Robust Machine Learning Systems: Reliability and Security for Deep Neural Networks

Muhammad Abdullah Hanif, Faiq Khalid, Rachmad Vidya Wicaksana Putra, Semeen Rehman, Muhammad Shafique
Vienna University of Technology (TU-Wien), Vienna, Austria
{muhammad.hanif, faiq.khalid, rachmad.putra, semeen.rehman, muhammad.shafique}@tuwien.ac.at

***Abstract*— Machine learning is commonly being used in almost all the areas that involve advanced data analytics and intelligent control. From applications like Natural Language Processing (NLP) to autonomous driving are based upon machine learning algorithms. An increasing trend is observed in the use of Deep Neural Networks (DNNs) for such applications. While the slight inaccuracy in applications like NLP does not have any severe consequences, it is not the same for other safety-critical applications, like autonomous driving and smart healthcare, where a small error can lead to catastrophic effects. Apart from high-accuracy DNN algorithms, there is a significant need for robust machine learning systems and hardware architectures that can generate reliable and trustworthy results in the presence of hardware-level faults while also preserving security and privacy. This paper provides an overview of the challenges being faced in ensuring reliable and secure execution of DNNs. To address the challenges, we present several techniques for analyzing and mitigating the reliability and security threats in machine learning systems.**

Keywords—Machine Learning, Deep Neural Networks, DNNs, Reliability, Security, Soft Errors, Aging, Process Variations, Hardware.

I. INTRODUCTION

The rapid advancements in technology and the ambition for automation in every aspect of life have triggered a huge amount of research and development in the area of machine learning (ML). The main concept behind ML is to enable the machines to learn complex tasks directly from the provided dataset, without any human involvement. This allows the machines to learn more appropriate features, specific to a particular application, which are normally unnoticeable by humans. Hence, these algorithms easily outperform the programs developed entirely by humans [1] and have recently reported to outperform humans as well in some particular applications [2].

ML algorithms are nowadays commonly being used in complex safety-critical applications. Therefore, it is highly desirable to build robust algorithms that offer high accuracy for such applications [14]. Deep Neural Networks (DNNs) are currently the state-of-the-art ML algorithms for a majority of the Artificial Intelligence (AI) applications. However, the high accuracy offered by these networks comes at the cost of high computational requirements. Accelerators and application-specific hardware designs have been proposed to boost the performance of these algorithms under defined resource constraints [14]. *Apart from robust algorithms and high-performance hardware, there is also a significant need for developing systems and hardware architectures that can execute these algorithms in a robust manner.*

Robustness in system design mainly refers to two crucial aspects: (1) Reliability, and (2) Security.

Reliability Threats: In hardware design, reliability refers to the resilience against vulnerabilities to faults in the hardware/devices. These vulnerabilities can mainly be attributed to the following factors (also shown in Fig. 1).

1. *Soft Errors:* Transient bit flips caused by external events like high energy particle strikes, and/or internal events like noise transients and electromagnetic interference [3].
2. *Aging:* Degradation in circuits' characteristics (e.g., delay) over time.
3. *Process Variations:* Variations in the leakage power and delay due to the imperfection in the chip fabrication process.
4. *Temperature:* It indirectly affects the reliability by increasing the Soft Error Rate (SER) as well as the rate of aging of a circuit.

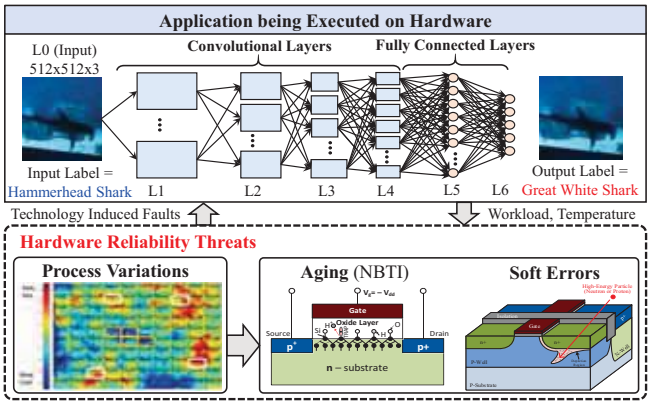

Fig. 1: Prominent types of reliability threats, i.e., Soft errors [3][9], Aging [9][10], and Process Variations [9].

