the power usage comparison between the two platforms by maintaining a high inference accuracy (more fps = better accuracy), we decided to run all the experimental results presented in this paper with 30 fps and GPU memory ratio = 1 (for the InceptionV3 and ResNet-50) and 0.8 (for the VGG-19 and MobileNetV2) on the laptop containing the Nvidia GTX 1060 GPU and with 16 fps and GPU memory ratio = 0.3 for all 4 DL model architectures on the Nvidia Jetson TX2.

Because the final goal is to run the inference in real-time on real-life scenarios, we decided to run the experiments regarding power usage only using the webcam and not also on a video like in the previously described experiments.

We calculated the power consumption for the Nvidia GTX 1060 GPU on our Linux laptop by running the command "sudo powerstat" and for the Nvidia Jetson TX2 board by using a convenient power measurement script [197] and also by using the command e.g. "sudo ./tegrastats".

We run the experimental results for 5 hours (30 samples/values taken every 10 minutes) for each of the 4 DL models, both with and without motion detection for both platforms and presented the results in Figs.4.31 and 4.32.

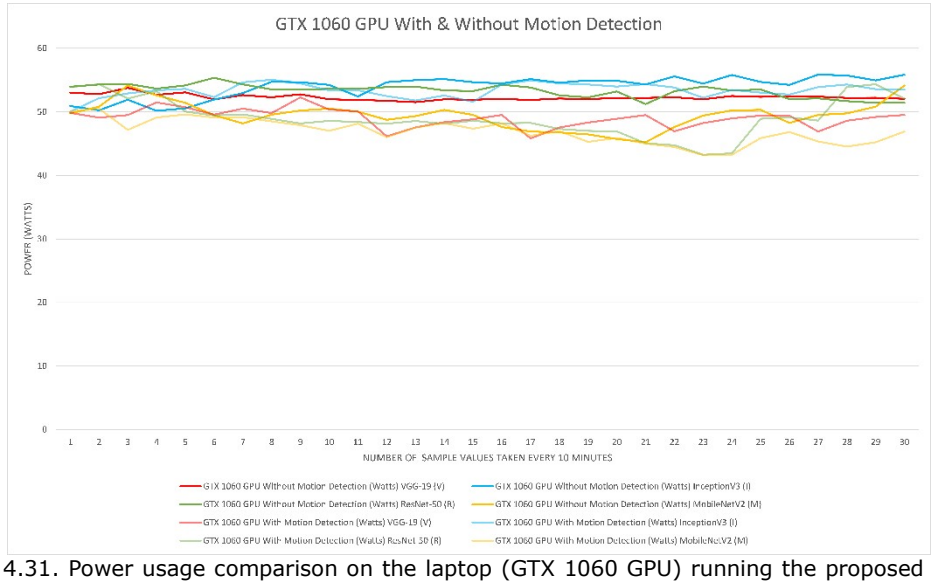


Fig. 4.31. Power usage comparison on the laptop (GTX 1060 GPU) running the proposed real-time animal class identification implementation during a 5 hours test using the webcam without and with motion detection method for VGG-19 (V), InceptionV3 (I), ResNet-50 (R) and MobileNetV2 (M) architectures. The y-axis represents the Watts value and the x-axis represents the total number of sample values taken every 10 minutes.

Without using the proposed motion detection method, the maximum power consumption of the laptop (Nvidia GTX 1060 GPU) was 24.79W when in idle state, 53.79W when running the VGG-19 model, 55.88W when running the InceptionV3 model, 55.36W when running the ResNet-50 model and 54.15W when running the MobileNetV2 model. Also, the maximum power consumption of the Nvidia Jetson

TX2 without using the proposed motion detection method was 4.11W when in idle state, 14.77W when running the VGG-19 model, 12.87W when running the InceptionV3 model, 11.74W when running the ResNet-50 model and 10.47W when running the MobileNetV2 model.

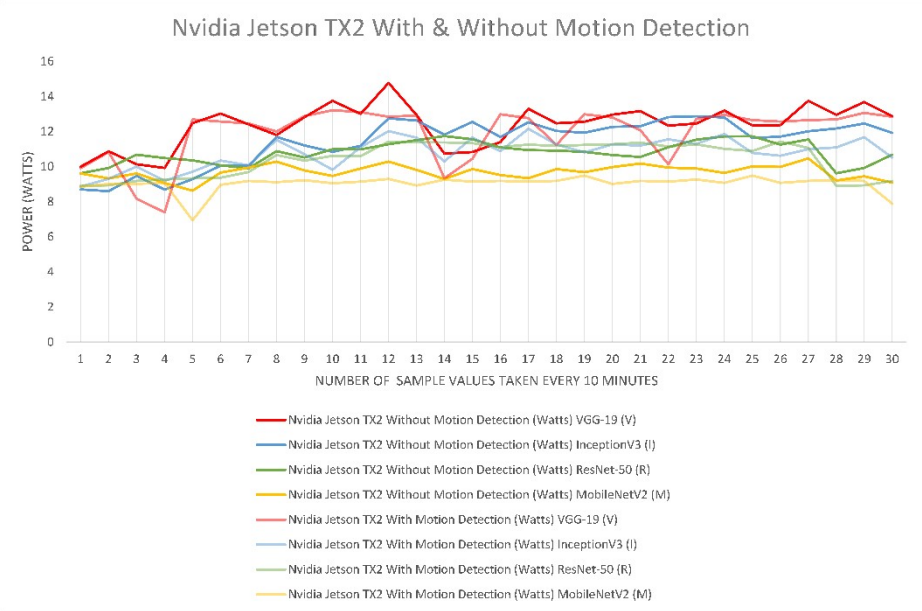


Fig. 4.32. Power usage comparison on the Nvidia Jetson TX2 board running the proposed real-time animal class identification implementation during a 5 hours test using the webcam without and with motion detection method for VGG-19 (V), InceptionV3 (I), ResNet-50 (R) and MobileNetV2 (M) architectures. The y-axis represents the Watts value and the x-axis represents the total number of sample values taken every 10 minutes.

With the proposed motion detection method, the power consumption is lower for both platforms, justifying our decision to implement it. More exactly, the maximum power consumption of the laptop (Nvidia GTX 1060 GPU) when using the proposed motion detection method was 52.58W when running the VGG-19 model, 55.06W when running the InceptionV3 model, 54.35W when running the ResNet-50 model and 50.51W when running the MobileNetV2 model. Also, the maximum power consumption of the Nvidia Jetson TX2 when using the proposed motion detection method was 13.22W when running the VGG-19 model, 12.16W when running the InceptionV3 model, 11.43W when running the ResNet-50 model and 9.49W when running the MobileNetV2 model. It is important to mention that in the case of the laptop we used the existent internal webcam, whereas for the Nvidia Jetson TX2 we used the Logitech C920 HD Pro webcam having an input voltage range from +9V to +15V DC and which was powered directly from the embedded board itself. Also, it can be observed that for both platforms, the motion detection method reduces the energy consumption by around 5%.

Considering the experimental results from Figs.4.31 and 4.32 which show that the laptop containing the Nvidia GTX 1060 GPU consumes around 5 times more energy than the Nvidia Jetson TX2 and because we wanted to minimize the investment in the improvement of our solar tracker (which otherwise, in the case of laptop would have required a 5× increase in the number of solar cells and solar

panel size, as well as updating the entire circuitry), we decided to make the Nvidia Jetson TX2 as the platform of choice for our solar-powered real-time DL-based system. One of the main reasons for implementing an efficient solar-powered real-time DL-based system is the consideration of recent efforts regarding climate change [8-10, 49] as well to bring awareness to future researchers about the possibility and necessity to use alternative sources of renewable and green energy such as that from the sun when designing real-time DL-based systems.

As seen previously in Figs.4.31 and 4.32, the maximum power consumed by the Nvidia Jetson TX2 was that of 14.77W without using motion detection and 13.22W when using motion detection for the VGG-19 model architecture during a 5 hours test. The architecture that had the lowest power consumption during the 5-hour test was the MobileNetV2 model architecture, with 9.69W when not using motion detection and 9.03W when using motion detection.

In order to make our Nvidia Jetson TX2 board also autonomous from the energy needs point of view when running inference using the 4 DL model architectures [15] in real-time, instead of using a traditional power plug, we decided to connect it to our previous proposed solar tracking device that uses the Cast-Shadow principle [17] which we updated and described earlier. A diagram block of the summarized autonomous solar-powered real-time DL-based system can be seen in Fig.4.33 below.

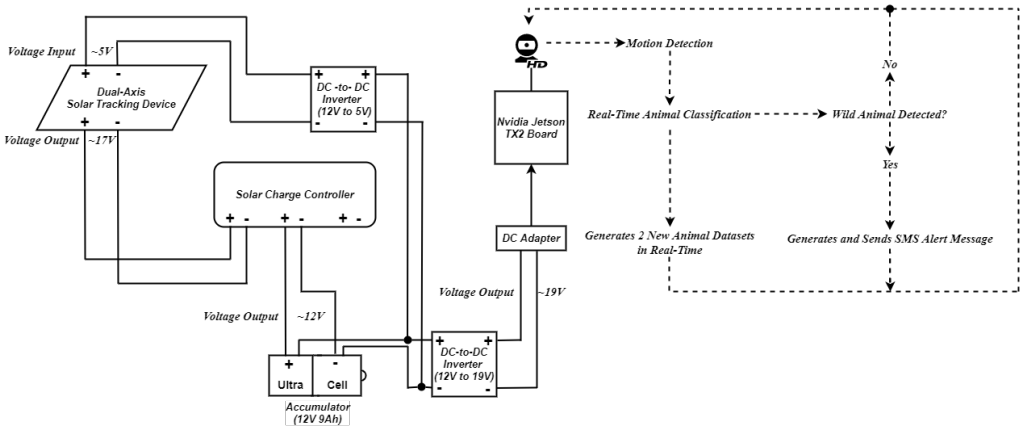


Fig. 4.33. Connection diagram of the proposed autonomous solar-powered real-time DL-based system.

Our improved solar panel comes equipped with 60 polycrystalline cells that are able to provide a maximum output voltage of around 17V, as can be seen in Table 21. The increase from 40 to 60 in the number of PV solar cells is justified by the fact that it reduces the risk of voltage drops below 12V in order to keep the battery charged continuously even under extreme weather conditions (e.g. cloudy days).

Table 21. Energy generated by our solar tracker when the Nvidia Jetson TX2 is running the VGG-19 (V), InceptionV3 (I), ResNet-50 (R) and MobileNetV2 (M) model architectures in real-time using the external webcam with motion detection during a 5 hours test time.

Energy Generation of our Solar Tracker				
Test Time (Hour)	**V**	**I**	**R**	**M**

	Voltage Gain [V]			
9:00	17.3	16.98	16.67	16.35
10:00	16.03	16.3	16.59	17.06
11:00	17.14	17.06	16.99	16.91
12:00	16.83	16.64	16.46	16.29
13:00	16.1	16.08	16.07	16.05
Avg. Value	**16.68**	**16.61**	**16.55**	**16.53**
	Current Gain [A]			
9:00	1.34	1.36	1.37	1.39
10:00	1.4	1.22	1.04	0.86
11:00	0.67	0.67	0.66	0.66
12:00	0.66	0.64	0.64	0.75
13:00	0.66	0.65	0.65	0.69
Avg. Value	**0.94**	**0.90**	**0.87**	**0.87**
	Power Gain [W]			
9:00	23.18	23.09	22.83	22.72
10:00	22.44	19.88	17.25	14.67
11:00	11.48	11.43	11.21	11.16
12:00	11.10	10.64	10.53	12.21
13:00	10.62	10.45	10.44	11.07
Avg. Value	**15.76**	**15.09**	**14.45**	**14.36**

According to the schematic presented in Fig.4.33, we linked the output of the PV solar panel to the dedicated solar module input of the solar charge controller in order to obtain the parameter readings (voltage and current) from our solar tracking device (generator) and storage component (accumulator). The solar charge controller is a robust all-in-one control system that provides three input-output ports: one dedicated to solar modules, one dedicated to feeding the battery from the PV panel with collected voltage, and one output module for connecting a current load. Since our main objective is to store solar energy in the accumulator, we only use two of the available ports.

A few notable features of the solar charge controller are microcontroller unit control, built-in timer, settable voltage and full protection from overvoltage, overcurrent, etc. The Ultra Cell accumulator is a 12V, 9Ah acid-plumb battery that is generally used nowadays in UPS systems to provide energy for desktop systems in case of local power outages. Due to its chemical composition and charging current of around 1A, the charging and discharging time can be analyzed both theoretically as well as in real-time scenarios. The main formula that is generally used in charging time calculus is given by the following equation (4.5a):

$$T=Ah/A \tag{4.5a}$$

where T represents the charging time, Ah depicts the Ampere hour rating of the battery and A denotes the charging current in Amperes. In our experimental results, first, we calculated the charging current for the 9Ah battery in theory as well as in practice:

1) As we know, in theory, the charging current should be 10% of the battery's Ah rating. Therefore, charging current for a 9Ah Battery = 9 Ah × (10/100) = 0.9 Amperes. However, due to some possible current losses that can appear on the battery, instead of exactly 0.9 Amperes, we consider only a value between 0.9 and 1.1 Amperes for the charging purpose. Supposing we

take 1 Amp for charging purposes, so charging current for 9Ah Battery = 9/1 = 9 Hrs, a situation that usually occurs only in theory

2) As we know, in practice, it has been noted that 40% of losses occur in the case of battery charging. Consequently, the formula will be: 9 × (40/100) = 3.6 resulting in 9Ah x 40% of losses. Therefore, 9 + 3.6 = 12.6Ah resulting in 9Ah + Losses. According to formula (4.5b), we will now substitute the new values and obtain:

12.6/1=12.6Hrs (4.5b)

Therefore, because the accumulator requires 1A charging current, its 9Ah capacity takes almost 13 Hrs to fully charge with solar energy from the solar tracker. However, because our solar-powered real-time DL-based system does not drain any solar energy during the night time, this does not influence our experimental outcomes. Consequently, the total discharging time of the accumulator can be determined by considering the 40% losses and by applying the following formula (4.5c):

12.6/0.6=21Hrs (4.5c)

Since our accumulator is limited to a 12V storage capacity, as can be seen in Fig.4.33, we used two voltage inverters. The first DC-to-DC inverter was interconnected in parallel so that the battery's output voltage would be increased to around 19V as can be seen in Table 22, in order to satisfy the Nvidia Jetson TX2 board's (consumer) supply voltage requirements in a real-life scenario.

Table 22. Energy stored by our accumulator using the solar tracker when the Nvidia Jetson TX2 is running the VGG-19 (V), InceptionV3 (I), ResNet-50 (R) and MobileNetV2 (M) model architectures in real-time using the external webcam with motion detection during a 5 hours test time.

Energy Storage of our Solar Tracker				
Test Time (Hour)	**V**	**I**	**R**	**M**
	Voltage [V]			
9:00	12.8	12.74	12.7	12.66
10:00	12.6	12.66	12.71	12.75
11:00	12.8	12.8	12.79	12.78
12:00	12.78	12.76	12.75	12.74
13:00	12.73	12.76	12.8	12.87
Avg. Value	**12.74**	**12.74**	**12.75**	**12.76**
	Charging Current [A]			
9:00	0.84	0.87	0.89	0.92
10:00	0.94	0.86	0.78	0.65
11:00	0.9	0.87	0.84	0.82
12:00	0.92	0.85	0.8	0.79
13:00	0.88	0.83	0.81	0.8
Avg. Value	**0.89**	**0.85**	**0.82**	**0.79**
	Power [W]			
9:00	10.75	11.08	11.30	11.64
10:00	11.84	10.88	9.91	8.28
11:00	11.52	11.13	10.74	10.47
12:00	11.75	10.84	10.2	10.06

13:00	11.20	10.59	10.36	10.29
Avg. Value	**11.41**	**10.90**	**10.50**	**10.14**
Voltage Readings for DC-to-DC Inverter (12V to 19V)				
	Voltage Output [V]			
9:00	19.20	19.15	19.16	19.18
10:00	19.17	19.14	19.12	19.10
11:00	19.09	19.10	19.11	19.05
12:00	19.02	19.04	19.05	19.06
13:00	19.03	19.02	19.07	19.00
Avg. Value	**19.10**	**19.09**	**19.10**	**19.07**

The second DC-to-DC inverter was connected between the energy storage element and the back of our solar panel in order to power the automation equipment (1× Arduino UNO, 1× Optocoupler, 2× L298N, 2× stepper motors) directly from the accumulator. Due to the implemented mechanical blocking elements, when in idle state, our solar tracking device consumes less energy (0.32W) with the Arduino UNO and L298N ICs and reaches 2W power consumption [17] when it updates its position to optimize sun ray exposure (a process which usually takes up to 5 seconds).

This 2W power consumption can be successfully covered by the accumulator's solar energy provision, proving that our entire solar-powered real-time DL-based system can run 100% using renewable and green energy from the sun. Finally, we linked the output of the first DC-to-DC inverter to the input of the Nvidia Jetson TX2 board with the help of a dedicated DC adapter, as seen in Fig.4.33 as well.

The experimental cases were carried out with our previously described setup over a 5 hours time span for each of our previously trained architectures (VGG-19, InceptionV3, ResNet-50, and MobileNetV2) [15] during 4 days test time. Our results show that the output voltage and current values of our solar panel are always maintained at an optimum level despite changing weather conditions (e.g. partial clouds in the afternoon).

Also, regarding the energy requirement of the Nvidia Jetson TX2 with the external webcam using the implemented motion detection method during a 5 hours test, we present the results in Table 23.

Table 23. Energy requirements for the Nvidia Jetson TX2 when running the VGG-19 (V), InceptionV3 (I), ResNet-50 (R) and MobileNetV2 (M) model architectures in real-time using the external webcam with motion detection during a 5 hours test time.

Energy Requirement of the Nvidia Jetson TX2 with External Webcam and using Motion Detection				
Test Time (Hour)	**V**	**I**	**R**	**M**
	Voltage Draw [V]			
9:00	19.1	19.1	19.1	19.09
10:00	19.08	19.08	19.08	19.07
11:00	19.07	19.07	19.07	19.06
12:00	19.07	19.08	19.07	19.08
13:00	19.07	19.07	19.06	19.05
Avg. Value	**19.07**	**19.07**	**19.07**	**19.07**
	Current Draw [A]			
9:00	0.58	0.55	0.51	0.46
10:00	0.52	0.51	0.49	0.46
11:00	0.56	0.62	0.52	0.47

12:00	0.42	0.56	0.53	0.47
13:00	0.66	0.54	0.52	0.47
Avg. Value	**0.54**	**0.55**	**0.51**	**0.46**
	Power Consumption [W]			
9:00	11.07	10.50	9.74	8.78
10:00	9.92	9.73	9.34	8.77
11:00	10.67	11.82	9.91	8.95
12:00	8	10.68	10.1	8.96
13:00	12.58	10.29	9.91	8.95
Avg. Value	**10.44**	**10.60**	**9.8**	**8.88**

These results prove that a real-time DL-based system can easily take advantage of renewable and green energy sources such as solar energy from a solar tracking device in order to become self-sustaining from the energy needs point of view. More exactly, we can observe that the improved solar tracker generates in average around 15 Wh, the accumulator stores around 11 Wh and the Nvidia Jetson TX2 board consumes not more than around 10 Wh when running all 4 DL models with the motion detection method in real-time.

The experimental cases were considered relevant for our work due to the fact that the DL-based system can run autonomously using free energy from the portable solar tracker, thus eliminating the need of connecting it to an AC network. In order to check the working conditions and take full control of the Nvidia Jetson TX2 board when connected to our solar tracker, we made use of a 7-inch portable monitor that was connected with the help of an HDMI as well as a micro-to-USB cable to the Nvidia Jetson TX2 board, as can be seen on the right side of Fig.4.28.

5. ENVIRONMENTALLY-FRIENDLY METRICS FOR DEEP LEARNING

With unprecedented growth in the number of platforms, e.g. CPUs, GPUs and FPGAs as well as in the number of DL algorithms, architectures, and frameworks such as Tensorflow and PyTorch, the need for a fair comparison between DL-based systems when performing training or inference by using appropriate metrics is crucial.

Until recently, it was difficult to fairly compare DL models due to the inexistent standard evaluation criteria. In the last years, efforts to deliver efficient tools for benchmarking DL implementations were made by various researchers from both academia and industry, an example in this direction being the MLPerf Benchmark [198] introduced initially (in 2018) only for training but very recently (in November 2019) also regarding inference [199] and being supported by a group of 40 organizations like e.g. Google and Microsoft. Regarding training, when measuring the performance of DL implementations, there were many types of metrics used in prior DL benchmarks, i.e. throughput (samples per second), but recently Time-To-Accuracy (TTA), an end-to-end training time to a specified validation accuracy level, is the accepted metric in the DL community, standardized initially by DAWNBench [200] and being also the main metric used in MLPerf. A consequence of this race towards occupying the first place in a Benchmark with the TTA as a metric for training is that the state-of-the-art DL models consume an enormous amount of energy, affecting the climate change and limiting the AI innovation, with a report from Allen Institute for AI [8] arguing that energy efficiency should be considered a more common evaluation criterion for AI papers, at least as important as accuracy and that the focus on a single metric is detrimental to our society, economy, and environment. In response to vast increases in computational capacity and energy needs, with Nvidia's recent NLP oriented Megatron Project, especially GPT-2 8B, a large and powerful Transformer-based language model that required 512 GPUs for training 8.3 Billion parameters [7], the massive impact which training such DL models have on the environment should be taken very seriously into consideration, with recent work in [9] even concluding that there is a very significant carbon footprint to DL.

Despite there being many available DL benchmarks [198, 199, 201, 202] that consider various metrics like time, cost, utilization, memory footprint, throughput, timing breakdown, strong scaling and communication as well as latency and load balancing, only MLPerf Benchmark is having energy as a metric for training (planning to improve the metric regarding measuring power in the inference benchmark only in a future update), with Deep500 [201], a benchmark introduced in 2019, planning to adopt energy as a metric only in the near future as well. Although training of DL models has considerable costs, with hardware (e.g. large mini-batch training [203] and reduced precision [204]), software (e.g. cuDNN [161]) and statistical (e.g. Adam Optimizer [205]) optimizations being proposed in the past for improving the computational performance of DL, a critical workload is and will always be the inference process. A reason for this is because the training of DL models is usually done once, whereas during the inference process because the

DL models are moved from the research side to the practical side, they are required in some cases (i.e. at Facebook) to serve around 200 trillion queries and perform more than 6 billion translations every day [206]. The growing computational demands of inference are pushing more than 100 companies to produce and optimize chips for inference; by comparison, only 20 companies are targeting training [199].

Considering these aspects, we strongly believe in the necessity of incorporating in the next generation DL benchmarks the ability to take into account the energy consumption that a DL system has when training or running inference. Furthermore, we think that it should be taken into account also the autonomy of such a system, i.e. its ability to work independently of a traditional power grid source and instead is able to use 100% green energy such as solar energy. For a more scalable and sustainable future, especially considering the emerging focus of Green AI [8], we propose four DL metrics, two for inference called APC and APEC and two for training called TTCAPC and TTCAPEC.

5.1. The Proposed Deep Learning Metrics for Inference and Training

The current most well-known DL metrics such as accuracy, F1-Score, and others fail in evaluating the performance of a DL-based system with regard to its impact on the environment due to the energy consumption when running inference (e.g. when two DL-based systems have the exact same accuracy but one of them will consume 10× more energy than the other, the existent DL metrics would consider them equal).

To solve the problem of lacking in accountability in energy consumption and costs, in this section, we will propose two new metrics: APC to tackle the problem of energy consumption and APEC to tackle the problem of energy cost. With regard to the APEC metric, we believe that this metric will encourage future researchers to use only green (e.g. solar) energy when running inference with their DL-based system [16].

We want the APC and APEC metrics to comply with the following important properties: Output range from 0 to 1; 100% accuracy and 0 energy consumption/cost imply the value of the metric is 1; 0% accuracy implies the metric is 0 regardless of energy consumption/cost; The value of the metric increases with accuracy and decreases with energy consumption/cost; Consumption/cost from inaccurate inferences are weighted more heavily. We consider these to be the most important requisites for a combination of two measures into one metric. Since it is a metric, it is desirable that it ranges from 0 to 1, so that it can be expressed in terms of percentage and give some sense of how close or distant the value of the metric is from the ideal (i.e. 1) result. When combining two measures into a single metric it is important to consider how we want each measure to influence the metric. Since lower consumption is desirable, consumption should lower the final metric, and since higher accuracy is desirable, accuracy should increase the final metric. We also want it to convey some common-sense properties: If the DL-based system running inference has 0% accuracy it doesn't matter how much or little it costs because we won't use it, and an inaccurate inference is a complete waste of energy by itself, so it makes sense to penalize its cost more heavily.

5.1.1. Weighted Consumption/Cost

With the previous properties in mind, we define a common function presented in equation (5.1) for both metrics. It is a prerequisite in order to be able to create the final APC and APEC metrics.

$$WC_\alpha(c, acc) = 2c((1-\alpha)(1-acc) + \alpha.acc) \quad (5.1)$$

This is a function $WC_\alpha(c, acc)$ to weight energy consumption/cost differently between accurate inferences and inaccurate ones, where *c* is the energy consumption/cost of a system, which could be measured per inference or per unit of time, *acc* is the accuracy of the model and α is a parameter (ranges from 0 to 0.5) that controls how much weight is assigned to accurate inferences (i.e if α= 0 the weight assigned to accurate inferences is 0; if α= 0.5, the weight assigned is the same in all cases / for accurate as well as inaccurate inferences).

The function WC_α has the following properties:

- If the system *a* has a higher energy consumption/cost than system *b* and both have the same accuracy the weighted consumption/cost of *b* is lower or the same;

$$if\ c_a > c_b\ then\ 2c_a((1-\alpha)(1-acc) + \alpha.acc) > 2c_b((1-\alpha)(1-acc) + \alpha.acc)$$

$$then\ WC_\alpha(c_a, acc) > WC_\alpha(c_b, acc)$$

- If the system *a* has better accuracy than system *b* and both consume/cost the same the weighted consumption/cost of *a* is lower or the same;

$$WC_\alpha(c, acc) = 2c(acc(2\alpha - 1) + 1 - \alpha)\ and\ (2\alpha - 1) \leq 0$$

$$if\ acc_a > acc_b\ then\ acc_a(2\alpha - 1) + 1 - \alpha \leq acc_b(2\alpha - 1) + 1 - \alpha$$

$$then\ 2c(acc_a(2\alpha - 1) + 1 - \alpha) \leq 2c(acc_b(2\alpha - 1) + 1 - \alpha)$$

$$therefore\ WC_\alpha(c, acc_a) \leq WC_\alpha(c, acc_b)$$

- If energy consumption/cost of a system is 0 the weighted consumption/cost is 0;

$$WC_\alpha(0, acc) = 2.0((1-\alpha)(1-acc) + \alpha.acc) = 0$$

- Consumption/cost from inaccurate inferences is weighted more heavily.

Inaccurate inferences as weighted by $(1-\alpha) \geq 0.5$, since $\alpha \leq 0.5$;

5.1.2. Accuracy Per Consumption (APC) Inference Metric

Following, we will present the APC metric. This metric is a function that takes into account not only the accuracy of a system (*acc*) but also the energy consumption of the system (*c*), as can be seen in equation (5.2):

$$APC_{\alpha,\beta}(c, acc) = \frac{acc}{\beta . WC_{\alpha}(c,acc) + 1} \quad (5.2)$$

where *c* stands for the energy consumption of the system and it's measured in Wh and *acc* stands for accuracy; α is the parameter for the WC_{α} function, the default value is 0.1; β is a parameter (ranges from 0 to infinity) that controls the influence of the cost in the final result: higher values will lower more heavily the value of the metric regarding the cost. The default value is 1. It is important to mention here that, as a rule of thumb, our recommendation is to use a value for β in the ballpark of $1/avg$ where avg is the average cost of the systems to evaluate. This average cost is among different systems that perform the same task, not each individual cost average from a system to measure. For example, if the commonly used methods to solve a task have an average cost of B, then, when measuring the APC for these systems, in order to compare them to our own, we would use as β the value 1/B.

In the APC metric "*c*" means **consumption** and is proposed to be a measure of the energy consumption of a single inference in a system, having its value always greater than 0. The APC metric's properties are the following:

- Ranges from 0 to 1;

$$acc \leq 1 \; and \; WC_{\alpha}(c, acc) \geq 0 \Rightarrow APC_{\alpha,\beta}(c, acc) \leq \frac{1}{0+1} = 1 \Rightarrow APC_{\alpha,\beta}(c, acc) \leq 1$$

$$acc \geq 0 \; and \; WC_{\alpha}(c, acc) \geq 0 \Rightarrow APC_{\alpha,\beta}(c, acc) \geq 0$$

- 100% accuracy and 0 energy consumption imply the APC is 1;

$$APC_{\alpha,\beta}(0,1) = \frac{1}{\beta . WC_{\alpha}(0,1) + 1} = \frac{1}{0+1} = 1$$

- 0% accuracy implies an APC of 0 regardless of energy consumption;

$$APC_{\alpha,\beta}(c,0) = \frac{0}{\beta . WC_{\alpha}(c,0) + 1} = 0$$

- APC increases with accuracy and decreases with energy consumption; The nominator increases with accuracy (since it's accuracy itself) and the denominator decreases with accuracy (or stays constant) and increases with consumption, therefore the premise is valid.

- Consumption from inaccurate inferences is weighted more heavily, as can be seen earlier regarding the proof for WC_{α}.

In order to see how accuracy and consumption affect the APC value, we plot APC over the consumption for different values of accuracy. In Fig.5.1 we can see most of the properties demonstrated in this section.

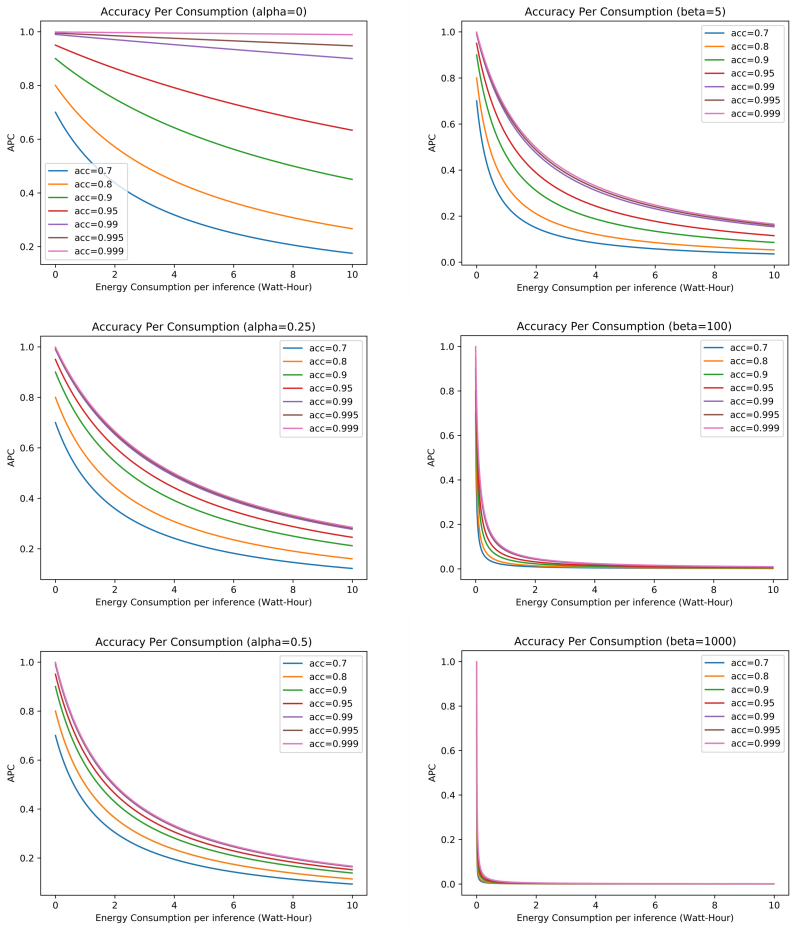


Fig. 5.1. How different values of α and β affect the APC metric.

Where α is 0, the consumption is not measured for correct inferences which imply that a model with 100% accuracy will not be penalized by its consumption (e.g. the constant pink colored line that is seen on the top-left side of Fig.5.1) as compared to where α is 0.25 and 0.5.

Similarly, in order to show how different values of β affect the APC metric, some variations in β are presented on the right side of Fig. 5.1 where β is 10, 100, and 1000. We can see how the higher the β the heavier the impact of the energy consumption is in the value of the APC metric.

5.1.3. Accuracy Per Energy Cost (APEC) Inference Metric

Following, we will present the APEC metric. The function presented in equation (5.3) for this metric is in appearance the same as the APC function.

$$APEC_{\alpha,\beta}(c, acc) = \frac{acc}{\beta \cdot WC_{\alpha}(c, acc) + 1} \tag{5.3}$$

However, in practice, the two metrics are fundamentally different. Here, the main difference lays in the meaning of the input "*c*". In APEC, "*c*" means **cost** and is proposed to be a measure of the energy cost of a single inference in a system, therefore, it is measured in different units and in different ranges. In Germany, for example, 1 kWh of energy costs 30.5 cents EUR [207]. Therefore, if our system pays 100Wh of energy for each inference, the cost "*c*" of our system will be 3.05 (cents EUR). However, it is possible to set up a system in which one doesn't pay for the energy, for example, if the energy it consumes is a renewable type of energy such as the green energy, e.g. solar energy that comes from the sun with the help of a solar tracker [16]. In these kinds of systems, the cost of electricity would be 0, and the APEC of these systems would be the same as the accuracy. Only in these cases would it be theoretically possible to obtain 100% APEC.

The APEC metric's properties are all the same as the APC metric's properties presented earlier. The only difference is the meaning of *c*, which here means cost, thus the impact that different values for α and β have on the APEC metric is similar to the APC metric, as seen earlier in Fig.5.1.

5.1.4. Time To Closest Accuracy Per Measured Energy (TTCAPME) Training Metric

Following, we will define a metric called Time to closest Accuracy Per Measured Energy (TTCAPME) that takes into account the energy consumption/cost of a model, its accuracy, and the time it takes to train it up to that point. We also want to be able to compare with this metric for the same problem both different models and different systems.

For this, we define a delta in accuracy (δ_a) and another one in energy consumption/cost (δ_e) for each problem, such that variations within that delta are considered negligible. For example, if the accuracy delta (δ_a) is 0.01 and the energy delta (δ_e) is 0.1, then a model with 0.924 accuracy and 1.12 energy consumption/cost and a model with 0.921 accuracy and 1.18 energy consumption/cost would be considered equally good.

Having defined both deltas, the grid is formed by the intervals of accuracy and energy consumption/cost, and the value in each element of the grid is the Accuracy Per Measured Energy (APME) of the lowest value in that element of the grid, e.g. the element on the accuracy interval (0.25, 0.26) and energy interval (1.5, 1.6) would be APME (1.5, 0.25). APME is a function that increases with accuracy and decreases with energy consumption/cost. An example of this type of grid can be seen in Fig.5.2 where redder colors represent higher values of APC.

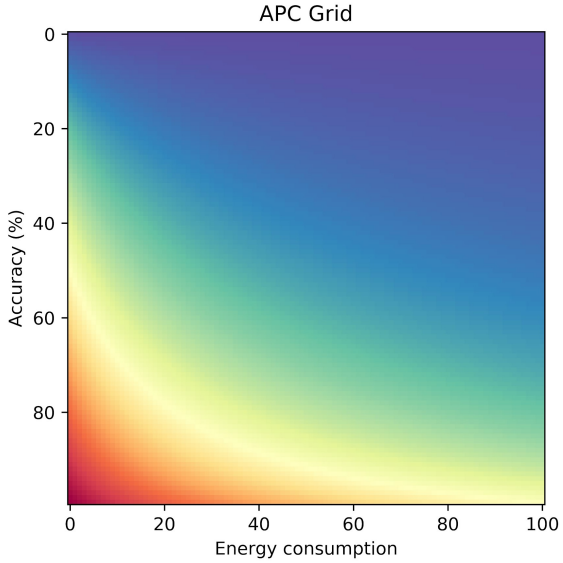


Fig. 5.2. APC Grid with energy delta (δ_e) = 1 and accuracy delta (δ_a) = 0.01. Redder colors represent higher values of APC.

The grid maps accuracies and energies to the "closest" APC values. This metric compares training times of models within the same grid interval, considering better the model that takes less time to fall into that interval.

For models on different APME values, we consider better the one with higher APME value. Then, the metric effectively maps the ternary of values (accuracy, energy consumption/cost, training time) to the ordered pair of values ("closest" APC, training time), and offers us a way to compare between these outputs.

Ordinality: In order to be able to compare between values of our metric's outputs, we need the mathematical tools to define <, =, >.

Definition 1.1: Let a_1, a_2 be real numbers between 0 and 1 and b_1, b_2 real positive numbers. Then we define the relationships between the ordered pairs $(a_1, b_1), (a_2, b_2)$ as follows:

If $a_1 = a_2$ then $(a_1, b_1) \prec (a_2, b_2)$ if and only if $b_1 > b_2$, $(a_1, b_1) = (a_2, b_2)$ if and only if $b_1 = b_2$ and $(a_1, b_1) \succ (a_2, b_2)$ if and only if $b_1 < b_2$

If $a_1 < a_2$ then $(a_1, b_1) \prec (a_2, b_2)$, and if $a_1 > a_2$ then $(a_1, b_1) \succ (a_2, b_2)$ regardless of b_1, b_2

We will prove that the set of ordered pairs with the previously defined ordinality is **well-ordered**.

Trichotomy: Since we defined the relations case by case, for two pairs only one and exactly one of the relations is true.

Transitivity: We want to prove that if v, w, and z are ordered pairs with the previously defined ordinality and v < w and w < z then v < z:

$v = (v_a, v_b), w = (w_a, w_b), z = (z_a, z_b)$

$v \prec w$, then either $v_a < w_a$ or $v_a = w_a$ and $v_b > w_b$

$w \prec z$, then either $w_a < z_a$ or $w_a = z_a$ and $w_b > z_b$

If $v_a < w_a$ then $v_a < z_a$ therefore $v < z$
If $v_a = w_a$ and $v_b > w_b$ and $w_a < z_a$ then $v_a < z_a$ therefore $v < z$
If $v_a = w_a$ and $v_b > w_b$ and $w_a = z_a$ and $w_b > z_b$ then $v_a = z_a$ and $v_b > w_b > z_b$ therefore $v < z$.

Well-foundedness: We want to prove that every nonempty set of ordered pairs has a least element, that is, it has an element x such that there is no other element y in the subset where x > y. This is easy to prove: from a set of ordered pairs we can find the elements that have the least value in the first component. Then, from these elements, we find the one with greater second component value, and that is the least element.

Parameters Properties: As mentioned earlier, this metric has two parameters, energy delta (δ_e) and accuracy delta (δ_a). Accuracy delta (δ_a) reflects inversely how important accuracy is for the model. High values for this parameter will mean that a larger range of accuracies will be grouped together as if they were the same value, therefore making smaller improvements in accuracy is not relevant. Low values for these parameters will tend to keep different accuracies separated, which will consider better models those with slightly better accuracies than others.

Following, because the training metric TTCAPME requires a function that increases with accuracy and decreases with energy consumption/cost, for simplicity, we will define two training metrics by using either APC or APEC as this function.

5.1.5. Time to closest Accuracy Per Consumption (TTCAPC) Training Metric

The objective of this metric is to combine training time and the APC inference metric in an intuitive way. The formula for TTCAPC is presented in (5.4):

$$TTCAPC_{\beta,\delta_a,\delta_e}(trainingTime, c, acc) = (trainingTime, APC_{\alpha,\beta}(rounded_{\delta_e}(c), rounded_{\delta_a}(acc))) \quad (5.4)$$

where *trainingTime* is the training time in seconds for the system, *c* is the energy **consumption** per unit of time (Wh) of the system, *acc* is the accuracy of the system and *rounded* is a function that maps values into a grid of values separated by δ, where δ is a positive real value. Some *rounded* examples:
$rounded_{0.1}$ (1.14) = 1.1; $rounded_{0.1}$ (1.08) = 1; $rounded_{0.1}$ (0.8) = 0.8;
$rounded_{0.5}$ (1.14) = 1; $rounded_{0.5}$ (1.08) = 1; $rounded_{0.5}$ (0.8) = 0.5.

This will mean that higher accuracies will be celebrated and higher net energy consumptions and higher training times will be penalized.

5.1.6. Time to closest Accuracy Per Energy Cost (TTCAPEC) Training Metric

The objective of this metric is to combine training time and the APEC inference metric. The formula for TTCAPEC is presented in (5.5) and appears the same as the one presented earlier in (5.4) for the TTCAPC, but here the meaning of *c* is different, meaning the energy **cost** of the system.

$$TTCAPEC_{\beta,\delta_a,\delta_e}(trainingTime, c, acc) = (trainingTime, APEC_{\alpha,\beta}(rounded_{\delta_e}(c), rounded_{\delta_a}(acc))) \quad (5.5)$$

Similar to TTCAPC, this will mean that higher accuracies will be celebrated, but higher energy costs and training times will be penalized.

5.1.7. Experimental Setup and Results Regarding APC, APEC, TTCAPC, and TTCAPEC Metrics

In order to realize the experiments with the above-defined metrics, we needed to measure and extract two types of data: the accuracy of the DL models and the energy consumption of the system they run training and inference on.

For this tasks, regarding the inference, we made use of one of our previously trained DL models from the work in [15], namely the MobileNetV2 as well as of the systems (i.e. Nvidia Jetson TX2 and a laptop containing an Nvidia GTX 1060 GPU) on which this DL model was running inference in real-time [16]. Regarding the training, in this case, we make use of all four DL models from [15]. It is important to mention that regarding the training time (seconds) for the Nvidia Jetson TX2, the values are simulated. For the environment used to perform the calculations of the proposed metrics, we decided to use the Python programming language due to its simplicity and availability.

We naturally want to measure the APC for different values of accuracy and power consumption, this being the reason why we run the tests on two different platforms mentioned earlier, in order to see how they stand against each other.

For this, first, we run experiments for 2 hours on both the laptop containing the Nvidia GTX 1060 GPU as well as on the Nvidia Jetson TX2 platform and feed their power consumption values into the APC equation presented earlier in (5.2), where "*c*" in this case stands for the power consumption of the system running the MobileNetV2 DL model in real-time using motion detection [16]. Because both platforms run Linux Ubuntu, these power consumption values are taken 12 times (one power consumption value every 10 minutes) with the help of "sudo powerstat" for the laptop containing the Nvidia GTX 1060 GPU and with the help of a power measurement script [197] as well as "sudo ./tegrastats" for the Nvidia Jetson TX2 platform.

Secondly, we noted the accuracy values also every 10 minutes for a total of 12 times (2 hours), but in this case, instead of measuring the inference accuracy for both platforms, we presented them only once, since presenting them for both doesn't influence our experimental results at all. Because of the weather, lighting and image quality conditions, to name only a few, it resulted in many different accuracy values, as seen in Table 24.

Table 24. APC with alpha=0.1 and beta=0.1 for our MobileNetV2 DL model [15, 16] running inference in real-time for 2 hours, with 12 samples taken every 10 minutes.

Power Consumption [W]		**Inference Accuracy [%]**	**APC [%]**	
Laptop	**Nvidia Jetson TX2**		**Laptop**	**Nvidia Jetson TX2**

50.07	8.85	99.7	65.84	91.39
50.51	9.01	92.11	49.42	79.81
47.16	9.01	91.32	49.63	78.69
49.11	9.07	94.54	54.57	83.27
49.6	6.94	50.25	13.51	36.4
49.12	8.96	25.57	5.34	15.13
49.15	9.19	80.69	34.39	64.46
48.51	9.11	47.31	12.49	31.06
47.9	9.23	60.14	18.8	42.25
47.03	9.05	85.86	41.5	71.21
48.15	9.15	99.42	65.98	90.68
46.01	9.3	98.31	64.25	88.79

This situation was very helpful in our experiment because it can be easily seen how well our metrics perform beside only with big differences in power consumption values. We used alpha=0.1 as the default and beta=0.1 since the average consumption is close to 10 and the inverse of this number is 0.1.

As we can see, the APC metric succeeds in unifying the two metrics of accuracy and energy consumption into one, and therefore it is a better metric in the cases where both accuracy and energy consumption are required to be taken into account in the final result.

We also want to measure the APEC of our DL models in order to see how they stand against each other and more importantly to see the difference between the two types of energy: green energy (solar power) and traditional energy grid.

For simplicity and because it is out of the scope of this chapter to experiment with data regarding electricity costs for all the countries in the world, we will just take Germany as an example. According to "Strom Report" (based on Eurostat data) [207], German retail consumers paid 0.00305 Euro cents for a Wh of electricity in 2017. We will use that value to calculate the cost of energy by plugging it in the equation presented in (5.3), where "*c*" in this case stands for the energy cost. In Table 25 we can see these results.

Table 25. APEC with alpha=0.1 and beta=50 for our MobileNetV2 DL model [15, 16] running inference in real-time for 2 hours with regular (paid) energy as well as with solar (free) energy.

Power Cost [Cents EUR]		**Inference Accuracy [%]**	**APEC [%]**		**APEC Green (Solar)-Powered [%]**
Laptop	**Nvidia Jetson TX2**		**Laptop**	**Nvidia Jetson TX2**	
0.1527	0.0269	99.7	55.87	87.56	99.7
0.1540	0.0274	92.11	39.74	74.58	92.11
0.1438	0.0274	91.32	40.03	73.36	91.32
0.1497	0.0276	94.54	44.65	78.37	94.54
0.1512	0.0211	50.25	9.77	31.80	50.25
0.1498	0.0273	25.57	3.77	12.46	25.57
0.1499	0.0280	80.69	26.43	58.31	80.69
0.1479	0.0277	47.31	9.01	26.31	47.31
0.1460	0.0281	60.14	13.82	36.54	60.14
0.1434	0.0276	85.86	32.64	65.36	85.86
0.1468	0.0279	99.42	56.08	86.69	99.42
0.1403	0.0283	98.31	54.36	84.50	98.31

We used alpha=0.1 as the default and beta=50 since the average cost is close to 0.03 and the inverse of this number is rounded up to 50.

As we see in Table 25, the difference is remarkable between using green energy (solar power) or not. In the cases where we use solar energy to power our DL-based systems, the APEC is in every case around 20% higher for the Nvidia Jetson TX2 platform and around 50% higher for the laptop containing the Nvidia GTX 1060 GPU in terms of absolute values. As can be observed, the APEC metric succeeds in taking into account the availability of the energy by unifying the two metrics of accuracy and energy cost into one. In this way, the APEC metric is superior in cases where not only the accuracy but also the energy cost matter in the final result.

Regarding the results presented in Table 26, for the models trained on different systems, we can see that if we choose an accuracy delta of 0.1 and energy delta of 1, the APC is different for each of them, therefore, this means that the best system is the one with the higher APC and training time is not considered.

Table 26. TTCAPC with Accuracy delta = 0.1, Energy delta = 1, beta = 0.1, alpha=0.1 for four different DL models (V-VGG-19, I-InceptionV3, R-ResNet-50, M-MobileNetV2) in two different hardware platforms.

	Laptop				**Nvidia Jetson TX2**			
	V	**I**	**R**	**M**	**V**	**I**	**R**	**M**
Accuracy	90.56	93.41	93.49	94.54	90.56	93.41	93.49	94.54
Energy Consumption (Wh)	49.9525	53.0523	50.2638	48.4508	11.6138	10.3338	9.9792	8.9069
Rounded Accuracy	90.55	93.45	93.45	94.55	90.55	93.45	93.45	94.55
Rounded Energy Consumption	48.5	53.5	50.5	48.5	8.5	10.5	9.5	8.5
Closest APC	47.721	50.503	51.839	54.879	78.243	80.084	81.190	**83.918**
Train seconds	20.273	38.853	21.396	38.847	20.273	38.853	21.396	38.847

However, in Table 27, with the same models but with larger deltas we see that two models result in the same APC, and therefore the deciding factor is the training time.

Table 27. TTCAPC with Accuracy delta = 5, Energy delta = 10, beta = 0.1 for four different DL models (V-VGG-19, I-InceptionV3, R-ResNet-50, M-MobileNetV2) in two different hardware platforms.