In the past few decades, these vulnerabilities have increased significantly because of the reducing size of technology nodes [4]. These vulnerabilities introduce different types of transient and permanent faults in hardware and thereby disrupt the overall functionality of the applications. Various techniques have been proposed for the detection and mitigation of these faults, which are mainly based upon redundancy (e.g., DMR: dual modular redundancy [5], and TMR: triple modular redundancy [6]) and Razor [7][8] based approaches.

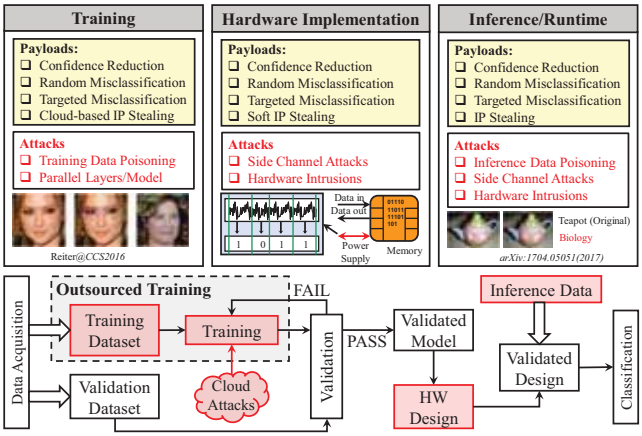

Fig. 2: An Overview of Security Threats/Attacks and their respective payloads for Machine Learning Algorithms during Training, Inference, and their respective Hardware Implementations.

Security Threats: Similarly, due to increasing trends of outsourcing and data dependencies, machine learning algorithms (e.g., DNNs) have several inherent security vulnerabilities, which can be exploited to perform several security attacks [14] (i.e., confidence reduction (ambiguity in classification), random or targeted misclassification). Therefore, with respect to the manufacturing cycle, security threats on ML algorithms can be categorized as follows:

1. *Training:* An attacker can exploit the training dataset, tools or architecture/model [15], i.e., adding parallel layers or neurons, to perform security attacks, as shown in Fig. 2.

978-1-5386-5992-2/18/$31.00 ©2018 IEEE

2. *Hardware Implementation*: In this phase, an attacker can either intrude the hardware or steal the IP of a trained model (i.e., side channel attacks) [15], as shown in Fig. 2.
3. Inference: In this phase, an attacker can either manipulate the inference data or intrude the hardware [15]. Moreover, the attacker can also perform side-channel attacks for IP stealing (see Fig. 2).

Therefore, to address these threats, several countermeasures have been developed, i.e., *data encryption* (to mitigate data poisoning) [16][17][18] [19], *local training/data sanitization* (to mitigate data poisoning) [20][21], *formal verification* (to mitigate hardware intrusions) [22][23][31][32], *runtime monitoring* (to mitigate hardware intrusions) [32][33][34], and *obfuscation* (to mitigate IP stealing) techniques [35][36].

In the light of the above discussion, the **main contributions** of this paper are as follows:

- We identify the research challenges associated with developing reliable and secure hardware/systems for machine learning.
- We provide an overview of the state-of-the-art techniques used for ensuring robustness in machine learning systems.
- We provide an overview of our methodology for building robust systems for machine learning based applications.

II. Research Challenges

This section highlights the main challenges in developing robust machine learning systems for AI applications.

A. Reliability related challenges

As mentioned earlier in Section I, the shrinking geometries, lower power voltages, and higher frequencies in modern on-chip systems have a negative impact on the reliability of a circuit, as they collectively increase the overall number of occurrences of transient and permanent faults [3]. Here we highlight few of the most important/prominent research challenges that need to be addressed for handling reliability in high-performance DNN architectures.

1. *How to characterize reliability-wise importance* of different parts of DNNs?
2. *Which hardware and software level techniques can be employed for detecting and mitigating reliability* related issues while preserving resource efficiency of the machine learning system?
3. *How to enable cross-layer reliability* for ensuring near-optimal resource efficiency?