	Laptop	**Nvidia Jetson TX2**

	V	I	R	M	V	I	R	M
Accuracy	90.56	93.41	93.49	94.54	90.56	93.41	93.49	94.54
Energy Consumption (Wh)	49.9525	53.0523	50.2638	48.4508	11.6138	10.3338	9.9792	8.9069
Rounded Accuracy	92.5							
Rounded Energy Consumption	45	55		45	15		5	
Closest APC	52.744	48.146		52.744	73.926		85.352	
Train seconds	20.273	38.853	21.396	38.847	20.273	38.853	**21.396**	38.847

Regarding the experiments for the TTCAPEC metric, we use the same country (Germany) and price for electricity (0.00305 Euro cents for a Wh) [207] as mentioned earlier regarding the experiments with the APEC metric.

Similarly to the results regarding TTCAPC presented in Table 26, on Table 28 we can see that for the models we trained on different systems, if we choose an accuracy delta of 0.1 and energy delta of 0.001, the APEC is different for each of them, therefore the best system is the one with the best APEC and training time is not considered.

Table 28. TTCAPEC with Accuracy delta = 0.1, Energy delta = 0.001, beta = 50, alpha=0.1 for four different DL models (V-VGG-19, I-InceptionV3, R-ResNet-50, M-MobileNetV2) in two different hardware platforms.

	Laptop				Nvidia Jetson TX2			
	V	I	R	M	V	I	R	M
Accuracy	90.56	93.41	93.49	94.54	90.56	93.41	93.49	94.54
Energy Cost (cents)	0.1524	0.1618	0.1533	0.1478	0.0354	0.0315	0.0304	0.0272
Rounded Energy Cost	0.1525	0.1615	0.1535	0.14775	0.0355	0.0315	0.0305	0.0275
Rounded Accuracy	90.55	93.45		94.55	90.55	93.45		94.55

Closest APEC	37.557	40.924	42.096	45.000	68.161	74.739	75.217	78.468
Closest APEC Green (Solar) Powered	90.55	93.45		94.55	90.55	93.45		**94.55**
Train seconds	20.273	38.853	21.396	38.847	20.273	38.853	21.396	38.847

However, as seen in Table 29, with the same models but with larger deltas, all models result in the same APEC, and therefore the deciding factor is the training time.

Table 29. TTCAPEC with Accuracy delta = 5, Energy delta = 0.1, beta = 50 for four different DL models (V-VGG-19, I-InceptionV3, R-ResNet-50, M-MobileNetV2) in two different hardware platforms.

	Laptop				**Nvidia Jetson TX2**			
	V	**I**	**R**	**M**	**V**	**I**	**R**	**M**
Accuracy	90.56	93.41	93.49	94.54	90.56	93.41	93.49	94.54
Energy Cost (cents)	0.1524	0.1618	0.1533	0.1478	0.0354	0.0315	0.0304	0.0272
Rounded Energy Cost	0.15				0.05			
Rounded Accuracy	92.5							
Closest APEC	40.997				65.198			
Closest APEC Green (Solar) Powered	92.5							
Train seconds	20.273	38.853	21.396	38.847	**20.273**	38.853	21.396	38.847

It is important to mention that despite using the term accuracy in our APC and APEC metrics, both metrics can work well also by using another metric in place of accuracy (as long as it ranges from 0 to 1, meaning that 0 represents a negative score and 1 represents a positive one), such as the ones used by MLPerf Benchmark

[198, 199]. Also, the metrics proposed in this paper can work for any DL-based system; all that is needed is to have the training time, the consumption, the cost, and the accuracy measured.

5.2. Deep Learning-Based Computer Vision Application with Multiple Built-In Data Science-Oriented Capabilities

As mentioned earlier, data is at the core of every DL application. Because the ML lifecycle consists of four stages such as data management, model learning, model verification and model deployment [208], in order to collect, analyze, interpret and make use of this data, e.g. training accurate models for real-life scenarios, in recent years, new specializations were introduced in universities around the world such as ML and Data Science, to name only a few. Additionally, also new career positions were created recently such as ML Engineer and Data Scientist, being some of the top paid positions in the industry [6].

Regarding Computer Vision applications for image classification tasks, a major bottleneck before training the necessary DL models is considered to be the data collection which consists mainly of data acquisition, data labeling and improvement of the existing data in order to train very accurate DL models [209]. Another bottleneck is that, because the amount of data needed to train a DL model is usually required to be very large in size and because most of this important data is not released to the general public but is instead proprietary, the need of an original dataset for a particular DL project can be very crucial. In general, data can be acquired either by: a) buying it from marketplaces or companies such as Quandl [210] and URSA [211]; b) searching it for free on platforms like Kaggle [212]; c) crawling it from internet resources with the help of search engine crawlers [213]; d) paying to a 24/7 workforce on Amazon Mechanical Turk [214] like the creators of the ImageNet dataset did to have all of their images labeled [29]; e) creating it manually for free (e.g. when the user takes all the photos and labels them himself), which can be impossible most of the time because of a low-budget, a low-quality camera, time constraints, etc. Also, the importance of image deduplication can be seen in the fields of Computer Vision and DL where a high number of duplicates can create biases in the evaluation of a DL model, such as in the case of CIFAR-10 and CIFAR-100 datasets [215]. It is recommended that before training a DL classification model, one should always check and make sure that there are no duplicate images found in the dataset. Finding duplicate images manually can be very hard for a human user and a time-consuming process, this being the reason why a software solution to execute such a task is crucial. Some of the drawbacks of existent solutions are that they usually require the user to buy the image deduplication software or pay monthly for a cloud solution, they are big in size or are hard to install and use.

Despite all of these options, especially in the case of scraping the images from the internet, once stored they can still be unorganized or of a lower quality than expected, with images needed to be sorted out each in their respective class folder in order for the user (e.g. data scientist) to be able later to analyze and use this data for training a performant DL model. This kind of sorting task can take a tremendous amount of time even for a team, from several days or weeks to even months [216]. Another difficult task is that once the data is cleaned, organized and ready to be trained from scratch or using transfer learning, because of the variety of DL architectures, each with different sizes and training time needed until reaching

convergence [217], it can be very difficult to know from the beginning which DL architecture fits the best a given dataset and will, at the end of the training, result in a DL model that has high accuracy. Because energy consumption in DL started to become a very debated aspect in recent months, especially regarding climate change [8, 10, 16, 20, 50], the necessity of evaluating the performance of DL models also by their energy consumption and cost is very crucial.

Considering these aspects, our work introduces a DL-based Computer Vision application that has multiple unique built-in Data Science-oriented capabilities which give the user the ability to train a DL image classification model without any programming skills. It also automatically searches for images on the internet, sort these images each in their individual class folder and is able to remove duplicate images as well as to apply data augmentation in a very intuitive and user-friendly way. Additionally, it gives the user an option to evaluate the performance of a DL model and hardware platform not only by considering its accuracy but also its power consumption and cost by using the environmentally-friendly metrics APC, APEC, TTCAPC, and TTCAPEC [20].

5.2.1. The Proposed Deep Learning-Based Computer Vision Application

The proposed DL-based Computer Vision application is summarized in Fig. 5.3 and is built using the Python programming language. It is composed of the most common features needed in the Computer Vision field and facilitate them in the form of a GUI, without requiring the user to have any knowledge about coding or DL in order to be able to fully use it.

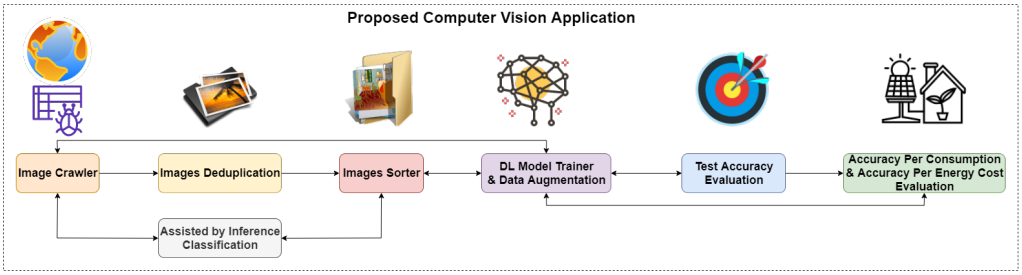


Fig. 5.3. Summarized view of the proposed Computer Vision application that incorporates features such as an automatic Image Crawler and Image Sorter assisted by inference classification, an Image Deduplicator, a DL Model Trainer with Data Augmentation capabilities as well as calculators regarding Accuracy, APC, APEC, TTCAPC, and TTCAPEC.

Regarding the system, the compilation dependencies and installation requirements of the proposed application are Python 3, Windows 10 (or later version) or Linux (Ubuntu 12 or later version). Regarding the Python libraries, we use PyQt5 for creating the GUI, HDF5 for loading DL model files, Tensorflow for training and inference, OpenCV for image processing, Numpy for data processing, Shutil for copying images in the system, TQDM for showing the terminal progress bar, Imagededup [137] for deduplication of images, Icrawler [213] for crawling the images and fman build system (fbs) for creating installers.

There are certain conventions that are common in all the features of the proposed application:

1. Model files: These are .h5 files that contain the architecture of a Keras model and the weights of its parameters. These are used to load (and save) a previously trained model in order to be able to use it.
2. Model class files: These are extensionless files that contain the labels of each of the classes of a DL model. It contains *n* lines, where *n* is the number of classes in the model, and the line *i* contains the label corresponding to the *ith* element of the output of the DL model.
3. Preprocessing function: In this convention, a preprocessing function is a function that takes as input the path to an image and a shape, loads the image from the input path, converts the image to an array and fits it to the input of the model.
4. Images folders structures: We use two different folder structures: unclassified structures and classified structures. The unclassified images folders structure is the simplest one, consisting of just one folder containing images, presumably to be classified or deduplicated. The classified images folders structure consists of a folder which in turn contains subfolders. Each subfolder represents a class of images, is named the same as the label for that class, and contains images tagged or classified belonging to that class.

Following, we will present all the built-in features: Automatic web crawler assisted by inference classification, Images deduplication, Images Sorter assisted by inference classification, DL Model Trainer with Data Augmentation capabilities, Accuracy calculator as well as the APC, APEC, TTCAPC and TTCAPEC [20] calculators.

Regarding the image crawler assisted by inference classification, the purpose of this feature is to collect images related to a keyword (representing a class) from the web and by using a classification algorithm, to make sure that the images are indeed belonging to this class. During the inference process needed for cleaning the images, a preprocessing is happening in the background, which, depending on the pretrained or custom DL model that is chosen, will resize the images, making them have the correct input shape (e.g. 28×28×1 for MNIST and 224×224×3 for ImageNet) for the DL model.

A summarized view of the implemented Image Crawler feature can be seen in Fig.5.4 and is composed of the following elements: 'Model' - a combo box containing all the existent pretrained in-built DL models such as "mnist" or "resnet50" as well as the 'Custom' option which gives the user the possibility to load his own previously trained DL model; Confidence Slider ('Confidence required') - a slider to select the minimum accuracy value to be used when classifying the images and which ranges from 0 to 99; Image Class Selector ('Select a class of images') - a combo box containing the labels of all the classes from the pretrained built-in selected DL model (e.g. 10 classes for when the "mnist" model is selected and 1000 classes when the "resnet50" model is selected). Additionally, the box contains an autocomplete search function as well; Images Amount ('Max amount to get') - a slider to select the number of images that should be crawled from the internet, ranging from 1 to 999 and 'Destination Folder' - a browser to select the path for the final location of the obtained images.

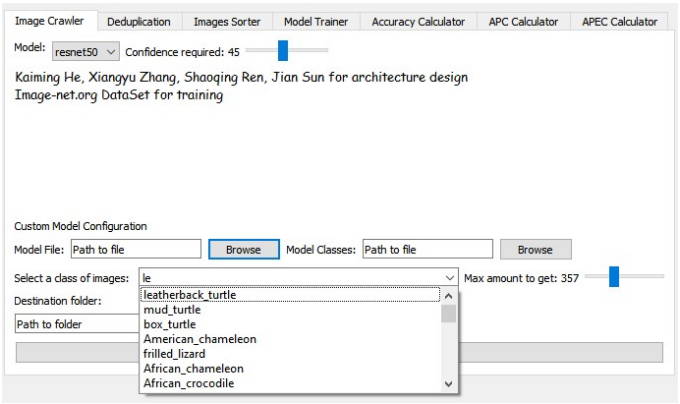


Fig. 5.4. Summarized view of the proposed Image Crawler feature assisted by inference classification.

The options under 'Custom Model Configuration' only apply when the DL model selected is "Custom" and is not built-in in the proposed Computer Vision application, e.g. when it was trained by the user itself. These options are: 'Model File' - a browser to select the .h5 file the user wishes to use for inference and Model Classes - a browser to select the extensionless file containing the name of each output class on which the selected DL model (.h5 file) was trained on. Finally, this feature's GUI interface has a button ('Add Images!') that begins the web crawling process. With the help of this feature, images are automatically crawled by the crawler and downloaded to a temporal folder location. After that, each image is classified with the selected DL model, and if the classification coincides with the selected class and the confidence is higher than the selected threshold, the image is moved to the 'Destination folder', where each image will be saved in its own class folder. This feature automatizes the population of image classification datasets by providing a reliable way of confirming that the downloaded images are clean and correctly organized.

Regarding images deduplication, the purpose of this feature is to remove duplicate images found in a certain folder. For this, we incorporated the Imagededup techniques found in [137]. A summarized view of the implemented Images Deduplication feature can be seen in Fig.5.5.

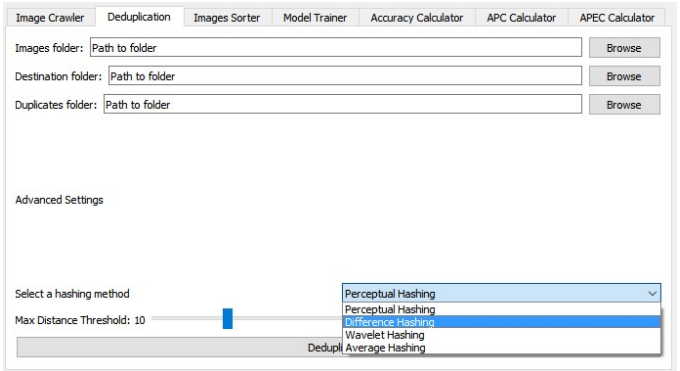


Fig. 5.5. Summarized view of the proposed Image Deduplication feature.

This feature is composed of the following elements: 'Images folder' - a browser to select the location of the folder containing the images that need to be analyzed for duplicate images; 'Destination folder' - a browser to select the location of the folder where the deduplicated images will be stored; 'Duplicates Folder' - a browser to select the location of the folder where the found duplicate images will be stored. Each duplicate image found will be stored in a subfolder. Regarding advanced settings, it is composed of: Hashing method selector ('Select a hashing method') - a combo box containing 4 hashing methods that can be used for deduplication (Perceptual Hashing (default), Difference Hashing, Wavelet Hashing, and Average Hashing) as well as a 'Max Distance Threshold' - the maximum distance by which two images will be considered to be the same (default value is 10). Finally, this interface has a button ('Deduplicate!') that begins the deduplication process according to the selected parameters.

Following, we will shortly describe the types of hashes we are using in the images deduplication feature: a) **Average Hash:** the Average Hash algorithm first converts the input image to grayscale and then scales it down. In our case, as we want to generate a 64-bit hash, the image is scaled down. Next, the average of all gray values of the image is calculated and then the pixels are examined one by one from left to right. If the gray value is larger than the average, a 1 value is added to the hash, otherwise a 0 value; b) **Difference Hash:** Similar to the Average Hash algorithm, the Difference Hash algorithm initially generates a grayscale image from the input image. Here, from each row, the pixels are examined serially from left to right and compared to their neighbor to the right, resulting in a hash; c) **Perceptual Hash:** After gray scaling, it applies the discrete cosine transform to rows and as well as to columns. Next, we calculate the median of the gray values in this image and generate, analogous to the Median Hash algorithm, a hash value from the image; d) **Wavelet Hash:** Analogous to the Average Hash algorithm, the Wavelet Hash algorithm also generates a gray value image. Next, a two-dimensional wavelet transform is applied to the image. In our case, we use the default wavelet function called the Haar Wavelet. Next, each pixel is compared to the median and the hash is calculated. Regarding this deduplication feature, first, the hasher generates hashes for each of the images found in the images folder. With these hashes, the distances between hashes (images) are then calculated and if they are lower than the maximum distance threshold (e.g. 10), then they are considered duplicates. Secondly, for each group of duplicates, the first image is selected as "original" and a folder is created in the duplicates folder with the name of the "original" folder. Then all duplicates of this image are stored on that folder. This feature successfully integrates the image deduplication technique [137] and provides a simple and quick way to utilize it.

Regarding the image sorter assisted by inference classification, this feature helps a user to sort an unsorted array of images by making use of DL models. A summarized view of the implemented Images Sorter feature assisted by inference classification can be seen in Fig.5.6 and is composed of elements similar to the ones presented earlier for the Image Crawler feature, but in this case with the function of selecting the path to the folders from which and where images should be sorted.

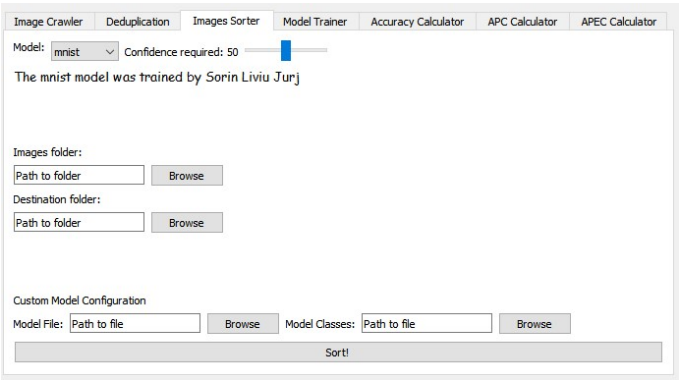


Fig. 5.6. Summarized view of the proposed Image Sorter feature assisted by inference classification.

In the destination folder, a new folder is created for each possible class, with the name extracted from the extensionless file that contains all the names of the classes, plus a folder named 'Undetermined'. Then, each image from the 'Images Folder' is automatically preprocessed, feed as input to the selected DL model and saved in the corresponding class folder. The highest value from the output determines the predicted class of the image: if this value is less than the minimum 'Confidence required', value, then the image will be copied and placed in the 'Undetermined' folder, otherwise, the image will be copied to the folder corresponding to the class of the highest value from the output. We took the decision of copying the files instead of moving them, for data security and backup reasons. This feature heavily reduces the amount of time required to sort through an unclassified dataset of images by not only doing it automatically but also removing the need to set up coding environments or even write a single line of code.

Regarding the model trainer with data augmentation capabilities, this feature gives the user a simple GUI to select different parameters in order to train and save a DL image classifier model. A summarized view of the implemented DL Model Trainer feature assisted by inference classification can be seen in Fig.5.7.

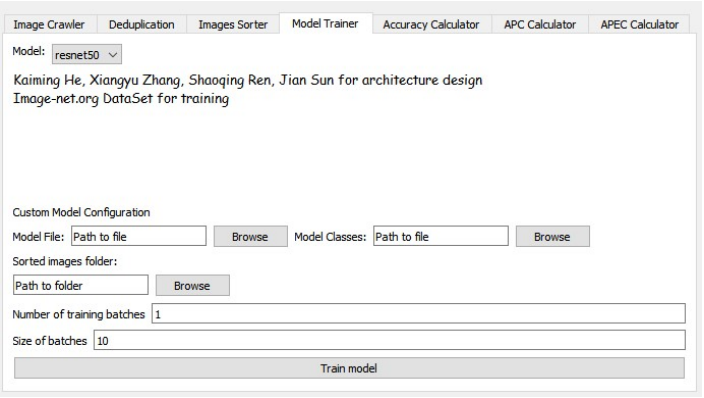


Fig. 5.7. Summarized view of the proposed DL Model Trainer feature.

This feature is composed of the following elements: 'Model' – as described earlier for the Image Crawler feature; 'Sorted images folder' - a browser to select the folder that contains the classified folder structure with the images to be trained on; 'Number of training batches' - an integer input, to specify the number of batches to train and 'Size of batches' - an integer input, to specify the number of images per batch. Regarding the custom options, they are the same as mentioned earlier regarding the Image Crawler feature. Next, this interface has a button ('Train model') that, when clicked on, prompts a new window for the user to be able to visualize in a very user-friendly way all the image transformations that can be applied to the training dataset in a random way during training. More exactly, as can be seen in Fig.5.8, the user can input the following parameters for data augmentation: Horizontal Flip - if checked the augmentation will randomly flip or not images horizontally; Vertical Flip - if checked the augmentation will randomly flip or not images horizontally; Max Width Shift - Slider (%), maximum percentage (value between 0 and 100) of the image width that it can be shifted left or right; Max Height Shift - Slider (%), maximum percentage (value between 0 and 100) of the image height that it can be shifted up or down; Max Angle Shift - Slider (degrees °), the maximum amount of degrees (value between 0 and 90) that an image might be rotated and Max Shear Shift - Slider (%), maximum shear value (value between 0 and 100) for image shearing.

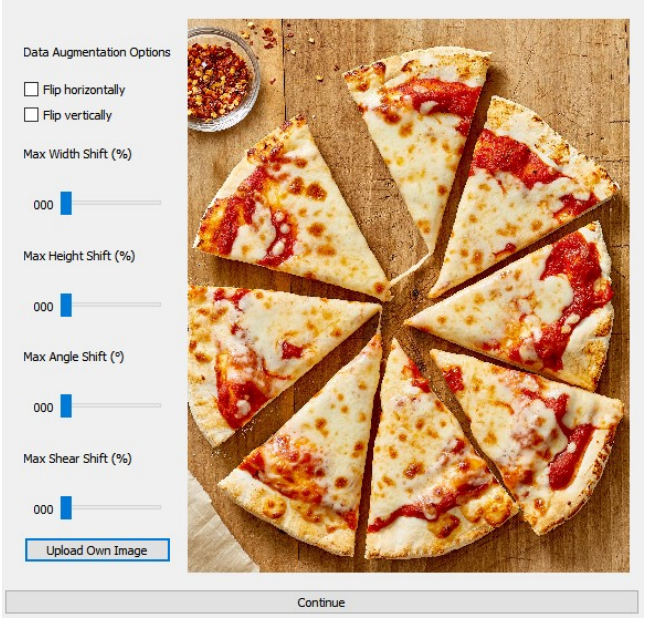


Fig. 5.8. Summarized view of the proposed Data Augmentation feature.

The data augmentation feature allows the user to visualize the maximum possible changes that can be made to an image in real-time, without the need of guessing the right parameters. Following, a training generator is defined with the selected parameters; The generator randomly takes images from the folder structure and fills batches of the selected size, for the number of batches that are selected. These batches are yielded as they are being generated. Regarding the training, first, the selected DL model is loaded, its output layer is removed, the

previous layers are frozen and a new output layer with the size of the number of classes in the folder structure is added. The model is then compiled with the Adam optimizer [205] and the categorical cross-entropy as the loss function. Finally, the generator is fed to the model to be fitted. Once the training is done, the total training time is shown to the user and a model file (.h5) is created on a prompted input location. This feature achieves the possibility of training a custom DL model on custom classes just by separating images in different folders. There is no knowledge needed about DL and this feature can later also be easily used by the Image Sorting feature described earlier in order to sort future new unsorted images.

Regarding the accuracy calculator, this section of the application GUI gives a user the option to compute the accuracy of a DL model on the given dataset in the classified images folder structure. A summarized view of the implemented Accuracy Calculator feature can be seen in Fig.5.9 and is composed of the following elements: 'Model' - as described earlier for the Image Crawler feature; 'Test images folder' - a browser to select the folder that contains the classified folder structure to measure the accuracy of a DL classification model; 'Size of batches' - an integer input, to specify the number of images per batch. The custom options are the same as mentioned earlier regarding the Image Crawler feature. Finally, this interface has a button ('Calculate Accuracy') that starts the accuracy evaluation process.

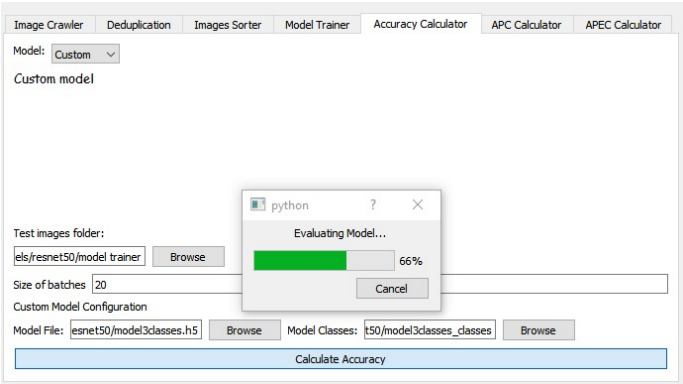


Fig. 5.9. Summarized view of the proposed Accuracy Calculator feature.

After loading the DL model and the list of classes, it searches for the classes as subfolders names in the classified images folder structure. Then, for each class (or subfolder) it creates batches of the selected batch size, feeds them to the DL model and counts the number of accurate results as well as the number of images. With these results, it calculates the total accuracy of the DL model and shows it to the user directly in the application GUI. This feature provides a simple and intuitive GUI to measure the accuracy of any DL image classification model.

Regarding the APC calculator, this GUI feature makes use of our APC metric [20] and which is a function that takes into account not only the accuracy of a system but also its energy consumption. The application GUI gives a user the option to define the values for α and β as well as to specify and calculate the accuracy and energy consumption of a DL model. A summarized view of the implemented APC Calculator feature can be seen in Fig.5.10 and is composed of the following elements: 'Model test accuracy (%)' - this widget gives a user the option to input the accuracy or to use the previously described Accuracy Calculator feature to

measure the accuracy of a DL model and 'Energy Consumption (Wh)' - float input to specify the power consumption of a user's DL model.

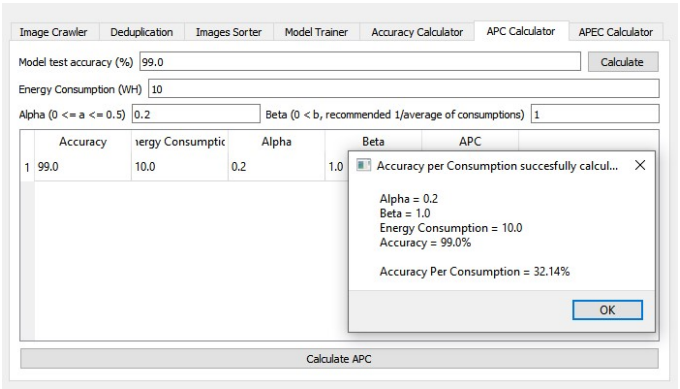


Fig. 5.10. Summarized view of the proposed APC Calculator feature.

Regarding the advanced options, it has: Alpha (α) - float input to specify the desired value of α (default 0.2) and Beta (β) - float input to specify the desired value of β (default 1). For simplicity, a table is shown with the following columns: Accuracy, Energy Consumption, Alpha, Beta, and APC. Whenever a value is changed, the table is automatically updated as well. Finally, the application GUI has a button ('Calculate APC') to begin the calculation of the APC metric. The function itself is an implementation on Numpy of our previously defined APC metric [20] and takes as input parameters the values defined in the application GUI. The implemented feature brings this new APC metric to any user by allowing them to easily calculate the APC and know the performance of their DL model with regards to not only the accuracy but also to the impact it has on the environment (higher energy consumption = higher negative impact on nature). However, the drawback of the current version of this APC calculator feature in the proposed application GUI is that the user has to measure the energy consumption of the system manually.

Regarding the APEC calculator, the APEC feature is presented in Fig.5.11 and lets a user define the values for α and β, specify or calculate the accuracy of a DL model, specify the energy consumption of the DL model and specify the cost of Wh, and calculates the resulting APEC.

The APEC feature of the proposed Computer Vision application is composed of the following elements: 'Model test accuracy (%)' – works similar to the APC widget described earlier; 'Energy Consumption (Wh)' - works also similar to the APC widget described earlier and Wh Cost - float input to specify the cost in EUR cents of a Wh. Regarding the advanced options, we have: Alpha (α) - float input to specify the desired value of α (default 0.2) and Beta β - float input to specify the desired value of β (default 1). A similar table like the one for APC Calculator is shown also here, with the following columns: Accuracy, Energy Cost, Alpha, Beta, and APEC. Whenever a value is changed, the table is automatically updated here as well. Finally, the application GUI has a button ('Calculate APEC') to begin the calculation of the APEC metric. The function itself is an implementation on Numpy of our previously defined APEC metric [20] and takes as input parameters the values defined in the application GUI.

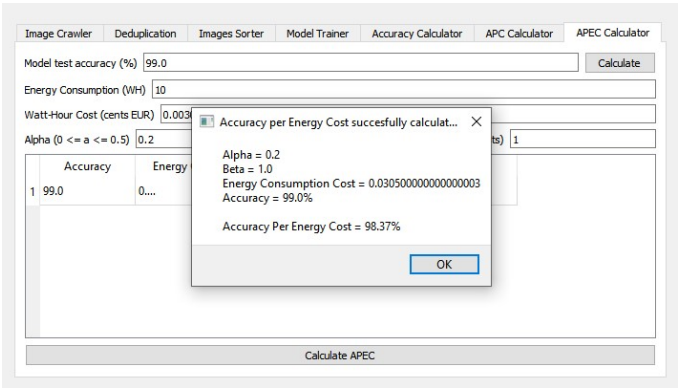


Fig. 5.11. Summarized view of the proposed APEC Calculator feature.

The implemented feature brings this new APEC metric to any user by allowing them to easily calculate the APEC and evaluate the performance of their DL model with regards to the impact it has on the environment (higher energy consumption = higher cost = negative impact on nature). However, the drawback of the current version of this APEC calculator feature is that the user has to measure the energy consumption of the system and calculate its Wh cost manually.

Regarding the TTCAPC calculator, the objective of the TTAPC metric [20] is to combine training time and the APC inference metric in an intuitive way. The TTCAPC feature is presented in Fig.5.12 and is composed of the following elements: 'Model test accuracy (%)' and 'Energy Consumption (Wh)', both working similar to the APEC widget described earlier; 'Accuracy Delta' – float input to specify the granularity of the accuracy axis; 'Energy Delta' – float to specify the granularity of the energy axis. Regarding the advanced options, they are the same as the ones presented earlier regarding the APEC feature.

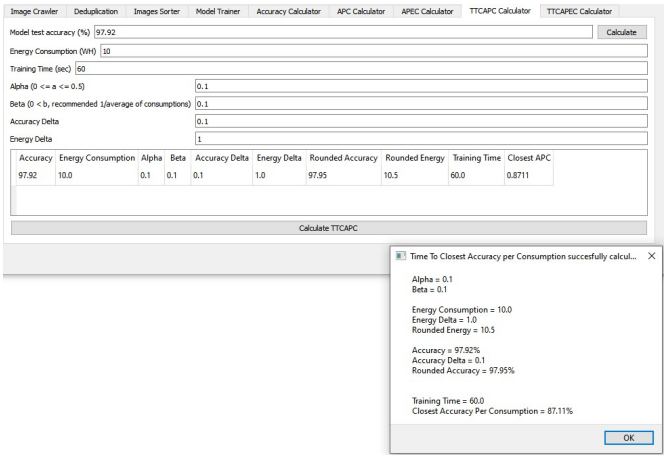


Fig. 5.12. Summarized view of the proposed TTCAPC Calculator feature.

A similar table like the one for APEC Calculator is shown also here, with the following columns: Accuracy, Energy Consumption, Alpha, Beta, Accuracy Delta, Energy Delta, Rounded Accuracy, Rounded Energy, Training Time and Closest APC. Whenever a value is changed, the table is automatically updated here as well.

Finally, the application GUI has a button ('Calculate TTCAPC') to begin the calculation of the TTCAPC metric.

Regarding the TTCAPEC calculator, the objective of the TTCAPEC metric [20] is to combine training time and the APEC inference metric. The TTCAPEC feature is presented in Fig.5.13 and is composed of the same elements like the TTCAPC feature presented earlier and one additional element called 'Energy Cost (EUR cents per Wh)' which is similar to the one presented earlier regarding the APEC metric calculator and where the user can specify the cost in EUR cents of a Wh.

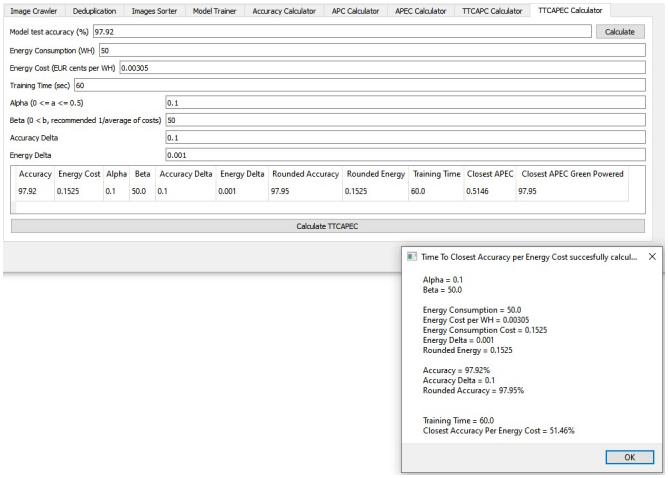


Fig. 5.13. Summarized view of the proposed TTCAPEC Calculator feature.

A similar table like the one for TTCAPC Calculator is shown also here, with the following columns: Accuracy, Energy Cost, Alpha, Beta, Accuracy Delta, Energy Delta, Rounded Accuracy, Rounded Energy, Training Time and Closest APEC. Finally, the application GUI has a button ('Calculate TTCAPEC') to begin the calculation of the TTCAPEC metric.

5.2.2. Experimental Setup and Results

Following, we will show the experimental results regarding all the implemented features in comparison with existing alternatives found in the literature and industry. We run our experiments on a Desktop PC with the following configuration: on the hardware side we use an Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz, 6 Core(s), 12 Logical Processor(s) with 32 GB RAM and an Nvidia GTX 1080 Ti as the GPU; on the software side we use Microsoft Windows 10 Pro as the operating system with CUDA 9.0, CuDNN 7.6.0 and Tensorflow 1.10.0 using the Keras 2.2.4 framework.

As can be seen in Table 30, our proposed Image Crawler feature outperforms existent solutions and improves upon them.

Table 30. Comparison between existent and the proposed Image Crawling solution.

Features	**Existent Solutions [213]**	**Proposed Solution**
Image Crawler	Yes	Yes

Built-In DL Models	No	Yes
Custom DL models	No	Yes
Cleans Dataset automatically?	No	Yes
Speed Test (sec)		
Crawling 97 Images	23	23
Cleaning 97 Images	47	10

Even though the crawling took the same amount of time, this is not the case regarding the cleaning part, where, because this feature is not available in any of the existent solutions, this needed to be done manually and took 47 seconds for a folder containing 97 images as compared to only 10 seconds for our proposed solution which executed the task automatically. A comparison between "dirty" images and clean images can be seen in Fig.5.14 where, for simplicity, we searched for 97 pictures of "cucumber", which is one class from the total of 1000 classes found in the ImageNet dataset.

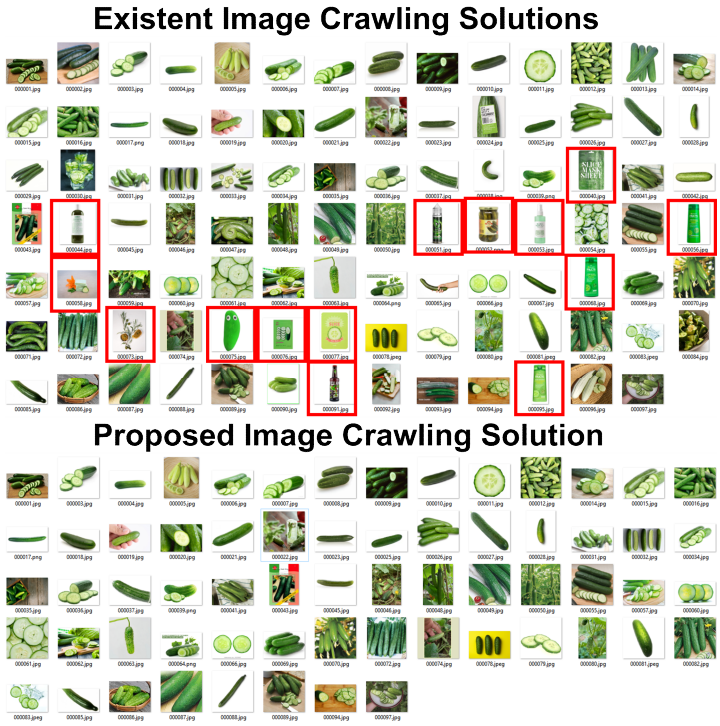


Fig. 5.14. Summarized view of comparison between existent and the proposed image crawling solution. The pictures marked with a red rectangle are some examples of "dirty" images found in existent solutions. By comparison, the proposed image crawling feature assisted by DL inference contains only clean images.

It can be easily observed how the existent solutions provide images that don't represent an actual cucumber, but products (e.g. shampoos) that are made out of it. After automatically cleaning these images with a confidence rate of 50% with the proposed feature, only 64 clean images remained in the folder.

For the experiments seen in Table 31, we tested the speed time of the proposed built-in image deduplication feature that uses the Imagededup python package [137]. We run these experiments on finding only exact duplicates on the same number of images with a maximum distance threshold of 10 for all four hashing methods. As can be seen, the average speed is about 16 seconds for finding duplicates in a folder containing 1.226 images, with Difference Hashing being the fastest hashing method from all four.

Table 31. Speed Results for the 4 hashing methods of the proposed Image Deduplication feature.

Nr. of Images	Hashing Method	Speed Time (sec)
1.226	Perceptual Hashing	16
	Difference Hashing	15
	Wavelet Hashing	17
	Average Hashing	16

For our experiments regarding the sorting of images with the proposed images sorter feature, we used both the MNIST as well as the ImageNet pre-trained models with a confidence rate of 50% and presented the results in Table 32.

Table 32. Speed Time for the proposed Images Sorting feature.

DL Model	Nr. of Classes	Nr. of Images	Undetermined Images	Speed Time (sec)
MNIST	10	70.000	69	307
ImageNet	1000	456.567	135.789	40.817
Custom [15]	34	2.380	34	223

Regarding MNIST experiments, we converted the MNIST dataset consisting of 70.000 images of 28×28 pixels to PNG format by using the script in [218] and mixed all these images in a folder. After that, we run our image sorter feature on them and succeeded to have only 0.09% of undetermined images, with a total speed time of around 6 minutes. Regarding ImageNet, we used the ImageNet Large Scale Visual Recognition Challenge 2013 (ILSVRC2013) dataset containing 456.567 images belonging to 1000 classes with a confidence rate of 50%. Here we successfully sorted all images in around 11 hours and 20 minutes, more exactly in 40.817 seconds, with 29.74% (135.789) undetermined images. Regarding the custom model, we used one of our previously trained DL models (ResNet-50) that can classify 34 animal classes [15] on a number of 2.380 images of 256×Ratio pixels (70 images for each of the 34 animal classes) with a confidence rate of 50%. Here we succeeded to have 1.42% undetermined images, with a total speed time of almost 4 minutes. The percentage of the undetermined images for all cases can be improved by modifying the confidence rate, but it is out of this work's scope to experiment with different confidence values.

The time that a DL prediction task takes depends on a few variables, mainly the processing power of the machine used to run the model, the framework used to call the inference of the model and the model itself. Since processing power keeps changing and varies greatly over different machines, and all the frameworks are optimized complexity wise and keep evolving, we find that among these three, the most important to measure is therefore the model itself used in the prediction.

Models vary greatly in their architecture, but all DL models can be mostly decomposed as a series of floating points operations (FLOPs). Because, generally, more FLOPs equal more processing needed and therefore more time spent in the whole operation, we measured the time complexity of the built-in ImageNet ('resnet50') and MNIST ('mnist') models in FLOPS and achieved 3.800 MFLOPS or 3.8 GFLOPS regarding ImageNet and 9 MFLOPS or 0.009 GFLOPS regarding MNIST.

For the experiments regarding the DL model training feature, because we want to evaluate the application on a real-world problem, we will attempt to show that this feature could be very useful for doctors or medical professionals in the aid of detecting diseases from imaging data (e.g. respiratory diseases detection with x-ray images). In order to prove this, we will attempt to automatically sort between the images of sick patients versus healthy patients regarding, firstly, pneumonia [64], and secondly, COVID-19 [65], all within our application and doing it only with the training feature that the application provides. For this, first, in order to classify between x-ray images of patients with pneumonia versus x-ray images of healthy patients, we made use of transfer learning and trained a 'resnet50' architecture for around 2 hours without data augmentation on pneumonia [64] dataset containing 5.200 train images by selecting 10 as the value for the number of training batches and 10 as the value for the size of batches (amount of images per batch) and achieved 98.54% train accuracy after 10 epochs. Secondly, in order to classify between x-ray images of patients with COVID-19 versus x-ray images of negative patients, we again made use of transfer learning and trained a 'resnet50' architecture for around 1 hour without data augmentation on the COVID-19 [65] dataset containing 107 train images by selecting the same values for the number and size of training batches as the pneumonia model mentioned above and achieved 100% train accuracy after 100 epochs.

For the experiments regarding the accuracy calculator feature, we used the two custom DL models trained earlier to classify x-ray images of patients with pneumonia versus x-ray images of healthy patients and between x-ray images of patients with COVID-19 versus x-ray images of negative patients, with 20 as the size of batches (20 images per batch). The evaluation took in both cases around 50 seconds with a test accuracy of 93.75% regarding the pneumonia model on 620 test images and 91% regarding the COVID-19 model on 11 test images, proving that the proposed Computer Vision application can easily be used by any medical personal with very basic computer knowledge in order to train and test a DL classification model for medical work purposes.

Regarding the simulated experiments with the proposed APC [20] calculator feature, we presented the results for different model test accuracy (%) and energy consumption (Wh) values in Table 33. We run all the experiments with 0.2 as the alpha value and with 1.0 as the beta value.

Table 33. Summarized Results of the proposed APC Calculator feature.

Energy Consumption (Wh)	DL Model Test Accuracy (%)	APC (%)
10	99.0	32.14
2	99.0	69.91
1	99.7	82.91
10	99.7	32.96
50	99.7	8.96

10	94.5	27.47
50	50.0	1.61
1	50.0	31.25
10	50.0	7.14
10	40.0	5.12
1	40.0	23.8
1	100	83.33

It is important to mention that our recommendation for a correct comparison between 2 DL models, is that it is always necessary that they are both tested with the same alpha and beta values. As can be seen in Table 33 where we experimented with random energy consumption and test accuracy values, our APC Calculator feature is evaluating the performance of a DL model by considering not only the accuracy but also the power consumption. Therefore, DL models that consume around 50 Wh (e.g. when running inference on a laptop) instead of 10 Wh (e.g. when running inference on a low-cost embedded platform such as the Nvidia Jetson TX2) [16], are penalized more severely by the APC metric.

Regarding the simulated experiments with the proposed APEC [20] calculator feature, we presented the results for different model test accuracy (%) and energy cost in Table 34. We run all the experiments with 0.2 as the alpha value and with 1.0 as the beta value.

Table 34. Summarized Results of the proposed APEC Calculator feature.

Energy Consumption [Wh]	**Power Cost [cents EUR]**	**DL Model Test Accuracy [%]**	**APEC [%]**	**APEC Green Energy [%]**
10	0.03050	99.0	98.37	99.0
2	0.0061	99.0	98.87	99.0
1	0.00305	99.7	99.63	99.7
10	0.03050	99.7	99.08	99.7
50	0.1525	99.7	96.71	99.7
10	0.03050	94.5	93.8	94.5
50	0.1525	50.0	45.8	50.0
1	0.00305	50.0	49.9	50.0
10	0.03050	50.0	49.1	50.0
10	0.03050	40.0	39.18	40.0
1	0.00305	40.0	39.91	40.0
1	0.00305	100	99.93	100

For simplicity, regarding electricity costs, we took Germany as an example. As mentioned earlier, according to "Strom Report" (based on Eurostat data) [207], German retail consumers paid 0.00305 Euro cents for a Wh of electricity in 2017. We used this value to calculate the cost of energy. As can be seen, the APEC metric favors lower power consumption and cost, favoring the use of green energy (free and clean energy).

Regarding the experiments with the proposed TTCAPC [20] calculator feature, we simulated a custom DL model on two platforms and presented the results in Table 35.

Table 35. TTCAPC with Accuracy delta = 0.1, Energy delta = 1, beta = 0.1, alpha = 0.1.

	Desktop PC	**Nvidia Jetson TX2**
Accuracy	97.92	
Energy Consumption (Wh)	50	10
Rounded Accuracy	97.95	
Rounded Energy Consumption	50.5	10.5
Closest APC	61.28	87.11
Train seconds	60	

As can be seen, even though the accuracy and training time is the same for both platforms, the TTCAPC feature favors the platform which has less power consumption.

Regarding the experiments with the proposed TTCAPEC [20] calculator feature, we simulated with the same DL model values used also in the experiments regarding the TTCAPC calculator earlier and presented the results in Table 36.

Table 36. TTCAPEC with Accuracy delta = 0.1, Energy delta = 1, beta = 0.1, alpha = 0.1.

	Desktop PC	**Nvidia Jetson TX2**
Accuracy	97.92	
Energy Cost (cents)	0.1525	0.0305
Rounded Energy Cost	0.1525	0.0305
Rounded Accuracy	97.95	
Closest APEC	51.46	82.96
Closest APEC Green (Solar) Powered	97.95	
Train seconds	60	

As can be also seen in this case, the TTCAPEC feature favors the lower power consumption of a system because it results in a lower cost. Additionally and more importantly, it favors DL-based systems that are powered by green energy, because they have 0 electricity costs and no negative impact on our environment.

6. AFFORDABLE FLYING PROBE-INSPIRED IN-CIRCUIT-TESTER FOR PRINTED CIRCUIT BOARDS EVALUATION WITH APPLICATION IN TEST ENGINEERING EDUCATION

Education is considered one of the most important factors that drive our society into new horizons, bringing new understandings of our reality and thus resulting in new and better technologies. Recent efforts in bringing affordable and equal access to education are seen also on the UN agenda, one example being the UN's Sustainable Development Goals [189]. Regarding test engineering education, a major issue found in many of the technical schools and universities around the globe is the huge amount of technical books available but without giving the students also a chance to have a hands-on experience with real parameter values of a PCB. Concerning FPTs, this situation is usually the result of expensive ICT versions found in the industry [12], which is the leading factor for the lack of proper testing equipment in engineering laboratories.

One of the main concepts that help students be familiarized with the important and delicate process of evaluating the performance of PCB testing measurements is called testability. Testability is the property of a PCB to enable the test engineer to easily define the electronic board checking procedure at the desired level. Generally, it is given by a) *mechanical parameters* (the shape of the populated PCB and the test adapter design); and b) *electrical parameters* (access to the test samples, test methods and electrical isolation possibilities of surrounding components). ICT enables a very fast testing procedure where access can be made simultaneously on all test fields. However, this type of testing is demanding and requires a suitable electronic board design with test pads.

Despite the fact that FPTs are able to perform high-speed testing with flawless accuracy for each tested board, incorporating the latest technologies such as Boundary Scan, ICT and even Optical Inspection [219], these features require additional expensive hardware such as optical sensors and encoders as well as forcing the test engineer to reconstruct the fixture every time a new board under test is used, resulting in high costs and time consumption.

Considering these aspects, in this work, we propose an efficient FPICT that has educational purposes and which is based on an Arduino MEGA2560 microcontroller, three ULN2003A motor drivers with their associated 28BYJ-48 Steppers as well as three Mechanical Endstop Limit Switches (MELS). Our education-oriented FPICT is designed in a way to leverage the difficulties students have when trying to learn new concepts in the domain of test engineering.

6.1. Hardware Components of the Proposed FPICT

Our FPICT is summarized in Fig.6.1 and resembles the characteristics of a Flying Probe design and the operation features of a CMM. CMMs typically specify the

position of a probe from a reference position in a three-dimensional Cartesian coordinate system in terms of its displacement. Inspired by its simple yet efficient design, we constructed our own tridimensional platform (axis X, Y, and Z) motorized by Steppers which are controlled directly from an Arduino Mega 2560 mainboard. In this section, we provide a detailed overview of our proposed FPICT system, which can be divided into mechanical and electrical components.