B. Security related challenges

Based on the inherent security vulnerabilities of ML algorithms during training, inference, and their hardware implementations, we formulate the following research challenges to ensure security, privacy and confidentiality of the ML algorithms.

1. *How to ensure the privacy* of the training (especially, in outsourced training) and hyper-parameters (especially, in transfer learning)?
2. *How to avoid input data manipulation* during the inference stage?
3. *How to protect and obfuscate the ML-based IPs* from IP stealing attacks, i.e., side channel attacks?
4. *How to validate the correctness* of ML algorithms in hardware implementation?
5. *How to securely execute ML algorithms* on third-party hardware accelerators?

III. Robust Machine Learning Systems

This section provides a brief overview of our methodology for developing robust machine learning systems. The section is divided into two parts: the first part covers the methodology for making a DNN system reliable, while the second part covers the methods for ensuring security and privacy in DNN execution.

A. Reliability

1) Case study to illustrate the significance of reliability in ML

To illustrate the significance of reliability for machine learning systems, we present a case study where we simulate a neural network in the presence of memory faults (only in the region where the parameters, i.e., weights, of a network are stored). The case-study is based on image classification on the ImageNet dataset [24] using the VGG-f network [25]. For this evaluation, we assumed 32-bit floating point precision for both weights and inputs. An illustration of the scenario is shown in Fig. 3.

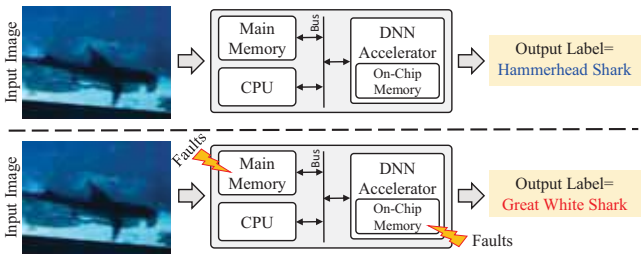

Fig. 3: Experimental setup for illustrating the impact of memory faults in DNN execution.

The structure of the floating point number system used for this analysis is shown in Fig. 4. We divided the analysis into two parts: (1) Bit flips from 0'b to 1'b; and (2) Bit flips from 1'b to 0'b. Also, to analyze the impact of the bit flips at particular locations in the words, we injected bit-flip errors individually at different bit location of a fraction of weights of a network layer. The results of 0'b to 1'b bit flip error injection in the first layer of the network is presented in Fig. 5a. As can be seen from the figure that *even a single error at the "e_7" location of a single weight of the layer leads to unacceptable accuracy loss*. The reported accuracy was computed by performing 1000 tests on randomly selected images from the ImageNet validation set, while using fault injection in the network. A decreasing trend in the impact of these bit flips was observed with the decreasing significance of the bit locations. This can be observed by analyzing the accuracy reported for the e_3, e_2, e_1, m_{22}, m_{21}, and m_{20} bit locations in the same figure.

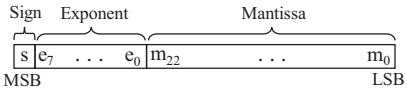

Fig. 4: Single precision floating point storage format used in our DNN design.

Similarly, an analysis was performed where 1'b to 0'b bit flips were injected in the weights. Fig. 5b illustrates the results of the experiment. The results clearly indicate that *the 1'b to 0'b bit flip does **not** have drastic effects on the accuracy of the network*, when compared to 0'b to 1'b bit flips. This is mainly because of the fact that the 0'b to 1'b bit flips in the exponent part of the floating point number results in a significant increase in the magnitude of the output which is then propagated to the output and results in random classification of the sample.

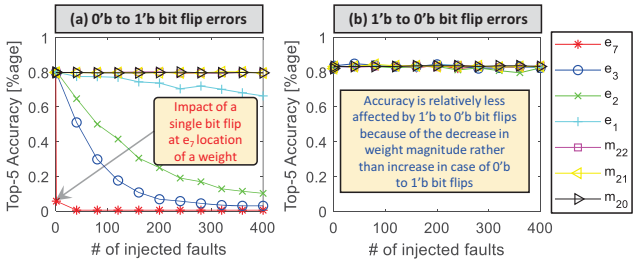

Fig. 5: Impact of bit flip errors on the accuracy of *VGG-f* network used for an image classification application.

The aforementioned case-study highlights the need studying the impact of different types of bit-flip faults in different components of a DNN systems, and leveraging this knowledge for designing and

optimizing fault-tolerant methods that can help avoiding errors which lead to catastrophic results in case of safety-critical applications in the areas of Internet-of-Things (IoT) and Cyber Physical Systems (CPS).

2) Our methodology for designing reliable ML systems

Fig. 6 illustrates our complete design methodology for building reliable systems that can produce reliable results in the presence of hardware-level faults. Our methodology employ different techniques at both (1) Design-Time, and (2) Run-Time, as shown in the figure.

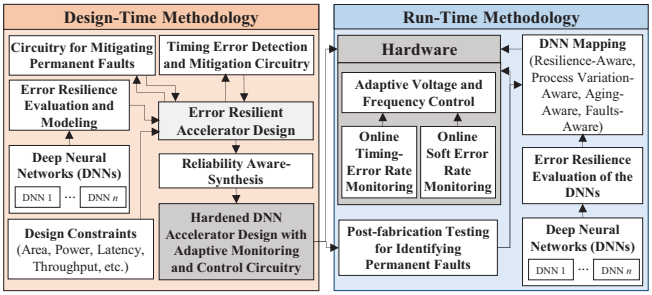

Fig. 6: Our methodology for building reliable systems for ML applications.

The **design-time** steps focus on building a highly reliable error-resilient DNN accelerator, which also provides support for run-time reliability methods. Given a set of design constraints and error resilience of the DNNs, a baseline accelerator is first designed. Techniques like [26] and [27] can be used for building the accelerator while considering techniques like [28] for error resilience evaluation and modeling of the DNNs. Additional circuitry for detecting and mitigating timing errors and mitigating permanent faults is added afterwards. Such techniques have also been employed in the recent works, like in [12][13]. The accelerator is then built using reliability aware-synthesis techniques (e.g., using the methodology of [11]), in order to improve the dependability of the complete/part of the circuit. Apart from the logic circuits, Error Correction Code (ECC)-based techniques can be used for introducing reliability in the memories. *In the proposed design-time steps of our methodology, redundancy-based approaches (like DMR and TMR) are highly discouraged because of the fact that they incur significant overheads and are highly resource hungry.* If required, we propose to build the system in a manner which can support instruction-level redundancy without incurring much power and area overheads (for instance, using the technique of [29]), or using partial hardening of critical hardware components only (as proposed in [30]).

The **run-time methods** considered in the proposed methodology are: (1) Fault-aware mapping of the network(s) based upon the information of permanent faults in the hardware and the error resilience of different part of the network(s); (2) Adaptive voltage and frequency scaling based upon the number of occurrences of soft and timing errors; and (3) Software-level redundancy (optional, if required). The first approach in the aforementioned list takes advantage of the error resilience property of the machine learning applications for offering as much performance and resource benefits as possible, while the rest of the two approaches ensure reliability by trading it off with performance and/or power/energy efficiency.

B. *Security*

In this section, we provide a brief overview of different threat models for the ML systems, an example of a random misclassification attack on LeNet, and a brief discussion on the possible security measures for ML-based applications and systems.

1) Threat Models

A threat model defines the capabilities and goals of an attacker under realistic assumptions. Defining a threat model is considered as one of the most important steps in designing security attacks or defense mechanisms. Therefore, to define a threat model, we provide a brief overview of the manufacturing cycle of ML-based applications/systems (see Fig. 7) that has the following four actors.

a) 3rd Party Cloud Training Platforms: In case 3rd party (3P) cloud platforms are untrusted, they can manipulate the training datasets, models/architectures or hyper-parameters to introduce ambiguity or to perform misclassification.

b) IP Providers: Similarly, in case, IP providers are untrusted, they can also manipulate the training datasets, models/architectures or other parameters, which are not accessible to 3P cloud platforms.

c) Manufacturers: In case, manufacturers are untrusted, they can intrude the hardware for security attacks (confidence reduction, misclassification). Additionally, they can perform side-channel attacks to steal the IP in form of the trained ML algorithm.

d) Users: Similarly, users (i.e., attackers) can manipulate the inference data and can intrude the hardware depending upon their capabilities. Moreover, they can also perform side-channel attacks to steal the IP in the form of the trained ML algorithm.