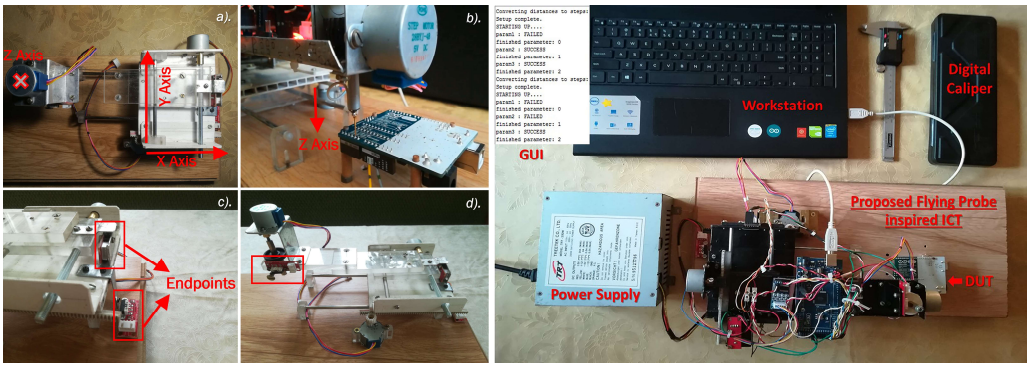


Fig. 6.1. Left: FPICT Mechanical Structure with Axis Array (a, b) and MELS Placement (c, d). Right: Complete experimental setup for the proposed FPICT.

6.1.1. Mechanical Components

The device was fixed on a parquet board that was cut according to the following sizes: Length (L) = 430 mm; Width (l) = 200 mm, resulting in a total space of A = L × l = 430 × 200 = 86.000 mm^2 allocated for testing. The main platform, from where the three axes (X, Y, and Z) gather their reference points was mounted on two 190 mm long metal rods. According to Fig.6.1, our initial variant is built around three main axes that we will describe in the following lines:

- ***X-Axis*** – is located at the inferior part of the main platform with the Stepper motor being fixed on the parquet board a). The translation from one direction to another is realized via a smaller cogwheel that interacts with a 130 mm long rack. This allows complete translation freedom equal to the length of the entire rack until it reaches the first MELS illustrated in the bottom part of scenario c).
- ***Y-Axis*** – is mounted on top of the main platform and contains two metal rods (both 120 mm long), one of them being a screw that interacts with the Stepper motor on the other end, as seen in scenario a). Since the stepper motor rotates the screw in two distinct directions, the secondary platform (formed of a thick Plexiglas) will be translated according to the straight drill rule, acting like a nut on the screw. However, due to mechanical constraints, the translation freedom of the Y-axis was reduced to 50 mm until it reaches the second MELS presented in the top part of scenario c).
- ***Z-Axis*** – viewed as the most complex to build of the three-axis, adopts a two-layered Plexiglas structure and is mounted directly on the previous axis system. Composed of a 60 mm hexagonal nut and combined with a screw, it functions exactly on the same principle as the previously described Y-axis. The translation limit is set to ~ 20 mm, which is sufficient for the probe (nail) to touch the contact of the PCB.

6.1.2. Electrical Components

Each of the three Cartesian axes (X, Y, Z) described above is controlled by electrical equipment consisting of a main Arduino MEGA2560 microcontroller, 3 ULN2003A motor drivers, 3 28BYJ-48 Steppers and their MELS. We will detail each of the individual parts and further argue why the chosen setup is effective from the power consumption and cost perspectives.

Arduino Mega is an ATmega2560-based microcontroller board with 54 digital input/output pins (14 of which can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), 16 MHz crystal oscillator, USB connection, energy jack, ICSP header, and a reset button. With all the listed characteristics and notably because of the large number of digital pins, it provides an optimal solution for complex projects. The board can operate on an external energy supply of 6V to 20V. However, if supplied with less than 7V, the 5V connector will provide less voltage while the board could become volatile. When using more than 12V, the voltage controller can overheat and damage the board. Therefore, the suggested range is between 7V and 12V. According to around 8 hours of measurements at the USB plug with a flowing current of 52-54 mA, the average usage of the Arduino Mega 2560 is rated at 0,27W. For our project, we use a total number of 15 digital inputs/outputs to assign *pins 22-25* to the X-axis, *pins 26-29* to the Y-axis, *pins 30-33* to the Z-axis and *pins 46-48* to receive feedback from the MELS.

ULN2003 is part of the well-known ULN200X IC series and represents a relay driver IC made up of an array of Darlington transistors. It consists of seven open pairs of Darlington collectors with prevalent emitters. In addition, ULN2003A has the ability to simultaneously handle seven different relays. A single pair of Darlington is made up of two bipolar transistors and operates between 500mA and 600mA current. ULN2003 operates on 5V using TTL and CMOS technologies. Its pin configuration provides an accessible design so that the input pins are on the left side of the IC while the output pins are placed on the opposite side. The chosen IC has a broad variety of applications being frequently used as relay drivers to drive different load types. ULN2003A can also be used to drive various engines (DC motors, Steppers), logic buffers, lamp, and line drivers LED displays and motor driver circuits.

The chosen 28BYHYJ-48 Steppers are lightweight engines that are generally incorporated into DVD drives, movement cameras, and other devices that require high accuracy for a set of specific functions. The engine has a 4-coil unipolar mount and each coil is rated at +5V, making it extremely simple to manage them with any traditional microcontroller. These motor types have a 5.625 ° /64 step angle, which means that the motor will have to take 64 steps to complete one rotation and cover a 5.625 ° level for each step. Usually, these stepper motors consume high current, thus requiring an IC driver like the ULN2003 that was listed earlier. As can be seen in Fig.6.1, the engine of a stepper motor has four coils: one end of the coil is tied to + 5V (Red) and the other ends (Orange, Pink, Yellow, and Blue) are grouped together and linked to the header connector of the ULN2003A. The operational voltage is rated at 5V and hence it provides sufficient torque for moving the testing probe around the DUT. Only when the coils are energized (grounded) in a logical sequence the stepper motor is able to rotate in a certain direction. The logical sequence can be implemented either by using a microcontroller or a dedicated digital circuit. These types of stepper motors can be used in a variety of applications

such as CNC machines, security cameras, DVD players, car side mirror tilt systems and precise control machines such as our FPICT.

A limit switch is known as an electromechanical element that contains an actuator that is mechanically connected to a set of contacts (terminals). During the test of a PCB, whenever the actuator interacts with a foreign object (e.g. metal object obstacle), the MELS device is triggered and starts sending a signal to the contacts (terminals) to decide if the electrical connection should be on or off (therefore, limit switches are practical and low-cost devices that allow the user to activate or deactivate a certain process when the MELS was stimulated by an external factor with the help of a lever-type of switch). The lever switch is wired up so that it can pull the signal to LOW when it is activated. The micro board also has an LED that will light up when the switch is activated. In our case, the MELS is used to detect endpoints for all three axes of the FPICT. Usually, MELS can be used together with RepRap Arduino Mega Pololu Shield (RAMPS) boards but can also be combined with other microcontrollers such as the AtMega2560. The maximum working voltage is rated at 200V while the current can go up to as much as 2A. The MELS serves as a reference point from which the FPICT setup will start inspecting the DUT.

6.2. Sensorless-Based Test Point Tracking

The proposed FPICT process is divided into several stages that are correlated with Fig.6.2.

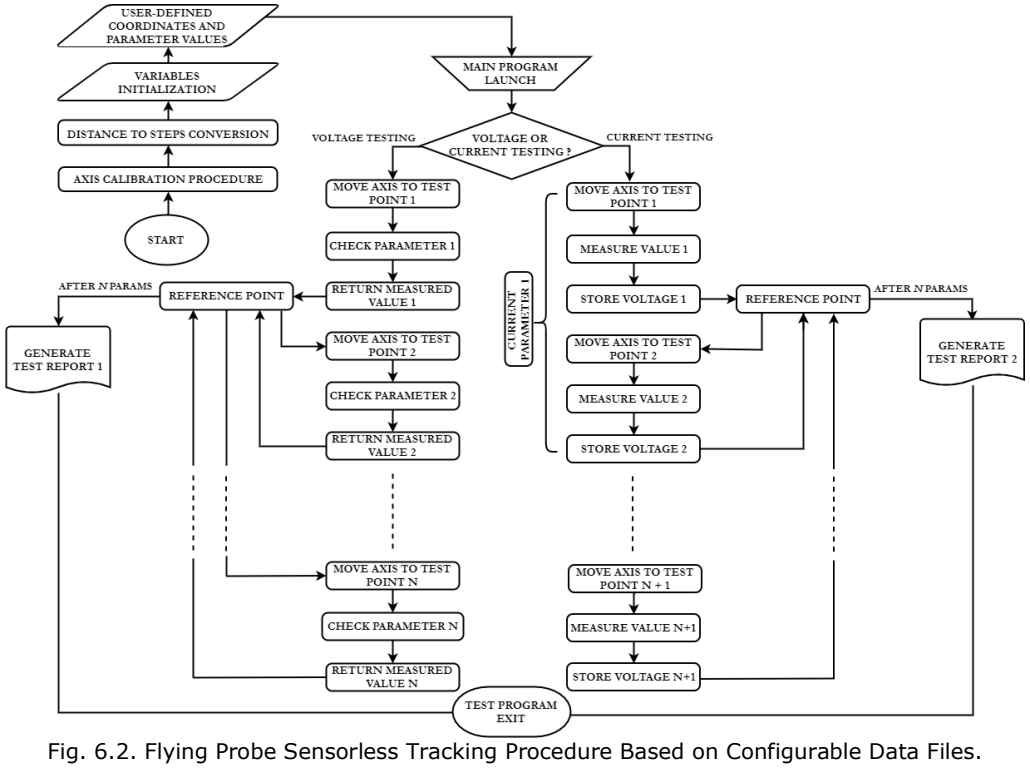


Fig. 6.2. Flying Probe Sensorless Tracking Procedure Based on Configurable Data Files.

It is worth mentioning that the FPICT procedure undergoes a number of preliminary preparations known as modules (Axis Calibration, Distance to Steps Conversion, Variables Initialization, Coordinates and Parameters Definition, and Main Program Launch) before launching the main test program.

The Axis Calibration module sets the initial coordinates (0,0,0) of the Cartesian landmark that the device sketches spatially. Any mechanical imperfection of the constructed X, Y, Z-axes may influence the accuracy of the measurements made according to the values in the Coordinates and Parameters Definition module.

Regarding the Distance to Steps Conversion module, the conversion is made in a unique way for each of the variables declared for testing the voltage and current. It is important to note that the voltage measurement procedure differs from the one performed for the current, in that the mobile probe is connected to a single analog input A0 of the Arduino Mega board and the voltage value is obtained by measuring the test point which is always connected to the ground point. In this case, the test program requires only three local variables denoted by *Dist_X_mm*, *Dist_Y_mm* and *Dist_Z_mm,* representing the distance of each axis to the origin of the Cartesian system. In the case of current measurement, at least two test points from where the values are collected through the main program are required. As a result, a set of variables noted with *Dist_X1_mm*, *Dist_X2_mm*, *Dist_Y1_mm*, *Dist_Y2_mm,* and *Dist_Z_mm* are declared, where the pair (X1, Y1) designates the first coordinates target point, the pair (X2, Y2) refers to the second test point and Z is the same because the height from which the probe drops stays always constant throughout the test. Because we use Stepper motors to move the axes, the test program will have to translate the values of distances in steps according to the following formula (6.1):

$$DistXStep = DistXmm \times StepsPerMM \tag{6.1}$$

where *DistXStep* represents the number of steps obtained by multiplying the Cartesian distance *DistXmm* by the value of the distance in millimeters executed by a single step of the *StepsPerMM* engine (variables to which we will return to with further explanation in this section). Additionally, because the Arduino Mega board has a built-in ADC, implicit conversion of the parameters entered by the user in the same module is performed. The ADC on the Arduino is a 10-bit ADC, which means that it has the ability to detect 1.024 (2^{10}) discrete analog levels. Some microcontrollers have 8-bit ADCs (2^8 = 256 discrete levels), and others have 16-bit ADCs (2^{16} = 65.536 discrete levels). Thus, the converter generates a ratiometric value because the ADC assumes that 5V is 1.023 discrete levels and any value less than 5V (1.023 discrete levels) will be a ratio between 5V and 1.023 discrete levels. The result of the ADC in our case will be retained in a variable that appears in the relation (6.2):

$$CountExpectedVoltage = \frac{1023 \times ExpectedVoltage}{5} \tag{6.2}$$

where *CountExpectedVoltage* will count the measurements with the expected results from the tests performed.

Regarding Variables Initialization, this module covers two types of variables used: global and local. Due to the size and complexity of our code, we will list the

most important variables. The global variables can be called anywhere in the code and allow a flexible modification by the domain expert:

- *STEPS_PER_REVOLUTION* – shows the total amount of motor steps for one complete rotation (360 degrees). According to the Stepper user manual, the recommended value is 2048
- *MOTOR_SPEED1* – the value of the speed set for the first Stepper motor corresponding to the X-axis, with a default value of 15
- *MOTOR_SPEED2* – the value of the speed set for the second Stepper motor corresponding to the Y-axis, with a default value of 15
- *MOTOR_SPEED3* – the value of the speed set for the third Stepper motor corresponding to the Z-axis, with a default value of 13
- *MM_PER_STEP* – the value in millimeters associated with a step executed by the Stepper motor. In order to determine this value, a trial and error experiment was used which resulted in 0.10 mm / step
- *TOTAL_CURRENT_PARAMS* – the total number of parameters associated with the current measurement, with a set value of 4
- *RES_CUR_MEASURE (RESCURMEASURE)* – the resistance expected in Ohms between the two test points (X1, Y1) and (X2, Y2) for measuring the current
- *STEPS_PER_MM – is a set value that approximates the number of steps performed per millimeter. The value found (determined by trial and error) was 100*

Local variables are the elements that underlie data processing such as the considered distances and the parameters targeted for verification. *Dist_X1_mm*, *Dist_Y1_mm*, *Dist_X2_mm*, *Dist_Y2_mm*, *Dist_Z_mm* are the distances from the reference point to the two test points associated with the measurement of the current, respectively *Dist_X1_step*, *Dist_Y1_step*, *Dist_X2_step*, *Dist_Y2_step*, the steps corresponding to the aforementioned distances. Variables *volExpectedL* and *volExpectedH* are float-type variables for setting a sensitive threshold for voltage measurements, while *CountExpectedL* and *CountExpectedH* will monitor the number of parameters outside the range of allowed values for each test. Additionally, *curExpectedL* and *curExpectedH* achieve the same minimum and maximum threshold but only for current measurement.

Regarding Coordinates and Parameters Definition, the set of variables and parameters stated above will continue to be composed in the form of configurable structures that are visible to the average user. For an easier understanding, all the data that is needed to test the voltage and current were organized in cells, as can be seen in Table 37.

Table 37. Example of the configuration structure for measuring voltage and current on the selected test points.

Parameter Number	**Cartesian Coordinates (mm)**				
	Current				
	Voltage Measuring				***Voltage Measuring***
	DIST X1	***DIST Y1***	***DIST X2***	***DIST Y2***	***DIST Z***
1	16.80	11.40	...	...	14
2	19.99	8	...	...	14
3	19.96	8	...	...	14
4	37.03	18.80	...	...	14
5	38	18.95	...	...	14

	Parameter Description						
	Assigned Pins (for Voltage)	***Targeted Microchip***		***Low [V]***	***High [V]***	***Low [mA]***	***High [mA]***
		(for Voltage)	***(for Current)***				
1	7	AtMega328	...	4.5	5	...	...
2	20	AtMega328	...	4.5	5	...	...
3	21	AtMega328	...	4.5	5	...	...
4	31	AtMega16u2	...	4	5	...	...
5	32	AtMega16u2	...	4	5	...	...

The verification can be performed for various areas on the test board, which in our case is an Arduino UNO with two microchips of interest: Atmega328 and Atmega16u2. Both micro-devices must be powered at a voltage not exceeding 5V and generally not falling below 4V. Each deviation from the range of values considered critical (less than 4V and greater than 5V) according to the specialized catalog, can lead to a decrease of the performance (if it falls below 4V) or even to the failure of the board (if it exceeds 5V value). Additionally, the user will be able to enter the data needed to test the current parameters in the cells where the free space is represented by dots.

Regarding the Main Program Launch, this module can be started directly from the Serial Monitor window in the Arduino IDE Suite Interface. Following the configuration of the preliminary values (Cartesian distances as well as the values of the voltage and current parameters), the main algorithm will move the calibrated probe from the considered reference point to the first test point. An example of a compact execution line for a voltage parameter during testing (which is very efficient also regarding memory) is presented in Fig.6.3.

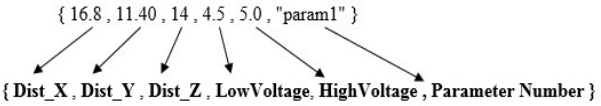


Fig. 6.3. The configuration file structure of an execution line for a voltage parameter.

Once the test point is reached, the probe will collect the voltage value through the analog input A0 which is converted according to formula (6.2) into the binary system. Then, it is compared with the *LowVoltage* minimum value and the *HighVoltage* maximum value, in order to decide if the parameter is within the imposed limits (between 4 and 5V). Once the first value is measured, the moving probe will move to the reference point to continue the measurement for the next number of nodes. The same procedure is repeated for all the test points in Table 37, up to the last value, in order to generate a complete report with the situation of each node (test point) separately.

$$curmA = \frac{voltage2 - voltage1}{RESCURMEASURE} \times 1000 \tag{6.3}$$

According to formula (6.3) presented above, the mobile probe will have to collect the values of the voltages from the ends of the resistor and to divide their difference by the global variable declared earlier regarding the Variables Initialization module, namely RES_CUR_MEASURE (RESCURMEASURE). The

determined value of the current will be compared with the two values in the tolerance field, in order to check if it is within the limits imposed by the user.

6.3. Experimental Setup and Results

The most fundamental resources needed when designing an effective FPT are probe positioning, measurements, test tools, development tools and time. These resources are taken into account also by our final prototype seen on the right side of Fig.6.1 and were obtained by analyzing the experimental dataset summarized in Table 38.

Table 38. Single Point, Multiple Point, and Measurement Testing Results.

Test Type	**No. of Test Samples (Parameters)**	**Precision Testing (%)**	**Average time per test cycle [s]**	**System Power Consumption [W]**	
				Idle	**Active**
Single Point Testing	500	100	10.35	0.360	3.895
Multiple Point Testing	500	91.40	62.69		3.850
Measurement Testing	1000	**95.70**	1.53		4.027

The entire input dataset was determined by consulting the specialist catalog of the Arduino board for the optimal operating sizes (voltage and current). The Cartesian coordinates were measured using a digital caliper with the precision of hundreds of millimeters from the chosen reference point. In terms of power consumption, measurements were made with the multimeter, both with the laptop connected via USB to the Arduino Mega as well as just with the proposed FPICT alone (without a laptop). In the idle state, with the laptop connected, our FPICT device consumed 72mA at a 5V power supply voltage, resulting in power consumption of 0.36 W for all test scenarios. In the active state, with the X, Y, Z-axes in motion and the laptop connected, the current consumption increased to 686mA at the same supply voltage, thus achieving a power consumption of 3.43W. The average values for all test types are shown in Table 38.

The success of testing a parameter (voltage or current) is in principle given by the positioning accuracy of the mobile pin and the ability of the FPT probe to correctly read the value of the voltage or current on the already reached pin. As can be seen from Table 38, the positioning accuracy when checking a single test point shows that our FPICT is capable of reaching maximum accuracy (100%), whereas, in the case of checking several nodes on the test board, the accuracy decreases in some proportion (91.40%), either from mechanical impediments that need to be revised or from the inability of the probe to take the voltage value correctly from the tested pin. The average accuracy obtained for all measurements was 95.70%, a relatively good percentage for a device built from low-cost components. The average time per test cycle (s) noted in Table 38 was determined for single test points and the entire test set composed of 5 parameters. Thus, 10.35 seconds were obtained for a fixed test point, with coordinates set from repeated tests and an average of about 1 minute for all 5 test points, each with different coordinates and

distances from the reference point. The measurement time of the probe was estimated to be around one and a half seconds (1.53s).

7. TECHNOLOGICAL SOLUTIONS FOR THROUGHPUT IMPROVEMENT OF A SECURE HASH ALGORITHM-256 ENGINE

In this chapter, we will present our work published in [23] which describes a set of techniques for improving the performance of a SHA-256 hardware implementation. The proposed solution reduces the latency incurred for updating the intermediate hash values and relies on using combinational tree structures of CSAs interconnected in a Wallace tree manner for multi-operand addition. Furthermore, the chapter investigates the throughout improvement provided by a combined implementation of architecture's binary adders with the round functions used by the hash computation process. The proposed acceleration techniques can be adapted to the other members of the SHA-2 family of algorithms. The architecture represents a case study for hardware optimization based on different combinational structures for binary addition and the effect of the carry propagate layer on the overall performance. The synthesis results of the proposed designs are provided as support for the performance analysis presented in this work.

7.1. Throughput Improvement Solutions for SHA-256

The hardware implementations of the SHA-2 hash functions, in general, and of the SHA-256 algorithm, in particular, are benefiting of higher throughput and of a comparatively more secure computing platform when compared to their software counterparts. The literature contains references of dedicated hardware architectures for offloading the computational-intensive operations of hash calculation in order to obtain higher throughput [220], [221]. The effect of hash computation over the system's throughput is, in particular, relevant for the case of servers offering services based around IPSec and SSL/TLS, for which the hash calculation latencies become a limiting factor in servicing all received requests. In this context, constructing a customized hashing accelerator relieves the CPU in such a server from computing hashes, allowing it to attend other tasks and thus, optimizing the clock cycles usage.

Another reason for implementing the hash computation in hardware relates to security. Software implementations of a hash function, running on a general-purpose processor, oftentimes lack the physical protection found in hardware implementations due to the relative ease with which an attacker is capable of inspecting and even modify the software implementation. Moreover, the intrusion can even be set up concurrently with the system's utilization, while the cryptographic application is operational, this being achieved by using debugging software. In the same class of attacks against software implementations of cryptographic functions can be mentioned the timing attacks and the cache attacks customized for breaking the security of other cryptographic services as well [222].

The SHA-256 architecture constructed in this chapter is an iterative design, instantiating one round of the message scheduler and one round of the data

compression stage in hardware. The reason for including a single iteration in hardware was to provide a rigorous comparison framework for other acceleration techniques, presented in the literature.

The message padding and parsing steps can be implemented either in hardware or in software and are not considered. The basic design is depicted in Fig. 7.1. The first architectural optimization that can be applied to a hardware realization of SHA-256 and the first design decision in our proposed architecture targets the message scheduler. Similar to other hardware designs for SHA-256 found in the literature, the architecture proposed in this chapter stores, at any given moment, only 16 words of the message scheduler, instead of providing storage space for all 64 words. This reduction in storage requirements can be realized because the relation used for calculating the next word of the message scheduler makes use of 4 words, all 4 being calculated no later than 16 iterations ahead [66].

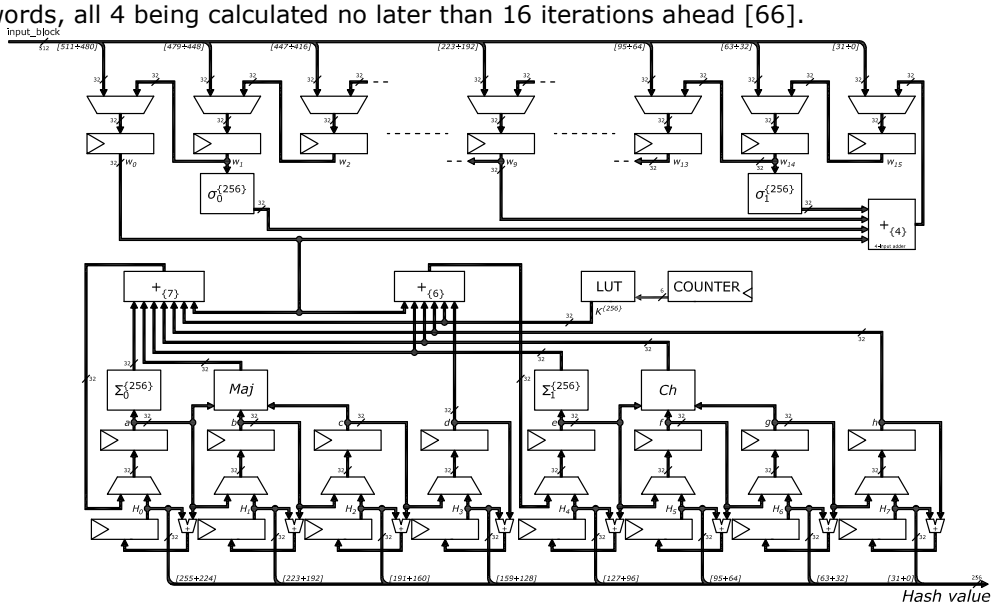


Fig. 7.1. The basic architecture for SHA-256.