Thus, the trustworthiness of the above-mentioned factors defines 15 possible threat models for ML-based applications. For instance, *if the 3P cloud training platforms are untrusted and the rest of the factors are trusted, it can be defined as one of the possible threat model.*

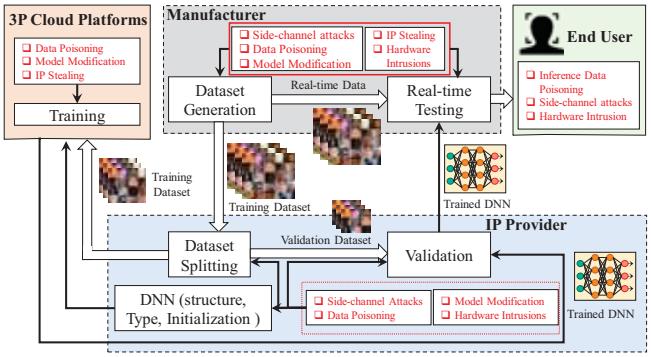

Fig. 7: Security threats with respect to different actors involved in Manufacturing cycle of ML-based Applications.

2) Random Misclassification Attack on the LeNet

To illustrate the significance of security vulnerabilities in machine learning, we demonstrate a random misclassification attack through *data poisoning* on the MNIST dataset [38] during the training of the LeNet [37]; as shown in Fig. 8. Only 1% of the data samples in the MNIST are poisoned. The main motivation behind choosing the noise based random attack during the training phase is that, the misclassification probability of training attacks is greater than the probability of the similar attack during inference.

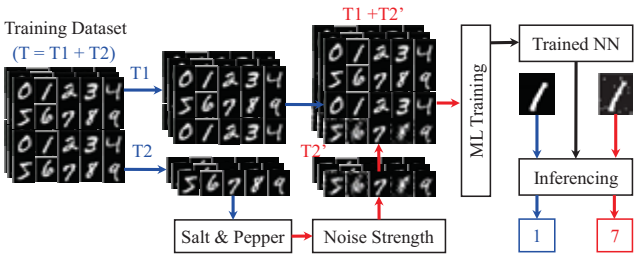

Fig. 8: Our experimental setup for random misclassification attack on LeNet.

To illustrate the effectiveness of this attack, we perform the inference using an intruded LeNet and poisoned as well as un-poisoned data samples, as shown in Fig. 8. Fig. 9 shows the random misclassification of different intruded samples. For example, the intruded LeNet randomly classifies the poisoned data sample with label 0 as 8. So, it can be concluded that introducing the noise during the training can introduce the random misclassification. However, randomly introduced noise in the training samples can affect the inference accuracy of the network, e.g., in the demonstrated attack, the accuracy loss is 0.5% (see Fig. 9), which in case of LeNet might be considered significant by the research community. Therefore, such attacks can be detected during the validation phase, and can only be used to analyze the security vulnerabilities in ML algorithms.

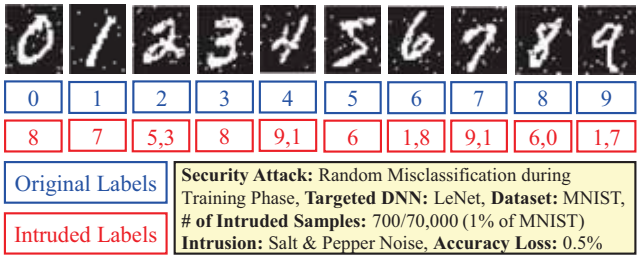

Fig. 9: Random misclassification attack (i.e., introducing Salt & Pepper noise in 1% samples of the MNIST dataset) on LeNet during the training phase.

3) Security Measures

To mitigate the security threats, we provide a brief overview of several possible countermeasures (see the highlighted boxes in Fig. 10; also see [14]), which can be classified into the following three categories based on their implication during the manufacturing cycle:

1. **Training:** Depending upon the targeted vulnerabilities, there are several possible countermeasures: encrypting the training dataset before outsourcing it [16][17][18][19], performing transfer learning based-local training to mitigate the weight/model manipulations [20][21], and outsourcing the training to multiple 3P cloud platforms to identify any intrusion by comparing the redundant trained models [39].
2. **Hardware Implementation:** During this phase, typical hardware security techniques can be applied, i.e., formal method-based analysis (mathematical modeling [31], property checker [32], SMT solver [22], SAT solver [23]), side channel analysis based runtime monitoring setups to detect dormant intrusions [32][33][34], and obfuscation (to mitigate IP stealing) techniques [35][36].
3. **Inference:** Similarly, obfuscation techniques and run-time monitoring setups can also be applied. However, in inference potentially facing several security threats make the security measures very complex [40]. For example, to avoid the data poisoning attacks, encryption is one of the possible countermeasures, as shown in Fig. 10.

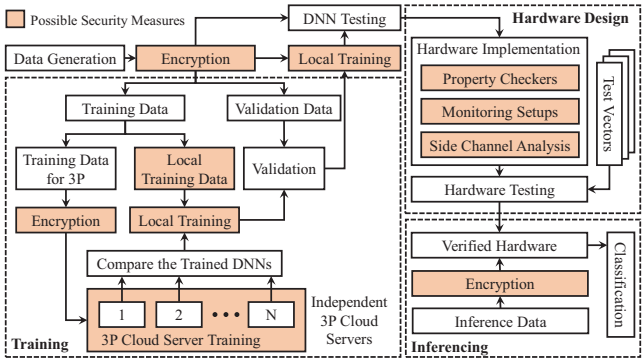

Fig. 10: Possible countermeasures (Highlighted blocks) against security attacks on ML-based applications.

IV. Conclusion

In this paper, we discussed reliability and security related challenges which are key for ensuring robustness in a machine learning-based system, especially when targeting DNNs. The reliability challenges are discussed from the perspective of ensuring reliability in a resource-efficient manner by considering application-specific characteristics of Deep Neural Networks (DNNs). Similarly, security challenges and attacks and possible threat models have been discussed with respect to different actors and phases in manufacturing cycle of ML-based applications/systems. Furthermore, the paper presents our methodology for making ML hardware reliable, as well as different countermeasures against security attacks.

Acknowledgment

This Work is supported in parts by the German Research Foundation (DFG) as part of the GetSURE project in the scope of SPP-1500 (http://spp1500.itec.kit.edu) priority program, "Dependable Embedded Systems".