Moreover, because the first 16 words of the message scheduler are the very 16 words of the 512-bit input block, the hardware design for the message scheduler stores the input block into the 16 words, at the beginning of a block processing. After delivering the word W_0, the message scheduler calculates the next one and shifts the least significant 15 words to the left in order to append the newly calculated value.

The message scheduler consists of the modules in the top part of Fig.7.1 in which, for clarity, some of the 16 registers storing the message scheduler words were omitted. In Fig.7.1, the "*COUNTER*" unit is a 6-bit iteration counter keeping track of the current algorithm's round. By means of the multiplexing layer connected on the inputs of the message scheduler registers, either the initial 512-bit block or the shifted content is delivered. The word generated by the scheduler each iteration is stored in the least significant position and its calculation delimits the critical path for the message scheduler. The middle register layer of Fig.7.1 is made up of the 8 registers storing the working variables, each having the associated variable symbolized next to its output.

One advantage of the SHA-256's hardware realizations over software is the straightforward use of concurrent processing. To this avail, the message scheduler and the data compression stages can be run in parallel because they both are iterated 64 times. The first iteration of the data compression stage makes use of the first word of the message scheduler. By the time the scheduler delivers the first word it has already stored the first 16 words and, concurrently with the first iteration of the data compression loop, the scheduler starts calculating the 17th one. It is for this reason that both register sets, storing the message scheduler words and the 8 working variables, are controlled by the same loading signal.

The current word of the message scheduler to be used by the data compression engine is in the most significant position, as depicted in Fig.7.1 and because of the content shifting of the scheduler's data words, by the time the last data compression iteration is executed, the scheduler will have generated 15 additional words, besides the last one, W_{63}. These 15 words will not be used by the hash computation. The computation of these words can be avoided; however, it would require additional hardware investment in selecting the current data to be delivered to the hash computation stage. Additional investment would be needed for disabling the load signal for the message scheduling unit. For an increased throughput architecture, the decision to have a common control line for loading the registers of the message scheduler and those storing the working variables, and being able to directly deliver word W_0 to the data compression unit is preferred, to the expense of computing 15 unused words.

The content of the 8 registers storing the working variables is updated either with the current hash value or with the new values generated by the data compression round. The multiplexors selecting the input of the working variable registers use the same selection line as those selecting the input for the scheduler registers because the two sets of registers are initially updated from alternate sources and they are initially updated at the same moment, at the beginning of the 64 iterations. The critical path of the hash engine in Fig.7.1 contains the components used for calculating the next value for working variable *a* and includes the modules evaluating the 4 round functions ($\Sigma_0^{\{256\}}$, $\Sigma_1^{\{256\}}$, *Ch* and *Maj*), which operate in parallel, followed by the 7-operand, modulo 2^{32} adder calculating the next value for *a* and followed by the multiplexer delivering the new value to the corresponding working variable register.

The final storage layer is made up of the 8 registers keeping the current hash value. The 8 registers are updated at the end of the 64 data compression iterations by adding the value of the working variables to their current content. Eight modulo 2^{32} adders are required for this operation. In addition, the hash value update demands one supplementary clock cycle, thus directly affecting the latency and the throughput of the SHA-256 unit.

The first throughput acceleration technique proposed in this chapter for the SHA-256 architecture reduces the number of cycles used for processing a 512-bit block by eliminating the previously mentioned hash update operation, performed at the end of data compression's loop. To achieve this, the last of the 64 iterations will have to update not only the working variables but the hash registers as well. In consequence, the final round of the data compression phase will have to additionally include the current hash value among the operands added together, in order to be able to generate the next hash value.

In addition to this computation strategy, the current hash value does not need to be loaded into the working variable registers at the start of a new block's processing. This is because the working variables were already updated with the same data as the hash registers in the last round of data compression's stage. However, this observation does not facilitate a further reduction of clock cycles because, typically, the loading of the new 512-bit block is performed concurrently with the update of the hash registers.

The manner in which the proposed architecture is updating the working variable registers and the hash registers is depicted in Fig.7.2, in which only the modules pertaining to the data compression phase were included. The message scheduler remains unmodified as in Fig.7.1, together with the control signals commanding the scheduler's operation.

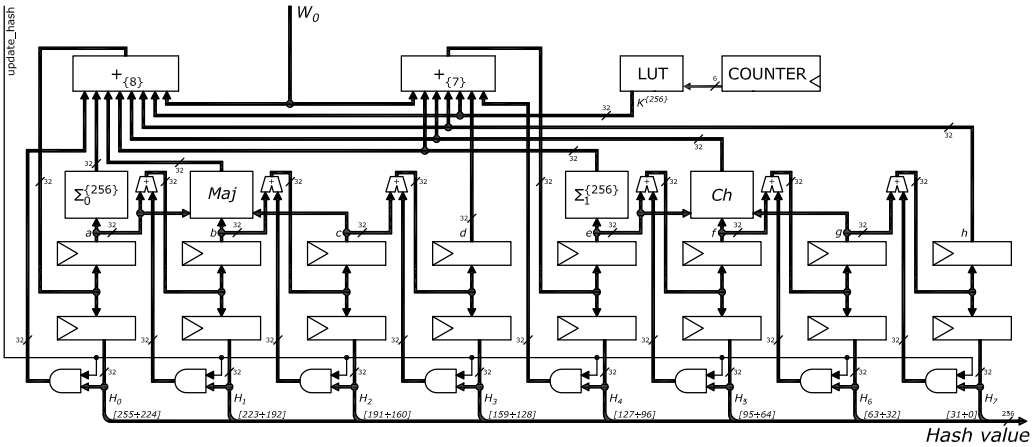


Fig. 7.2. Proposed architecture for SHA-256 hash calculation.

Because the addition of the current hash value is performed only in the last iteration, the content of the hash registers is enabled only once by means of an AND-gate layer, commanded by the gating signal *update_hash*, in Fig.7.2. The gating signal is, in fact, the signal enabling loading of the new hash value into the hash registers after the data compression finished. The critical path for this new design follows the signals' propagation through the 4 round functions, followed by the 8-operand modulo 2^{32} adder. The multi-operand adder architecture used throughout this article is a CSA based tree structure, organized in a Wallace manner [57] that generates the non-redundant sum by a final CPA.

Because the addition of the operands is performed, for SHA-256, modulo 2^{32}, all CSAs are on 32 bits and, since the carry vector is one position more significant than the sum vector, the most significant carry generated by a CSA level is omitted. As a result, the CPA module is also on 32 bits. However, due to the manner in which latency is propagated through the Full Adder Cells (FACs) of a CSA, the critical path of a CSA tree structure is considerably reduced only if a fast adder is used for CPA, avoiding serial propagation of the carry [223]. Because of this, a fast 32-bit Kogge-Stone [224] adder is used for the CPA level throughout this chapter. The replacement of the 7-operand adder in Fig.7.1 by the 8-operand addition structure from Fig.7.2 affects the adder's critical path only marginally, as the experimental results reveal.

The technological factor has an important influence on the synthesis result. With respect to this aspect, another latency incurring element in the architecture of Fig. 7.2 is the fan-out of the gating signal controlling the AND-gate layer. The fan-out of the *update_hash* signal is large due to its controlling the hash registers' load line and the gating layer. The large fan-out has an adverse effect on the critical path, in that, depending on the standard cell library used for synthesis, the *update_hash* signal's distribution tree can have a delay larger than the latency of the structures operating the 4 functions and the 8-operand adder together.

Further investigations evaluated alternative multi-operand addition structures and the effect of ordering addition's operands over the critical path. More precisely, in order to further improve the performance of the hash core, we investigated the possibility to implement the $\Sigma_0^{\{256\}}$, $\Sigma_1^{\{256\}}$, *Ch* and *Maj* functions in a combined manner with the multi-operand adders present in the architecture. The 4 functions are part of the critical path and reducing their number of logic levels improves the overall latency.

Fusing any of the 4 functions with the 32-bit CSAs is limited in outcome by the relatively simple structure of the FACs. However, considering the delay balancing technique introduced in [144], the fused implementation of the 4 functions with the CPA has a larger potential for latency improvement. Fig.7.3 illustrates a detail of the fused design combining variable *a*, in redundant form, with hash word H_0^i, also in redundant form, with the hash word H_1^i, the working variable *b* and with the working variable *c*. The fused module calculates $\Sigma_0^{\{256\}}$ and *Maj* functions together with the next working variable *b*.

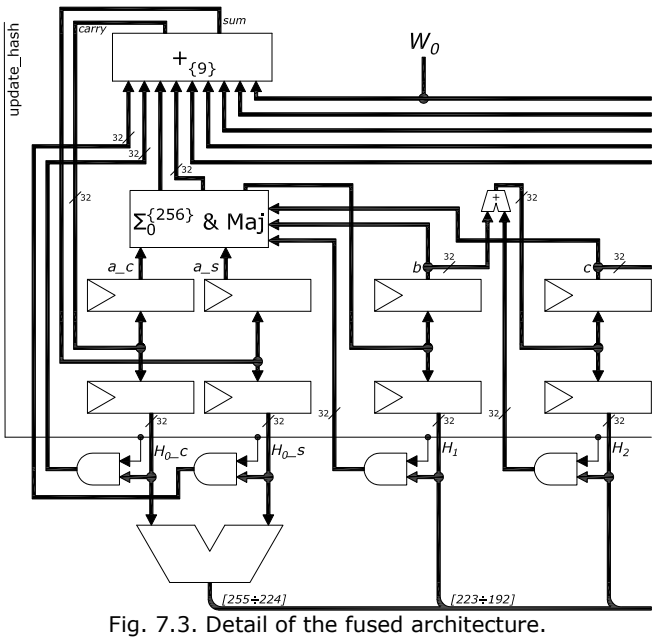


Fig. 7.3. Detail of the fused architecture.

Regarding the delay balancing implementation used in our proposal, it is applied for both working variables *a* and *e*, whereas Fig.7.3 illustrates this approach

only for *a*. The method requires doubling the registers storing the two variables into sum-carry pairs of registers. The corresponding hash value registers, H_0^i and H_4^i, need to be doubled as well in order to preserve the critical path reduction (otherwise, if the hash registers would not be doubled, dedicated CPAs would be needed for generating the non-redundant hash words which would defeat the purpose of delay balancing). As a result of using two registers for storing the redundant form of the hash word H_0^i, the multi-operand adder with 8 inputs from Fig.7.2 is replaced by a 9-operand redundant adder which will store the sum in the register pair (*a_c*, *a_s*), for *a*'s carry and sum vectors. Another consequence of storing H_0^i in redundant form is that it requires a CPA for calculating the word of the final digest, adder that is visible in Fig.7.3 as well.

Concerning the fused implementation, the *sigma* functions, $\Sigma_0^{\{256\}}$ and $\Sigma_1^{\{256\}}$, are composed of three rotations of the input words, each, followed by Exclusive-OR (EXOR) on the rotated vectors. The functions inputs are generated by the engine's CPAs and for achieving a combined implementation, the functions evaluation will be embedded in the CPAs. More precisely, for a Kogge-Stone adder, the carry bits are calculated by a tree structure and they are EXOR-ed with the half sum bits (EXOR result of input operands). Because the *sigma* functions employ the same EXOR operator, their results can be speeded up by balancing the EXOR tree. This solution removes one level of EXOR gates compared to the unfused approach. The *Maj* and *Ch* functions can be expressed in terms of the faster AND/OR primitives as: *Maj*(*a*,*b*,*c*)=*a* AND (*b* OR *c*) OR (*b* AND *c*) and *Ch*(*e*,*f*,*g*)=(*e* AND *f*) OR ($\overline{e}$ AND *g*). The negation of *e* is constructed by negating the half sum bits and EXOR-ing the result with the carry bits inside the Kogge-Stone CPA, for a rapid output generation. A final approach towards throughput improvement is to reorder the multi-operand adders' inputs in order to connect the signals generated at a later time on positions affected by smaller latencies. By means of synthesis results, the latest operands are identified and are correspondingly connected to the minimum delay inputs of the adder tree.

7.2. Experimental Results

All presented designs were modeled in Verilog and synthesized using the Design Compiler with the IIT Standard Cell Library for TSMC 0.18µm [225]. The synthesis results are presented in Table 39 for the basic architecture, together with the proposed and fused designs. The basic architecture uses as little of the throughput enhancing techniques as possible to prove that the analyzed techniques can be applied in conjunction with most of the other existing acceleration methods.

Table 39. SHA-256 Architectures Comparison.

Architecture	Max Frequency [MHz]	Area [µm²]	Throughput (Mbps)
Basic	326.80	480625	1287.08
Proposed	357.14	529690	1428.57
Fused	380.23	562704	1520.91

The basic architecture of Fig. 7.1 uses a multi-operand CSA adder tree, a Ripple Carry Adder for the final CPA layer and process a 512-bit block in 65 cycles. Although it requires the smallest area, the basic design is also the slowest one. The proposed design refers to the architecture described in Fig. 7.2. It makes use of the hash registers gating technique, uses 8-operand and 7-operand adders for generating variables *a* and *e*, respectively and requires only 64 iterations for a 512-bit block processing. The 10% performance improvement is obtained at the expense of increased area. Finally, the fused architecture takes advantage of the combined implementation of the 4 SHA-256 round functions with the existing CPAs yielding a performance increase of about 18% at the expense of a larger area overhead.

8. CONCLUSIONS AND FUTURE WORK

The AI revolution is happening and thanks to DL, better applications that surpass the human accuracy level are implemented day by day in many domains and industries. However, DL not only that it requires a high power consumption that results in high costs, but it also contributes to the carbon footprint, having a negative impact on our environment. To address these problems, we present solutions for powering and evaluating DL-based systems based on green energy.

This Ph.D. thesis focuses on eliminating the energy cost by not only building a dual-axis solar tracker in order to power a real-time DL-based system using solar energy but also on proposing environmentally-friendly metrics regarding inference and training.

Chapter 3 presents different DL applications that solve different problems related to fraud and security.

The first application created is, to the best of our knowledge, the first application that successfully detects receipt fraud, a common problem that occurs in many hypermarkets/supermarkets around the world. We implemented an OCR algorithm composed of image processing techniques and two CNN models into a smartphone application that helps the customers (in case of a wrong product price on the bill) as well as the supermarket employees (in case of different prices for products at the shelf compared to the prices stored in their computer system) to have (customer pays the correct price) and offer (ability to immediately update the correct price at the shelf) a better shopping experience. Experimental results show that the proposed CNN models have 99.96% test accuracy in identifying product prices and 99.35% test accuracy in identifying receipt prices, proving that our application can be used successfully in discovering wrong prices between a price tag belonging to a product seen at the shelf and the price paid after receiving the bill from the cashier. As future work, we plan to improve our CNN models to recognize also prices with multiple font types, from different hypermarkets/supermarkets that may use a different position of the two decimals in their prices. Additionally, to be able to detect characters representing the product name or identify the prices not only from images but also in real-time from videos. Finally, we plan to create a real-time application that is able to calculate special offers and indicate if buying in bulk is cheaper then buying a single piece of a particular product [150], [151].

The second application introduced a novel approach for identifying and classifying the Romanian traditional motifs found on 4 different categories (clothes, ceramics, carpets, and painted eggs) by training a CNN model on a modified ResNet-50 architecture. We also implemented a system that can detect and identify through a webcam if the object in front of it contains a learned motif. In the experimental results, we show that 5 categories from which 4 containing Romanian traditional motifs (e.g. carpets, ceramics, clothes, painted eggs) are being detected and identified with high accuracy and reduced processing time. We obtain an overall accuracy of 99.4% and proved that with the implemented Grad-CAM technique, the proposed CNN model brings more interpretability, transparency, and trust. As future work, we intend to implement our model on the cloud and develop a mobile application in order to detect and identify the Romanian traditional motifs with the help of a smartphone in real-time. Additionally, to create an IA Dataset which

contains all Romanian traditional clothes organized by the regions in Romania they originate from.

The third application presents a method for identifying 34 animal classes corresponding to the most conventional animals found in the domestic areas of Europe by using four types of CNNs, namely VGG-19, InceptionV3, ResNet-50, and MobileNetV2. We also built a system capable of classifying all these 34 animal classes from images as well as in real-time from videos or a webcam. Additionally, our system is capable to automatically generate two new datasets, one dataset containing textual information (i.e. animal class name, date and time interval when the animal was present in the frame) and one dataset containing images of the animal classes present and identified in videos or in front of a webcam. Our experimental results show a high overall test accuracy for all 4 proposed architectures (90.56% for VGG-19 model, 93.41% for InceptionV3 model, 93.49% for ResNet-50 model and 94.54% for MobileNetV2 model), proving that such systems enable an unobtrusive method for gathering a rich collection of information about the vast numbers of animal classes that are being identified such as providing insights about what animal classes are present at a given date and time in a certain area and how they look, resulting in valuable datasets especially for researchers in the area of ecology.

Chapter 4 presents the construction, testing, and deployment of our solar tracker in order to make use of renewable and clean energy when powering a real-time DL-based system that can identify animal classes, store their textual information as well as their pictures in real-time without having to pay for electricity bills.

Regarding the construction part, we proposed a novel approach in the field of renewable energies by designing an efficient solar tracking device composed of an Arduino UNO board, two stepper motors, a pair of specialized L298N circuits and an optocoupler. We present a sensorless energy-saving solution based on the Cast-Shadow principle and a low-cost blocking mechanism for the stepper motors to reduce the overall power consumption of the system by 86.93%. Additionally, the experimental results of our autonomous solar tracker show a 45.77% voltage, 48.21% current and 53.62% power increase over the static PV panel by using monocrystalline solar cells.

Regarding the testing part, first, we presented a novel technique in testing the software code of a solar tracking device by using a White-box testing approach that makes use of a Wi-Fi module. We succeed in verifying if the wireless data transfers controlling the movements of the solar tracking device are in correspondence with the software code run on the Arduino Uno. In order to find out all the loopholes and possible breakpoints in our solar tracker software, we investigated Communication and Calculation Errors, Control Flow and Error handling faults by implementing unit testing techniques as well as custom code. Experimental results show that the proposed White-box testing strategy achieves a total coverage of 70.12% for all targeted errors from a total number of 4334 test cases organized in 4 batches and proves to be efficient from the fault coverage as well as the cost point of view.

Secondly, we proposed an OBIST architecture that uses an LFSR as a TPG and a MISR as a result gatherer for testing our solar tracking equipment composed of an Optocoupler, an Arduino UNO and two L298N Dual-H Bridges ICs. Due to the proposed fault injection strategy, we concluded that all 4 CUTs are prone to hardware faults and thus we implemented software as well as hardware solutions for the proposed OBIST. Experimental results show that the software implementation is

efficient in injecting test vectors and collecting the outputted signatures of the MISR device. In addition, we constructed a valid signature database for 3 of the CUTs and compared the MISR valid output signatures with its previous generated pseudo-random output values, resulting in 93.93% coverage for single bit-flip errors (last 8 bits, mutant) and 100% coverage for single stuck-at-faults (for 8, 12 and 16 random bits). Finally, the proposed OBIST achieves a total global coverage of 96.96% for the targeted errors, resulting in an efficient architecture regarding coverage and cost points of view.

Regarding deployment, we presented, to the best of our knowledge, the first solar-powered real-time DL-based system in the literature that is self-sustaining from the energy point of view, can run inference using 100% solar energy and which is composed of a dual-axis solar tracking device based on Cast-Shadow principle [17] and a low-power embedded platform called Nvidia Jetson TX2. In order to justify the minimal improvement costs of the solar panel as well as the choice of this embedded platform, experimental results, especially regarding the energy consumption while running 4 DL model architectures (VGG-19, InceptionV3, ResNet-50, and MobileNetV2) in real-time [15] are made also on a laptop containing the Nvidia GTX 1060 (6GB) GPU. Additionally, in order to reduce the power consumption of the entire solar-powered real-time DL-based system, we also implemented a motion detection method that triggers the inference process only when there is movement in the frame. Details about the construction of the entire solar-powered real-time DL-based system as well as calculations regarding the time needed for our accumulator to be charged with solar energy as well as discharged by the Nvidia Jetson TX2 when running the 4 DL models are also taken into consideration. Experimental results show that the Nvidia Jetson TX2 platform is a very good choice when designing an efficient solar-powered real-time DL-based system, consuming only around 10 Wh of power as compared to around 50 Wh consumed by a laptop.

As future work, we plan to run similar experiments also on other low-power platforms such as the Nvidia Jetson Nano Developer Kit, Google Coral, Raspberry Pi 4 Model B (4GB) and also on FPGAs, in order to show that real-time DL-based systems can run inference 100% on solar energy using even less energy than we demonstrated. Additionally to inference, we also want to train a few other state-of-the-art DL model architectures using 100% solar energy from our solar tracker on the above-mentioned platforms, with the intent to encourage new researchers to investigate the combination of green energy and AI, eventually proposing new green energy-based DL metrics. We believe that a "green" approach can lead researchers to a better understanding of how to evaluate the performance of DL-based systems and will also result in a more friendly and respectful attitude towards nature and life on this planet.

In Chapter 5, first, we introduce four metrics, two for inference called APC and APEC and two for training called TTCAPC and TTCAPEC for evaluating the performance of DL models and systems not only regarding their accuracy but also their energy consumption and cost. In our experimental results, we succeeded to prove that all four metrics are efficient, showing, to the best of our knowledge, for the first time in literature, that by using high accuracy together with low power consumption, especially green energy (e.g. solar energy) during training and inference, a DL model or system is evaluated as being much more performant than one that, despite having same accuracy, consumes more energy and uses a traditional power grid (paid electricity). We believe that these metrics will encourage future researchers to develop and use greener energy-based systems and that they

will evaluate their performance only based on how "green" they are and how less negative impact they have on our planet.

Secondly, we present a Computer Vision application that succeeds in bringing common DL features needed by a user (e.g. data scientist) when performing image classification related tasks into one easy to use and user-friendly GUI. From automatically gathering images and classifying them each in their respective class folder in a matter of minutes, to removing duplicates, sorting images, training and evaluating a DL model in a matter of minutes, all these features are integrated in a sensible and intuitive manner that requires no knowledge of programming and DL. Experimental results show that the proposed application has many unique advantages and also outperforms similar existent solutions. Additionally, this is the first Computer Vision application that incorporates the APC, APEC, TTCAPC and TTCAPEC metrics [20], which can be easily used to calculate and evaluate the performance of DL models and systems based not only on their accuracy but also on their energy consumption and cost, encouraging new generations of researchers to make use only of green energy when powering their DL-based systems [16].

In Chapter 6, we propose a low-cost and portable FPICT device that was able to reach 100% precision in single-point testing, 91.40% precision in multiple-point testing and overall precision of 95.70% for the entire measurement testing. We believe that, due to the simplicity of our proposed FPICT and user-friendliness as compared to the ones found in the industry, students will find learning and practicing the testing of PCBs to be more fun and interesting experience. The proposed FPICT has several advantages, mainly that it is very easy to learn and use, especially because of the C written configuration files (e.g. which can be easily modified by the students in a laboratory), it has a friendly user interface and can be also quickly connected to any existent computing platform that has a USB port (Desktop PCs, laptops, tablets). Also, the proposed FPICT provides students easy access to study and experiment with the inner workings of an FPT when operating on a real PCB board, which otherwise would have been almost impossible, given the fact that the FPTs found in the industry are very expensive (i.e. thousands of dollars [12] compared to our FPICT which costs around 25 dollars and require no extra costly licenses).

As future work, we plan to combine the proposed FPICT with other testing methods such as Boundary Scan in order to test the entire circuitry of the proposed solar tracking equipment based on Cast-Shadow principle and to show the different types of errors that can occur.

Chapter 7 presents several acceleration techniques for improving the throughput of an SHA-256 engine. The first acceleration technique eliminates the clock cycle used for hash value update and allows delivering a higher throughput due to the marginal performance loss associated with using an 8-operand adder instead of a 7-operand one. The second technique for improving performance implements the CPAs of the multi-operand adders in a fused manner to speed up the generation of the round functions. The proposed, fused design further increases the hash engine's performance while the synthesis driven approach for arranging the operands' order in the CSA tree further reduces the critical path. In addition to their performance improvements, the presented techniques can be applied in conjunction with other methods presented in the literature, such as loop unrolling, data precomputation in the previous round, to name only a few.

As future work, we plan to combine these techniques with other encryption algorithms in order to increase the security of DL-based systems that store sensitive and confidential data such as that belonging to pneumonia or COVID-19 patients.

8.1. Publications

To this date, I have the following publications submitted, accepted and presented at international conferences, relevant to the domain of Computers and Information Technology:

8.1.1. Book Chapters at International Publishers

1. **Sorin Liviu Jurj**, Flavius Opritoiu, Mircea Vladutiu "Environmentally-Friendly Metrics for Evaluating the Performance of Deep Learning Models and Systems", 17th International Symposium on Neural Networks (ISNN 2020), Cairo, Egypt, 2020. To appear. **ISI indexed.**

2. **Sorin Liviu Jurj**, Raul Rotar, Flavius Opritoiu, Mircea Vladutiu, "Efficient Implementation of a Self-Sufficient Solar-Powered Real-Time Deep Learning-Based System". In: Iliadis L., Angelov P., Jayne C., Pimenidis E. (eds) Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference. EANN 2020. Proceedings of the International Neural Networks Society, vol 2. Springer, Cham, pp. 99-118, doi: 10.1007/978-3-030-48791-1_7. **ISI indexed.**

3. **Sorin Liviu Jurj**, Flavius Opritoiu, Mircea Vladutiu „Deep Learning-Based Computer Vision Application with Multiple Built-In Data Science-Oriented Capabilities". In: Iliadis L., Angelov P., Jayne C., Pimenidis E. (eds) Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference. EANN 2020. Proceedings of the International Neural Networks Society, vol 2. Springer, Cham, pp. 47-69, doi: 10.1007/978-3-030-48791-1_4. **ISI indexed.**

8.1.2. International Conferences

1. **Sorin Liviu Jurj**, Raul Rotar, Flavius Opritoiu, Mircea Vladutiu „Online Built-In Self-Test Architecture for Automated Testing of a Solar Tracking Equipment", 2020 20th International Conference on Environment and Electrical Engineering (EEEIC), Madrid, Spain, 2020. To appear. **ISI indexed.**

2. **Sorin Liviu Jurj**, Raul Rotar, Flavius Opritoiu, Mircea Vladutiu „Affordable Flying Probe-Inspired In-Circuit-Tester for Printed Circuit Boards Evaluation with Application in Test Engineering Education", 2020 20th International Conference on Environment and Electrical Engineering (EEEIC), Madrid, Spain, 2020. To appear. **ISI indexed.**

3. **Sorin Liviu Jurj**, Flavius Opritoiu, Mircea Vladutiu "Real-time identification of animals found in domestic areas of Europe", In: Proc. SPIE 11433, Twelfth International Conference on Machine Vision (ICMV 2019), 1143313 (31 January 2020). doi: 10.1117/12.2556376. **ISI indexed.**

4. **Sorin Liviu Jurj**, Allen-Jasmin Farcas, Flavius Opritoiu, Mircea Vladutiu „Mobile Application for Receipt Fraud Detection Based on Optical Character Recognition", Proc. SPIE 11433, Twelfth International Conference on Machine Vision (ICMV 2019), 1143313 (31 January 2020). **ISI indexed.**

5. **Sorin Liviu Jurj**, Flavius Opritoiu, Mircea Vladutiu „Identification of Traditional Motifs using Convolutional Neural Networks", 2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME), Iasi, Romania, pp. 191-196, 2018. **ISI indexed.**

6. **Sorin Liviu Jurj**, Raul Rotar, Flavius Opritoiu, Mircea Vladutiu, "White-Box Testing Strategy for a Solar Tracking Device using NodeMCU Lua ESP8266 Wi-Fi Network Development Board Module", 2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME), pp. 53-60, 2018. **ISI indexed.**

7. Raul Rotar, **Sorin Liviu Jurj**, Flavius Opritoiu, Mircea Vladutiu, "Position Optimization Method for a Solar Tracking Device Using the Cast-Shadow Principle", 2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME), pp. 61-70, (2018). **ISI indexed.**

8. Flavius Opritoiu, **Sorin Liviu Jurj**, Mircea Vladutiu „Technological solutions for throughput improvement of a Secure Hash Algorithm-256 Engine", 2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME), Constanta, Romania, pp. 159-164, 2017. **ISI indexed.**

BIBLIOGRAPHY

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning" Nature, vol. 521, no. 7553, pp. 436–444, 2015

[2] C. Chen, A. Seff, A. Kornhauser, J. Xiao "DeepDriving: Learning affordance for direct perception in autonomous driving" Proc. 2015 IEEE International Conference on Computer Vision (ICCV 2015), pp. 2722-2730, Dec. 2015

[3] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks" Nature, vol. 542, no. 7639, pp. 115–118, Jan. 2017

[4] Pranav Rajpurkar, Awni Y. Hannun, Masoumeh Haghpanahi, Codie Bourn, Andrew Y. Ng "Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks" Computing Research Repository (CoRR), abs/1707.01836, July 2017

[5] Gabriele Angeletti, Barbara Caputo, Tatiana Tommasi "Adaptive Deep Learning through Visual Domain Localization", arXiv:1802.08833, Feb. 2018

[6] Soni, N., et al.: Impact of Artificial Intelligence on Businesses: from Research, Innovation, Market Deployment to Future Shifts in Business Models. In: arXiv:1905.02092v1, May, (2019)

[7] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, Bryan Catanzaro "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism", arXiv:1909.08053v4, March 2020.

[8] Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green AI. In: arXiv:1907.10597v3, August, (2019).

[9] Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in NLP. In: arXiv:1906.02243, (2019)

[10] Schmidt, V., Luccioni, A., Mukkavilli, S.K., Balasooriya, N., Sankaran, K., Chayes, J., Bengio, Y.: Visualizing the consequences of climate change using cycle-consistent adversarial networks. In: arXiv:1905.03709, (2019)

[11] Mohammad Sadegh Norouzzadeh et al. "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning" Proceedings of the National Academy of Sciences of the United States of America (PNAS), Vol. 115, Nr. 25, pp. 6315-6512, June 2018

[12] Huntron Workstation Tracker and Prober. [Online]. Available: http://shop.huntron.com/Workstation-License--Tracker-and-Prober_p_22.html

[13] Sorin Liviu Jurj, Flavius Opritoiu, Mircea Vladutiu „Identification of Traditional Motifs using Convolutional Neural Networks", 2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME), Iasi, Romania, pp. 191-196, 2018

[14] Sorin Liviu Jurj, Allen-Jasmin Farcas, Flavius Opritoiu, Mircea Vladutiu „Mobile Application for Receipt Fraud Detection Based on Optical Character Recognition", Proc. SPIE 11433, Twelfth International Conference on Machine Vision (ICMV 2019), 1143313 (31 January 2020);

[15] Jurj, S.L., Opritoiu, F., Vladutiu, M.: Real-time identification of animals found in domestic areas of Europe. In: Proc. SPIE 11433, Twelfth International Conference on Machine Vision (ICMV 2019), 1143313 (31 January 2020). doi: 10.1117/12.2556376.

[16] Jurj S.L., Rotar R., Opritoiu F., Vladutiu M. (2020) Efficient Implementation of a Self-sufficient Solar-Powered Real-Time Deep Learning-Based System. In: Iliadis L., Angelov P., Jayne C., Pimenidis E. (eds) Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference. EANN 2020. Proceedings of the International Neural Networks Society, vol 2. Springer, Cham, pp. 99-118, doi: 10.1007/978-3-030-48791-1_7.

[17] Raul Rotar, Sorin Liviu Jurj, Flavius Opritoiu, Mircea Vladutiu, "Position Optimization Method for a Solar Tracking Device Using the Cast-Shadow Principle", 2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME), pp. 61-70, (2018)

[18] Sorin Liviu Jurj, Raul Rotar, Flavius Opritoiu, Mircea Vladutiu, "White-Box Testing Strategy for a Solar Tracking Device using NodeMCU Lua ESP8266 Wi-Fi Network Development Board Module", 2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME), pp. 53-60, 2018

[19] Sorin Liviu Jurj, Raul Rotar, Flavius Opritoiu, Mircea Vladutiu „Online Built-In Self-Test Architecture for Automated Testing of a Solar Tracking Equipment", 2020 20th International Conference on Environment and Electrical Engineering (EEEIC), Madrid, Spain, 2020. To appear.

[20] Jurj, S.L., Opritoiu, F., Vladutiu, M.: Environmentally-Friendly Metrics for Evaluating the Performance of Deep Learning Models and Systems. To appear.

[21] Jurj S.L., Opritoiu F., Vladutiu M. (2020) Deep Learning-Based Computer Vision Application with Multiple Built-In Data Science-Oriented Capabilities. In: Iliadis L., Angelov P., Jayne C., Pimenidis E. (eds) Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference. EANN 2020. Proceedings of the International Neural Networks Society, vol 2. Springer, Cham, pp. 47-69, doi: 10.1007/978-3-030-48791-1_4.

[22] Sorin Liviu Jurj, Raul Rotar, Flavius Opritoiu, Mircea Vladutiu „Affordable Flying Probe-Inspired In-Circuit-Tester for Printed Circuit Boards Evaluation with

Application in Test Engineering Education", 2020 20th International Conference on Environment and Electrical Engineering (EEEIC), Madrid, Spain, 2020. To appear.

[23] Flavius Opritoiu, Sorin Liviu Jurj, Mircea Vladutiu „Technological solutions for throughput improvement of a Secure Hash Algorithm-256 Engine", 2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME), Constanta, Romania, pp. 159-164, 2017

[24] A. M. Turing, "Computing machinery and intelligence," Mind, vol. 59, no. 236, pp. 433–460, 1950

[25] Vernor Vinge "The Coming Technological Singularity: How to Survive in the Post-Human Era" Whole Earth Review (Winter Issue), 1993

[26] Abien Fred Agarap "Deep Learning using Rectified Linear Units (ReLU)" arXiv preprint arXiv:1803.08375, 22 March 2018

[27] Kaiming He, Xiangyu Zhang, et al. "Deep residual learning for image recognition" arXiv preprint arXiv:1512.03385, 2015

[28] Terence Parr, Jeremy Howard "The Matrix Calculus You Need For Deep Learning" arXiv:1802.01528, Feb. 2018

[29] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database" 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, 2009, pp. 248-255. [Online] Available: http://www.image-net.org/

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge" International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton "Imagenet classification with deep convolutional neural networks" in Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS) 2012 USA, Curran Associates Inc., pp. 1097–1105, 2012

[32] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang, H. Yang, and W. J. Dally, "ESE: efficient speech recognition engine with compressed LSTM on FPGA" Computing Research Repository (CoRR), vol. abs/1612.00694, 2016

[33] Silver, D. et al. "Mastering the game of Go with deep neural networks and tree search" Nature 529, pp. 484–489, 2016

[34] David Silver, et al. "Mastering the game of Go without human knowledge" Nature 550, pp. 354–359, 19 October 2017

[35] Dominik Scherer, Andreas Müller, and Sven Behnke "Evaluation of pooling operations in convolutional architectures for object recognition" In Proceedings of the 20th international conference on Artificial neural networks: Part III (ICANN'10),

Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 92-101, 2010

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov "Dropout: A simple way to prevent neural networks from overfitting" Journal of Machine Learning Research, pp. 1929–1958, 2014

[37] Sergey Ioffe, Christian Szegedy "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", arXiv:1502.03167, 2015

[38] Karen Simonyan and Andrew Zisserman "Very Deep Convolutional Networks for Large-scale Image Recognition" arXiv:1409.1556v6, 2015

[39] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich "Going Deeper with Convolutions", arXiv:1409.4842v1, 2014

[40] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna "Rethinking the Inception Architecture for Computer Vision", arXiv:1512.00567v3, 2015

[41] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár Microsoft "COCO: Common Objects in Context" arXiv:1405.0312, 2014

[42] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv:1704.04861v1, 2017

[43] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen "MobileNetV2: Inverted Residuals and Linear Bottlenecks", arXiv:1801.04381v4, 2019

[44] Y. LeCun, C. Cortes "The MNIST database of handwritten digits". [Online] Available: http://yann.lecun.com/exdb/mnist

[45] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al., "Tensorflow: Largescale machine learning on heterogeneous systems" vol. 1, 2015. [Online] Available: https://www.tensorflow.org/

[46] Chollet François et. al "Keras". [Online] Available: https://keras.io/

[47] International Energy Outlook 2017 (IEO2017), US Energy Information Administration (EIA). [Online] Available: https://www.eia.gov/outlooks/archive/ieo17/pdf/0484(2017).pdf, 2017

[48] Yan, J., Yang, Y., Elia Campana, P., He, J.: City-level analysis of subsidy-free solar photovoltaic electricity price, profits and grid parity in China. Nat. Energy(4), 709–717, (2019).

[49] Ram M., et al. Global Energy System based on 100% Renewable Energy – Power, Heat, Transport and Desalination Sectors. Study by Lappeenranta University of Technology and Energy Watch Group, Lappeenranta, Berlin, March 2019. [Online]. Available: http://energywatchgroup.org/wp-content/uploads/EWG_LUT_100RE_All_Sectors_Global_Report_2019.pdf

[50] Rolnick, D., et al.: Tackling Climate Change with Machine Learning. In: arXiv:1906.05433v2, November (2019).

[51] What Is Green Power?. [Online]. Available: https://www.epa.gov/greenpower/what-green-power

[52] Aneesh Kulkarni, Tushar Kshirsagar, Akash Laturia, P.H. Ghare "An Intelligent Solar Tracker for Photovoltaic Panels" 2013 Texas Instruments India Educators' Conference, pp. 390 – 393, 2013

[53] Deepthi. S, Ponni. A, Ranjitha. R, R. Dhanabal "Comparison of Efficiencies of Single-Axis Tracking System and Dual-Axis Tracking System with Fixed Mount" International Journal of Engineering Science and Innovative Technology (IJESIT), Vol. 2, Issue 2, ISSN: 2319-5967, 2013

[54] Hamid Allamehzadeh "Solar Energy Overview and Maximizing Power Output of a Solar Array Using Sun Trackers" IEEE Conference on Technologies for Sustainability (SusTech), pp. 14 – 19, 2016

[55] A. K. Suria, R. Mohamad Idris "Dual-axis solar tracker based on predictive control algorithms" IEEE Conference on Energy Conversion (CENCON), pp. 238 – 243, 2015

[56] Kyle Williams, Amer Qouneh "Internet of Things: Solar array tracker" IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 1057 – 1060, 2017

[57] Mircea Vladutiu, „Computer Arithmetic: Algorithms and Hardware Implementations" Springer Publishing Company, Incorporated, 2012

[58] Patare Snehal Dilip, Geethu Remadevi Somanathan, Ramesh Bhakthavatchalu "Reseeding LFSR for Test Pattern Generation", 2019 International Conference on Communication and Signal Processing (ICCSP), pp. 0921-0925, India, 2019

[59] P. Nigh, "Scan-based testing: the only practical solution for testing ASIC/consumer products," Proceedings. International Test Conference, Baltimore, MD, USA, 2002, pp. 1198-.

[60] Lubaszewski M., Huertas J.L. (2004) Test and Design-for-Test of Mixed-Signal Integrated Circuits. In: Reis R. (eds) Information Technology. IFIP International Federation for Information Processing, vol 157. Springer, Boston, MA

[61] Dávid Honfi & Zoltán Micskei "Classifying generated white-box tests: an exploratory study", Software Quality Journal, vol. 27, pp.1339–1380, 2019

[62] H. Michail, A. Kakarountas, A. Milidonis, and C. Goutis, "A Top-Down Design Methodology for Ultrahigh-Performance Hashing Cores," IEEE Transactions on Dependable and Secure Computing, vol. 6, pp. 255-268, 2009

[63] N. Ferguson, B. Schneier, and T. Kohno, Cryptography Engineering: Design Principles and Practical Applications: Wiley Publishing, 2010

[64] Chest X-Ray Images (Pneumonia), https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia/, last accessed 2020/04/07.

[65] Cohen, J.P., Morrison, P., Dao, L.: COVID-19 Image Data Collection, arXiv:2003.11597, (2020), https://github.com/ieee8023/covid-chestxray-dataset, last accessed 2020/04/07.

[66] National Institute of Standards and Technology, "FIPS 180-4, Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180-4," 2015

[67] "Carrefour Pay". [Online] Available: https://itunes.apple.com/ro/app/carrefourpay/id1149276582

[68] S. A. Sabab, S. S. Islam, M. J. Rana, and M. Hossain "eExpense: A Smart Approach to Track Everyday Expense", 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), pp. 136-141, 2018

[69] "Tesseract Open Source OCR Engine". [Online] Available: https://github.com/tesseract-ocr/tesseract

[70] Helinski, M., Kmieciak, M., Parkola, T. "Report on the comparison of Tesseract and ABBYY FineReader OCR engines", IMPACT technical report, 2012. [Online] Available: https://www.digitisation.eu/download/website-files/IMPACT_D-EXT2_Pilot_report_PSNC.pdf

[71] F. De Sousa Ribeiro et al. "An End-to-End Deep Neural Architecture for Optical Character Verification and Recognition in Retail Food Packaging", 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, pp. 2376-2380, 2018

[72] Mengwei Xu, Jiawei Liu, Yuanqiang Liu, Felix Xiaozhu Lin, Yunxin Liu, Xuanzhe Liu "A First Look at Deep Learning Apps on Smartphones", arXiv:1812.05448v2, 2019

[73] Hui Li, Peng Wang, Chunhua Shen "Toward End-to-End Car License Plate Detection and Recognition With Deep Neural Networks", IEEE Transactions on Intelligent Transportation Systems, Vol. 20, Issue 3, pp.1126-1136, 2019

[74] R. R. Palekar, S. U. Parab, D. P. Parikh and V. N. Kamble "Real time license plate detection using OpenCV and Tesseract", 2017 International Conference on Communication and Signal Processing (ICCSP), pp. 2111-2115, 2017

[75] N. Somyat and S. Nakariyakul "Thai Lottery Number Reader App for Blind Lottery Ticket Sellers", 2018 10th International Conference on Knowledge and Smart Technology (KST), pp. 139-143, 2018

[76] Brian Lao, Karthik A Jagadeesh "Convolutional Neural Networks for Fashion Classification and Object Detection", 2015

[77] Li Z., Sun Y., Wang F., Liu Q. (2015) Convolutional Neural Networks for Clothes Categories. In: Zha H., Chen X., Wang L., Miao Q. (eds) Computer Vision. CCCV 2015. Communications in Computer and Information Science, vol 547. Springer, Berlin, Heidelberg, 2015

[78] Evgeny Smirnov, Anton Kulinkin, Karina Ivanova, Michael Pogrebnyak "Deep Learning for Fast and Accurate Fashion Item Detection" 2016 ACM, ML4Fashion '16 August 13, 2016, San Francisco, CA, USA, 2016

[79] Eshwar S G, Gautham Ganesh Prabhu J, A V Rishikesh, Charan N A, Umadevi V "Apparel classification using Convolutional Neural Networks" 2016 International Conference on ICT in Business Industry & Government (ICTBIG), pp. 1-5, 2016

[80] S. M. Sofiqul Islam, Emon Kumar Dey, Md. Nurul Ahad Tawhid, B. M. Mainul Hossain "A CNN Based Approach for Garments Texture Design Classification" Advances in Technology Innovation, vol. 2, no. 4, pp. 119-125, 2017

[81] Kevin Matzen, Kavita Bala, Noah Snavely "StreetStyle: Exploring world-wide clothing styles from millions of photos" arXiv:1706.01869, 2017

[82] "iNaturalist". [Online] Available: https://www.inaturalist.org/

[83] "Merlin Photo ID". [Online] Available: http://merlin.allaboutbirds.org/

[84] H. Nguyen et al., "Animal Recognition and Identification with Deep Convolutional Neural Networks for Automated Wildlife Monitoring", 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Tokyo, pp. 40-49, 2017

[85] Gyanendra K. Verma and Pragya Gupta "Wild Animal Detection Using Deep Convolutional Neural Network", Advances in Intelligent Systems and Computing, pp. 327–338, 2018

[86] Manohar, N & Kumar, Y H & Rani, Radhika & Hemantha Kumar, G. "Convolutional Neural Network with SVM for Classification of Animal Images", In Sridhar V., Padma M., Rao K. (eds) Emerging Research in Electronics, Computer Science and Technology. Lecture Notes in Electrical Engineering, Springer, Singapore, vol. 545, pp 527-537, 2019

[87] Manohar N., Sharath Kumar Y.H., Kumar G.H., Rani R. "Deep Learning Approach for Classification of Animal Videos". In: Nagabhushan P., Guru D., Shekar B., Kumar Y. (eds) Data Analytics and Learning. Lecture Notes in Networks and Systems, vol 43. Springer, Singapore, pp. 421-431, 2019

[88] Zhongqi Miao et al. "A comparison of visual features used by humans and machines to classify wildlife", Cold Spring Harbor Laboratory, Oct. 2018

[89] Michael A Tabak et al. "Machine learning to classify animal species in camera trap images: applications in ecology", Methods in Ecology and Evolution, pp. 1-6, 2018

[90] Falah I. Mustafa , A.Salam Al-Ammri, Farouk F. Ahmad "Simple Design and Implementation of Solar tracking System Two Axis with Four Sensors for Baghdad city" 9th International Renewable Energy Congress (IREC), pp. 1 – 5, 2018

[91] Sidharth Makhija, Aishwarya Khatwani, Mohd. Faisal Khan, Vrinda Goel, M. Mani Roja "Design and Implementation of an Automated Dual-Axis Solar Tracker with Data-Logging" International Conference on Inventive Systems and Control (ICISC), pp. 1 – 4, 2017

[92] Yingxue Yao, Yeguang Hu, Shengdong Gao, Gang Yang, Jinguang Du "A multipurpose dual-axis solar tracker with two tracking strategies", Renewable Energy 72, pp. 88 –98, 2014

[93] Mohammed Saifuddin Munna, Mohammad Ariful Islam Bhuyan, Kazi Mustafizur Rahman, Md. Ashiqul Hoque "Design, implementation and performance analysis of a dual-axis autonomous solar tracker", 3rd International Conference on Green Energy and Technology (ICGET), pp. 1 –5, 2015

[94] Arifur Rahman Sagar, Sadik Al Saim, A.S.M. Ittehad and Hasan U. Zaman "A Novel Design of A Bi-level Automatic Solar Tracker Using Rotations Around Orthogonal Axes", 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1 – 6, 2017

[95] Hassan Fathabadi "Novel Online Sensorless Dual-Axis Sun Tracker", IEEE/ASME Transactions on Mechatronics, pp. 1-1. 10.1109/TMECH.2016.2611564, 2016

[96] Samah Laamami, Mouna Benhamed, Lassaad Sbita "Artificial neural network-based fault detection and classification for photovoltaic system" 2017 International Conference on Green Energy Conversion Systems (GECS), pp. 1-7, 2017

[97] Zhehan Yi, Amir H. Etemadi "Line-to-Line Fault Detection for Photovoltaic Arrays Based on Multiresolution Signal Decomposition and Two-Stage Support Vector Machine" IEEE Transactions on Industrial Electronics, Vol. 64, Issue 11, pp. 8546 – 8556, 2017

[98] B. Pradeep Kumar, G. Saravana Ilango, M. Jaya Bharata Reddy, Nagamani Chilakapati "Online Fault Detection and Diagnosis in Photovoltaic Systems Using Wavelet Packets" IEEE Journal of Photovoltaics,Vol. 8, Issue 1, pp. 257 – 265, 2018

[99] Faizan Khan, M. Yasin Ali, V.K.Sood, Faruk Bhuiyan, Phillip Insull, Faraz Ahmad "Simulation of Microgrid System with Distributed Generation" 2017 IEEE Electrical Power and Energy Conference (EPEC), 2017

[100] Mahmoud Dhimish, Violeta Holmes, Bruce Mehrdadi, Mark Dales "Simultaneous fault detection algorithm for grid-connected photovoltaic plants" Institution of Engineering and Technology (IET) Renewable Power Generation, Vol. 11, Issue 12, pp. 1565 – 1575, 2017

[101] Q. Liyan, B. Shi, Z. Xin, W. Ge, "Design of generic embedded memory built in self test circuit", 2009 9th International Conference on Electronic Measurement & Instruments, 2009, pp. 2-244-2-247.