References

[1] A. Krizhevsky et al., "Imagenet classification with deep convolutional neural networks." **NIPS**. 2012.
[2] E. Gibney, "Google AI algorithm masters ancient game of Go", **Nature News**, 529(7587), 445, 2016.
[3] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies", IEEE **TDMR**, vol. 5, no. 3, pp.305-316, 2005.
[4] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," in IEEE **Micro**, vol. 25, no. 6, pp. 10-16, Nov.-Dec. 2005.
[5] R. Vadlamani et al., "Multicore soft error rate stabilization using adaptive dual modular redundancy", in IEEE **DATE**, pp. 27-32, 2010.
[6] R. R. Lyons, et al., "The use of triple-modular redundancy to improve computer reliability." IBM Journal of Research and Development 6.2 (1962): 200-209.
[7] D. Ernst et al., "Razor: circuitlevel correction of timing errors for low-power operation", in IEEE **Micro**, vol. 24, no.6, pp. 10-20, 2004.
[8] S. Das et al., "RazorII: In situ error detection and correction for PVT and SER tolerance", in IEEE Journal of Solid-State Circuits,vol. 44, no. 1, pp.32-48, 2009.
[9] S. Rehman, "Reliable Software for Unreliable Hardware – A Cross-Layer Approach", PhD Thesis, (150715/01), KIT, Karlsruhe, 2015.
[10] K. Kang et al., "NBTI induced performance degradation in logic and memory circuits: how effectively can we approach a reliability solution?", in **ASPDAC**, 2008.
[11] D. B. Limbrick,., et al. "Reliability-aware synthesis of combinational logic with minimal performance penalty." IEEE **TNS** 60.4 (2013): 2776-2781.
[12] T. Gu et al., "Analyzing and Mitigating the Impact of Permanent Faults on a Systolic Array Based Neural Network Accelerator." preprint arXiv:1802.04657 (2018).
[13] Zhang, Jeff, et al. "ThUnderVolt: Enabling Aggressive Voltage Underscaling and Timing Error Resilience for Energy Efficient Deep Neural Network Accelerators." IEEE **DAC**, 2018.
[14] M. Shafique et al., "An overview of next-generation architectures for machine learning: Roadmap, opportunities and challenges in the IoT era", IEEE **DATE**, 2018, pp. 827-832.
[15] N. Papernot et al., "Towards the science of security and privacy in machine learning." preprint arXiv:1611.03814 (2016).
[16] J. Gao et al., "PANDA: Facilitating Usable AI Development", preprint arXiv:1804.09997 (2018)..
[17] E. Hesamifard et al., "Privacy-preserving Machine Learning as a Service", Proceedings on Privacy Enhancing Technologies, vol. 3 (2018): pp. 123-142.
[18] Hesamifard et al., "CryptoDL: Deep Neural Networks over Encrypted Data." preprint arXiv:1711.05189 (2017).
[19] F. J. González-Serrano et al., "Supervised machine learning using encrypted training data." International Journal of Information Security (2017): pp. 1-13.
[20] Y. Liu et al., "Less is More: Culling the Training Set to Improve Robustness of Deep Neural Networks", preprint arXiv:1801.02850 (2018).
[21] J. Steinhardt, et al., "Certified defenses for data poisoning attacks", **NIPS**. 2017,
[22] G. Katz et al. "Reluplex: An efficient SMT solver for verifying deep neural networks", **CAV**. Springer, 2017, pp. 97-117.
[23] L. Kuper et al. "Toward Scalable Verification for Safety-Critical Deep Networks", preprint arXiv:1801.05950 (2018).
[24] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. IEEE **CVPR,** pp. 248-255.
[25] Vedaldi, A., & Lenc, K. (2015, October). Matconvnet: Convolutional neural networks for matlab. ACM Multimedia, pp. 689-692.
[26] P. N. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit." **ISCA**. ACM, 2017, pp. 1-12.
[27] A. Parashar et al., "Scnn: An accelerator for compressed-sparse convolutional neural networks." **ISCA** ACM, 2017, pp. 27-40.
[28] M. A. Hanif et al., "Error resilience analysis for systematically employing approximate computing in convolutional neural networks." IEEE **DATE**, 2018.
[29] M. Shafique et al. "Exploiting program-level masking and error propagation for constrained reliability optimization." IEEE **DAC**, 2013, pp. 1-9.
[30] F. Kriebel et al. "ASER: Adaptive soft error resilience for reliability-heterogeneous processors in the dark silicon era." IEEE **DAC**, 2014, pp. 1-6.
[31] D. Gopinath et al., "Deepsafe: A data-driven approach for checking adversarial robustness in neural networks", preprint arXiv:1710.00486 (2017).
[32] X. Huan et al., "Safety verification of deep neural networks", **CAV**. Springer, 2017.
[33] T. Hunt et al., "Chiron: Privacy-preserving Machine Learning as a Service", preprint arXiv:1803.05961 (2018).
[34] L. Wei et al., "I Know What You See: Power Side-Channel Attack on Convolutional Neural Network Accelerators", preprint arXiv:1803.05847 (2018).
[35] Q. Sun et al., "Natural and Effective Obfuscation by Head Inpainting", preprint arXiv:1711.09001 (2017).
[36] Rosenberg et al., "DeepAPT: nation-state APT attribution using end-to-end deep neural networks." ICANN. Springer, 2017, pp. 91-99.
[37] LeCun, Yann. "LeNet-5, convolutional neural networks." URL: http://yann.lecun.com/exdb/lenet (2015): 20.
[38] LeCun, Yann et al., http://yann.lecun.com/exdb/mnist/, May 2018.
[39] T. Li et al., "Outsourced privacy-preserving classification service over encrypted data", Journal of Network and Computer Applications (2018).
[40] X. Liu et al., "Privacy-Preserving Outsourced Support Vector Machine Design for Secure Drug Discovery." IEEE **TCC** (2018).