[102] P. Moorthy, S. Saranya Bharathy, "An efficient test pattern generator for high fault coverage in built-in-self-test applications", 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2013, pp. 1-4.

[103] Rungsuptaweekoon, K., Visoottiviseth, V., Takano, R.: Evaluating the power efficiency of deep learning inference on embedded GPU systems. In: 2017 2nd International Conference on Information Technology (INCIT), pp. 1–5, IEEE, Nakhonpathom, Thailand, (2017).

[104] Yudin, D., Slavioglo, D.: Usage of fully convolutional network with clustering for traffic light detection. In: 2018 7th Mediterranean Conference on Embedded Computing (MECO), pp. 1–6, IEEE, Budva, Montenegro, (2018).

[105] Shihadeh, J., Ansari, A., Ozunfunmi, T.: Deep learning based image classification for remote medical diagnosis. In: 2018 IEEE Global Humanitarian Technology Conference (GHTC), pp. 1–8. IEEE, San Jose, CA, USA, (2018).

[106] Yin, X., Chen, L., Zhang, X., Gao, Z.: Object detection implementation and optimization on embedded GPU system. In: 2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp. 1–5. IEEE, Valencia, Spain, (2018).

[107] Arechiga, A.P., Michaels, A.J., Black, J.T.: Onboard image processing for small satellites. In: NAECON 2018 - IEEE National Aerospace and Electronics Conference, pp. 234–240, IEEE, Dayton, OH, USA, (2018).

[108] Vandersteegen, M., Van Beeck, K., Goedemé, T.: Super accurate low latency object detection on a surveillance UAV. In: arXiv:1904.02024v1, (2019).

[109] Špaňhel, J., Sochor, J., Makarov, A.: Detection of traffic violations of road users based on convolutional neural networks. In: 2018 14th Symposium on Neural Networks and Applications (NEUREL), pp. 1–6, IEEE, Belgrade, Serbia, (2018).

[110] Yuan, L., Lu, F.: Real-time ear detection based on embedded systems. In: 2018 International Conference on Machine Learning and Cybernetics (ICMLC), pp. 115–120. IEEE, Chengdu, China, (2018).

[111] Liu, S., Li, X., Gao, M., Cai, Y., Nian, R., Li, P., Yan, T., Lendasse, A.: Embedded online fish detection and tracking system via YOLOv3 and parallel correlation filter. In: OCEANS 2018 MTS/IEEE Charleston, pp. 1–6, IEEE, Charleston, SC, USA, (2018).

[112] Saypadith, S., Aramvith, S.: Real-time multiple face recognition using deep learning on embedded gpu system. In: 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), pp. 1318–1324, IEEE, Honolulu, HI, USA, (2018).

[113] Zhang, W., Sun, H., Zhao, D., Xu, L., Liu, X., Zhou, J., Ning, H., Guo, Y., Yang, S.: A streaming cloud platform for real-time video processing on embedded devices. In: IEEE Transactions on Cloud Computing, pp. 1–1. IEEE, (2019).

[114] Goyal, M., Reeves, N.D., Rajbhandari, S., Yap, M.H.: Robust Methods for Real-Time Diabetic Foot Ulcer Detection and Localization on Mobile Devices. IEEE J. Biomed Health Inf.(23), pp. 1730–1741, (2019).

[115] Modasshir, M., Li, A.Q., Rekleitis, I.: Deep neural networks: a comparison on different computing platforms. In: 2018 15th Conference on Computer and Robot Vision (CRV), pp. 383–389, IEEE, Toronto, ON, Canada, (2018).

[116] Vaidya, B., Paunwala, C.: Comparative analysis of motion based and feature based algorithms for object detection and tracking. In: 2017 International Conference on Soft Computing and its Engineering Applications (icSoftComp), pp. 1–7, IEEE, Changa, India, (2017).

[117] Zivkovic, Z.: Improved adaptive Gaussian mixture model for background subtraction. In: Proceedings of the 17th International Conference on Pattern Recognition (ICPR), pp. 28–31, IEEE, Cambridge, UK, (2004).

[118] Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: Proceedings 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149), pp. 246–252, IEEE, Fort Collins, CO, USA, (1999).

[119] Moon, S., Lee, J., Nam, D., Yoo, W., Kim, W.: A comparative study on preprocessing methods for object tracking in sports events. In: 2018 20th International Conference on Advanced Communication Technology (ICACT), pp. 460–462, IEEE, Chuncheon-si Gangwon-do, Korea (South), Korea (South), (2018).

[120] Godbehere, A.B., Matsukawa, A., Goldberg, K.: Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. In: 2012 American Control Conference (ACC), pp. 4305–4312, IEEE, Montreal, QC, Canada, (2012).

[121] Kaewtrakulpong, P., Bowden, R.: An improved adaptive background mixture model for realtime tracking with shadow detection. In: Remagnino, P., Jones, G.A., Paragios, N., Regazzoni, C.S. (eds.) Video-Based Surveillance Systems, pp. 135–144, Springer, Boston, MA, (2002).

[122] Cai, E., Juan, D., Stamoulis, D., Marculescu, D.: NeuralPower: Predict and Deploy Energy-Efficient Convolutional Neural Networks. In: arXiv:1710.05420, (2017).

[123] Rodrigues, C.F., Riley, G., Luján, M.: SyNERGY: An energy measurement and prediction framework for Convolutional Neural Networks on Jetson TX1. In: The 24th Int'l Conf on Parallel and Distributed Processing Techniques and Applications (PDPTA'18), CSREA Press, pp. 375-382, Las Vegas (2018).

[124] Bhat, G., Bagewadi, K., Lee, H.G., Ogras, U.Y.: REAP: Runtime Energy-Accuracy Optimization for Energy Harvesting IoT Devices. In: 2019 56th ACM/IEEE Design Automation Conference (DAC), pp. 1-6, Las Vegas, NV, USA (2019).

[125] Milosevic, J., Pena, D., Forembsky, A., Moloney, D., Malek, M.: It All Matters: Reporting Accuracy, Inference Time and Power Consumption for Face Emotion Recognition on Embedded Systems. In: arXiv:1807.00046, June (2018).

[126] Qasaimeh, M., Denolf, K., Lo, J., Vissers, K., Zambreno, J., Jones, P.: Comparing Energy Efficiency of CPU, GPU and FPGA Implementations for Vision Kernels. In: arXiv:1906.11879, May (2019).

[127] Gauen K., Rangan, R., Mohan, A., Lu, Y., Liu, W., Berg, C.A.: Low-Power Image Recognition Challenge. In: 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 99-104. (2017). doi: 10.1109/ASPDAC.2017.7858303.

[128] Canziani, A., Paszke, A., Culurciello, E.: An Analysis of Deep Neural Network Models for Practical Applications. In: arXiv:1605.07678v4, (2017).

[129] Bianco, S., Cadene, R., Celona, L., Napoletano, P.: Benchmark Analysis of Representative Deep Neural Network Architectures. In: arXiv:1810.00736v2, (2018). doi: 10.1109/ACCESS.2018.2877890.

[130] García-Martín E., Lavesson N., Grahn H., Casalicchio E., Boeva V.: How to Measure Energy Consumption in Machine Learning Algorithms. In: Alzate C. et al. (eds) ECML PKDD 2018 Workshops. ECML PKDD 2018. Lecture Notes in Computer Science, vol 11329. Springer, Cham. doi: 10.1007/978-3-030-13453-2_20.

[131] García-Martín, E., Rodrigues, C.F., Riley, G., Grahn, G.: Estimation of energy consumption in machine learning. Journal of Parallel and Distributed Computing(134), 75-88 (2019). doi: 10.1016/j.jpdc.2019.07.007.

[132] Verma, S., Wu, Q., Hanindhito, B., Jha, G., John, E., Radhakrishnan, R., John, L.K.: Metrics for Machine Learning Workload Benchmarking. In: International Workshop on Performance Analysis of Machine Learning Systems (FastPath) in conjunction with ISPASS, March (2019).

[133] Hezel, Nico; Barthel, Kai-Uwe: Dynamic Construction and Manipulation of Hierarchical Quartic Image Graphs. In: ICMR '18 Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval , pp. 513-516, New York, 2018

[134] "Image Sorter". https://visual-computing.com/project/imagesorter/, last accessed 2019/12/05.

[135] Meg Pirrung, Nathan Hilliard, Artëm Yankov, Nancy O'Brien, Paul Weidert, Courtney D Corley, Nathan O Hodas "Sharkzor: Interactive Deep Learning for Image Triage, Sort, and Summary", arXiv:1802.05316, 2018

[136] "Apple Photos". https://www.apple.com/ios/photos/pdf/Photos_Tech_Brief_Sept_2019.pdf, last accessed 2019/12/05.

[137] Tanuj Jain, Christopher Lennan, Zubin John, Dat Tran "Image Deduplicator (Imagededup)". https://idealo.github.io/imagededup/, last accessed 2019/12/07.

[138] Livia Faes et al. "Automated deep learning design for medical image classification by health-care professionals with no coding experience: a feasibility study", The Lancet Digital Health, Volume 1, Issue 5, pp. e232-e242, 2019

[139] Christopher Lennan, Malgorzata Adamczyk, Gunar Maiwald, Dat Tran "Image ATM". https://github.com/idealo/imageatm/, last accessed 2020/01/27.

[140] P. Radev and M. Shirvaikar "Enhancement of flying probe tester systems with automated optical inspection", 2006 Proceeding of the Thirty-Eighth Southeastern Symposium on System Theory, Cookeville, TN, pp. 367-371, 2006

[141] Y. Hiratsuka, et al. Shin "A design method for minimum cost path of flying probe in-circuit testers", Proceedings of SICE Annual Conference 2010, Taipei, pp. 2933-2936, 2010.

[142] S. C. Tan and S. T. W. Kit, "Fast retrievals of test-pad coordinates from photo images of printed circuit boards," 2016 International Conference on Advanced Mechatronic Systems (ICAMechS), Melbourne, VIC, pp. 464-467, 2016

[143] Soh Ying Seah, et al. "Combining ATE and flying probe in-circuit test strategies for load board verification and test", 2009 IEEE Instrumentation and Measurement Technology Conference, Singapore, pp. 1380-1385, 2009.

[144] L. Dadda, M. Macchetti, and J. Owen, "The design of a high speed ASIC unit for the hash function SHA-256 (384, 512)," in Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings, Vol.3, pp. 70-75, 2004

[145] R. Lien, T. Grembowski, and K. Gaj, "A 1 Gbit/s Partially Unrolled Architecture of Hash Functions SHA-1 and SHA-512," in Topics in Cryptology – CT-RSA 2004: The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings, T. Okamoto, Ed., ed Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 324-338, 2004

[146] Karez Abdulwahhab Hamad, Mehmet Kaya "A Detailed Analysis of Optical Character Recognition Technology", International Journal of Applied Mathematics Electronics and Computers, pp. 244-249, 2016

[147] "ABBYY". [Online] Available: https://www.abbyy.com/

[148] "Caffe2". [Online] Available: https://caffe2.ai/

[149] M. Basu "Gaussian-Based Edge-Detection Methods-A Survey" IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 32, no. 3, pp. 252-260, Aug. 2002

[150] "Now it's illegal to write down prices in a Tesco supermarket". [Online] Available: https://www.theguardian.com/money/blog/2011/sep/16/tesco-shopping-supermarket-prices-check-writing

[151] "Supermarket price labelling: Your photos". [Online] Available: https://www.bbc.com/news/business-32394711

[152] "Kivy". [Online] Available: https://kivy.org/

[153] "Buildozer - Generic Python packager for Android and iOS". [Online] Available: https://github.com/kivy/buildozer

[154] Goutte C., Gaussier E. "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation." In: Losada D.E., Fernández-Luna J.M. (eds) Advances in Information Retrieval (ECIR). Lecture Notes in Computer Science, vol 3408. Springer, Berlin, Heidelberg, 2005

[155] Available [Online]: https://www.facebook.com/LaBlouseRoumaine10/posts/1303317489786582

[156] Bihor Couture. [Online] Available: http://www.bihorcouture.com/

[157] Ramprasaath R. Selvaraju; Michael Cogswell; Abhishek Das; Ramakrishna Vedantam; Devi Parikh; Dhruv Batra "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization" 2017 IEEE International Conference on Computer Vision (ICCV), pp. 618-626, Oct. 2017

[158] Zachary C. Lipton "The Mythos of Model Interpretability" arXiv:1606.03490, 2017

[159] Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin "Why Should I Trust You?: Explaining the Predictions of Any Classifier" arXiv:1602.04938, 2016

[160] CUDA Nvidia. Compute unified device architecture programming guide, 2007

[161] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer "cudnn: Efficient primitives for deep learning" arXiv preprint arXiv:1410.0759, 2014

[162] Lukas Bossard, Matthias Dantone, Christian Leistner, Christian Wengert, Till Quack, Luc Van Gool "Apparel classification with style" Asian Conference on Computer Vision, Springer Berlin Heidelberg, pp. 321-335, 2012

[163] Zhi Li, Yubao Sun, Feng Wang, and Qingshan Liu "Convolutional Neural Networks for Clothes Categories" CCF Chinese Conference on Computer Vision, pp. 120-129, 2015

[164] Wolfgang Fiedler "New technologies for monitoring bird migration and behaviour", Ringing & Migration, Vol. 24, Issue 3, pp. 175-179, 2009

[165] "AI for Earth". [Online] Available: https://www.microsoft.com/en-us/ai/ai-for-earth

[166] R. V. Zaitsev, M. V. Kirichenko, G. S. Khrypunov, R. P. Migushchenko, L. V. Zaitseva "Hybrid solar generating module" IEEE International Young Scientists Forum on Applied Physics and Engineering (YSF), pp. 112 – 115, 2017

[167] Vukica M. Jovanovic, Orlando Ayala, Michael Seek, Sylvain Marsillac "Single axis solar tracker actuator location analysis", SoutheastCon, pp. 1– 5, 2016

[168] L. Bird, M. Milligan, and D. Lew "Integrating Variable Renewable Energy: Challenges and Solutions" Technical Report, pp. 1-10, September 2013

[169] K.A.Moharram et al. "Enhancing the performance of photovoltaic panels by water cooling", Ain Shams Engineering Journal, Vol. 4, Issue 4, pp. 869-877, December 2013

[170] Tarlochan Kaur, Shraiya Mahajan, Shilpa Verma Priyanka, Jaimala Gambhir "Arduino based Low Cost Active Dual Axis Solar Tracker", 1st IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), pp. 1 – 5, 2016

[171] "AUnit". Available [Online]: https://github.com/bxparks/AUnit

[172] Jasenka Dizdarevic, Francisco Carpio, Admela Jukan, Xavi Masip-Bruin "Survey of Communication Protocols for Internet-of-Things and Related Challenges of Fog and Cloud Computing Integration", arXiv:1804.01747, April 2018

[173] C. Riordan, R. Hulstron "What is an air mass 1.5 spectrum? (solar cell performance calculations)" IEEE Conference on Photovoltaic Specialists, vol.2, pp. 1085-1088, 1990

[174] Barry W. Williams "Principles Elements of Power Electronics", University of Strathclyde Glasgow, pp. 987, 2006

[175] S.V. Mitrofanov, D.K. Baykasenov, M.A. Suleev "Simulation Model of Autonomous Solar Power Plant with Dual-Axis Solar Tracker", 2018 International Ural Conference on Green Energy (UralCon), Chelyabinsk, pp. 90-96, 2018

[176] Aditya Sawant, Deepak Bondre, Apurav Joshi, Prasad Tambavekar, Apurv Deshmukh "Design and Analysis of Automated Dual Axis Solar Tracker Based on Light Sensors", 2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), pp. 454-459, 2018

[177] Nursadul Mamun, Nahidul Hoque Samrat, Nur Mohamma, Md. Jahirul Islam, Abdullah-Al-Mamun, Md. Habibur Rahaman Prince, Ridwan Adib, Md. Iftakherahmed

"Multi-directional solar tracker using low cost photo sensor matrix" International Conference on Informatics, Electronics & Vision (ICIEV), pp. 1 – 5, 2014

[178] Szilárd Bulárka, Aurel Gontean "Hybrid-loop controlled solar tracker for hybrid solar energy harvester", 2017 25th Telecommunication Forum (TELFOR), Belgrade, pp. 1-4, 2017

[179] Shashwati Ray, Abhishek Kumar Tripathi "Design and Development of Tilted Single Axis and Azimuth-Altitude Dual Axis Solar Tracking Systems", IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), pp. 1-6, 2016

[180] Manish Kumar, Santosh Kumar Singh, Dr. R. K. Dwivedi "A Comparative Study of Black Box Testing and White Box Testing Techniques" International Journal of Advance Research in Computer Science and Management Studies (IJARCSMS), Volume 3, Issue 10, October 2015, pp. 32–44

[181] Muhammad Nouman, Usman Pervez, Osman Hasan, Kashif Saghar "Software testing: A survey and tutorial on white and black-box testing of C/C++ programs", IEEE Region 10 Symposium (TENSYMP), May 2016, pp. 225-230

[182] "Node-RED - Flow-based programming for the Internet of Things". Available [Online]: https://nodered.org/

[183] David R. Brooks, "Arduino-Based Dataloggers: Hardware and Software", Institute for Earth Science Research and Education, Vol. 1.3, pp. 29-31, February 2016

[184] "MQTT becomes OASIS Standard". Available [Online]: https://www.oasis-open.org/news/announcements/mqtt-version-3-1-1-becomes-an-oasis-standard

[185] Nidhi Gupta "Different Approaches to White Box Testing to Find Bug" International Journal of Advanced Research in Computer Science & Technology (IJARCST), Vol. 2, Issue 3, 2014

[186] "Arduino UNO Schematic". Available [Online]: https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf

[187] N. Song, J. Qin, X. Pan, Y. Deng, "Fault injection methodology and tools," Proceedings of 2011 International Conference on Electronics and Optoelectronics, Vol. 1, 2011, pp. 47-50

[188] M. A. Mioc, "Simulation study of using shift registers based on 16-th Degree Primitive Polynomials", In Proceedings of INASE Conference 2015, Wien, March 15-17, New Developments in Pure and Applied Mathematics, 2015, pp. 363-369. [Online]. Available: http://www.inase.org/library/2015/vienna/bypaper/MAPUR/MAPUR-58.pdf.

[189] UN Sustainable Development Goals, https://www.un.org/sustainabledevelopment/sustainable-development-goals/, last accessed 2020/05/27.

[190] Nvidia Jetson TX2, https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/, last accessed 2019/12/01.

[191] Carballo, J.A., Bonilla, J., Berenguel, M., Fernandez-Reche, J., García, G.: New approach for solar tracking systems based on computer vision, low cost hardware and deep learning. In: arXiv:1809.07048v1, (2018).

[192] Bradski, G.: The OpenCV library. In: Dr. Dobb's J. Softw. Tools(120), pp. 122–125, (2000).

[193] Berthod, C., Kristensen, S.T., Strandberg, R., Odden, J.O., Nie, S., Hameiri, Z., Sætre, T.O.: Temperature sensitivity of multicrystalline silicon solar cells. IEEE J. Photovolt. (9), pp. 957–964, (2019).

[194] Tensorflow Protobuf, https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/protobuf/config.proto/, last accessed 2020/05/27.

[195] Twilio Programmable SMS, https://www.twilio.com/docs/sms/send-messages, last accessed 2020/05/27.

[196] JetPack, https://developer.nvidia.com/embedded/jetpack/, last accessed 2020/05/27.

[197] Convenient Power Measurement Script on the Jetson TX2/Tegra X2, https://embeddeddl.wordpress.com/2018/04/25/convenient-power-measurements-on-the-jetson-tx2-tegra-x2-board/, last accessed 2020/05/27.

[198] Mattson, P., et al.: MLPerf Training Benchmark. In: arXiv:1910.01500v2, October (2019).

[199] Reddi, V.J., et al.: MLPerf Inference Benchmark. In: arXiv:1911.02549, November (2019).

[200] Coleman, C., Narayanan, D., Kang, D., Zhao, T., Zhang, J., Nardi, L., Bailis, P., Olukotun, K., Ré, C., Zaharia, M.: DAWNBench: An End-to-End Deep Learning Benchmark and Competition. In: ML Systems Workshop @ NIPS, (2017).

[201] Ben-Nun, T., Besta, M., Huber, S., Ziogas, A.N., Peter, D., Hoefler, T.: A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning. In: arXiv:1901.10183v2, June, (2019).

[202] Zhu, H., Akrout, M., Zheng, B., Pelegris, A., Phanishayee, A., Schroeder, B., Pekhimenko, G.: TBD: Benchmarking and Analyzing Deep Neural Network Training. In: arXiv:1803.06905v2, April (2018).

[203] Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. In: arXiv:1706.02677v2, April, (2018).

[204] Han, S., Mao, H., Dally, W.J.: Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In: arXiv:1510.00149v5, February, (2016).

[205] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: arXiv:1412.6980v9, January, (2017).

[206] Accelerating Facebook's infrastructure with application-specific hardware, https://engineering.fb.com/data-center-engineering/accelerating-infrastructure/, last accessed 2020/05/27.

[207] Strom-Report, https://1-stromvergleich.com/electricity-prices-europe/, last accessed 2019/11/30.

[208] Ashmore, R., Calinescu, R., Paterson, C.: Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. In: arXiv:1905.04223, May, (2019)

[209] Roh, Y., Heo, G., Whang, S.E.: A Survey on Data Collection for Machine Learning: a Big Data -- AI Integration Perspective. In: arXiv:1811.03402v2, August, (2019)

[210] Quandl, https://www.quandl.com/, last accessed 2020/02/29.

[211] URSA, https://www.ursaspace.com/, last accessed 2020/02/29.

[212] Kaggle, https://www.kaggle.com/datasets, last accessed 2020/02/29.

[213] Icrawler, https://pypi.org/project/icrawler/, last accessed 2020/02/29.

[214] Amazon Mechanical Turk, https://www.mturk.com/, last accessed 2020/02/29.

[215] Barz, B., Denzler, J.: Do we train on test data? Purging CIFAR of near-duplicates. In: arXiv:1902.00423, February (2019)

[216] Swanson, A., et al.: Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna. In: Sci Data 2, 150026 (2015), doi:10.1038/sdata.2015.26

[217] Nakkiran, P., et al.: Deep Double Descent: Where Bigger Models and More Data Hurt. In: arXiv:1912.02292, December, (2019)

[218] MNIST converted to PNG format, https://github.com/myleott/mnist_png, last accessed 2020/02/29.

[219] T. Nguyen and N. Rezvani "Printed Circuit Board Assembly Test Process and Design for Testability", 9th International Symposium on Quality Electronic Design (ISQED), San Jose, CA, pp. 594-599, 2008

[220] Y. Niu, L. Wu, L. Wang, X. Zhang, and J. Xu, "A Configurable IPSec Processor for High Performance In-Line Security Network Processor," in Computational

Intelligence and Security (CIS), 2011 Seventh International Conference on, pp. 674-678, 2011

[221] A. Thiruneelakandan and T. Thirumurugan, "An approach towards improved cyber security by hardware acceleration of OpenSSL cryptographic functions," 2011 International Conference in Electronics, Communication and Computing Technologies (ICECCT), pp. 13-16, 2011

[222] Y. Yarom, D. Genkin, and N. Heninger, "CacheBleed: A Timing Attack on OpenSSL Constant Time RSA," in Cryptographic Hardware and Embedded Systems – CHES 2016: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings, B. Gierlichs and Y. A. Poschmann, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 346-367, 2016

[223] R. Zimmermann, "Binary Adder Architectures for Cell-Based VLSI and Their Synthesis," Ph.D. Thesis, Swiss Federal Institute of Technology, Zurich, 1997.

[224] S. Roy, M. Choudhury, R. Puri, and D. Z. Pan, "Polynomial Time Algorithm for Area and Power Efficient Adder Synthesis in High-Performance Designs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, pp. 820-831, 2016.

[225] J. E. Stine, J. Grad, I. Castellanos, J. Blank, V. Dave, M. Prakash, et al., "A Framework for High-Level Synthesis of System-on-Chip Designs," presented at the Proceedings of the 2005 IEEE International Conference on Microelectronic Systems Education, 2005.