# An Overview of Next-Generation Architectures for Machine Learning: Roadmap, Opportunities and Challenges in the IoT Era

Muhammad Shafique[*], Theocharis Theocharides[‡], Christos-Savvas Bouganis[†], Muhammad Abdullah Hanif[*], Faiq Khalid[*], Rehan Hafiz[§], Semeen Rehman[*]

[*] Vienna University of Technology, Vienna, Austria

[‡]University of Cyprus, Nicosia, Cyprus

[†]Imperial College London, London, UK

[§] Information Technology University, Lahore, Pakistan

***Abstract*— The number of connected Internet of Things (IoT) devices are expected to reach over 20 billion by 2020. These range from basic sensor nodes that log and report the data to the ones that are capable of processing the incoming information and taking an action accordingly. Machine learning, and in particular deep learning, is the de facto processing paradigm for intelligently processing these immense volumes of data. However, the resource inhibited environment of IoT devices, owing to their limited energy budget and low compute capabilities, render them a challenging platform for deployment of desired data analytics. This paper provides an overview of the current and emerging trends in designing highly efficient, reliable, secure and scalable machine learning architectures for such devices. The paper highlights the focal challenges and obstacles being faced by the community in achieving its desired goals. The paper further presents a roadmap that can help in addressing the highlighted challenges and thereby designing scalable, high-performance, and energy efficient architectures for performing machine learning on the edge.**

***Keywords*— *Deep Learning, Convolutional Neural Networks, IoT, Machine Learning.***

## I. INTRODUCTION

Market analysts forecast that almost 10 billion connected devices will be utilized by 2018, and that by 2020, this number will surpass 20 billion [1]. These devices collect and aggregate volumes of data, and in doing so, they are revolutionizing our society in multiple ways; from healthcare, to social networks, to consumer electronics and many more [2]. To process these immense volumes of data, machine learning, and in particular, deep learning via convolutional neural networks, is emerging as the de facto analysis tool, that powers several aspects of our Big Data society. Applications spanning from infrastructure (smart cities, intelligent transportation systems, smart grids, to name a few), to social networks and content delivery, to e-commerce and smart factories, and emerging concepts such as self-driving cars, are powered by machine learning technologies. Some applications require analytics and trends identification; an abundance of emerging systems however, especially systems within the IoT paradigm, require real-time action, based on the data, and typically rely on low-power hardware, and limited connectivity and bandwidth. As such, computation shifts from the datacenters and the cloud, to the fog and on the edge; near-sensor computation and near-sensor intelligence are starting to emerge as necessities, in order to continue supporting the paradigm shift of our connected world. This need is further emphasized, as deployment of sensors and edge devices is typically done over low-bandwidth connectivity, and thus, communication between the edge/sensor and the cloud/fog is constrained both in terms of bandwidth but also in terms of latency.

Deep Convolutional Neural Networks (CNNs), the core computing paradigm of these systems, has received significant attention in terms of building systems and architectures which are geared to extract maximum performance, minimum energy, and maximum reliability. Unlike traditional computing applications, CNNs are complex structures that dynamically redirect their computational flow depending on the input data. Furthermore, there are several steps that are required prior to processing input data that involve among others dimensionality reduction algorithms, filtering, etc. As such, traditional Von Neumann architectures typically employed in modern datacenters, are not sufficient [3], primarily because of limitations in both performance and power efficiency [3]. Thus, alternative platforms, led mainly by general purpose graphics processing units (GPGPUs) have been sought by academia and industry due to their higher degree of parallelism and availability of multiple convolution kernel support through the large number of computing cores embedded in modern GPUs. While GPGPUs have been indeed successful in driving the industrial push, their energy requirements are not in line to make them a suitable compute accessory for resource-constrained IoT paradigm. Furthermore, due to the connected nature of IoT sensors, security, reliability, safety, privacy, and trust-worthiness are also desirable features for the required compute paradigm. There are several industry and academia initiatives towards custom architectures and systems built to support the diverse requirements of Machine Learning applications. Among others, neuromorphic and bio-inspired approaches have received significant attention [1], but industrial emphasis has been placed in custom-built architectures, including among others, cognitive-based architectures such as IBM's Watson [4], Microsoft's Brainwave accelerated with FPGAs [5], and Google's Tensor Processing Architecture [6] which is designed for high-volume of low-precision computations, thus focusing mostly on satisfying the computational demands of modern ML algorithms. However, such systems are not yet suitable for near-sensor processing and edge computing. Recent focus therefore, has shifted in designing low-power, low-bandwidth architectures, that facilitate near-sensor intelligence, shifting a significant workload within the sensor, or limited within the fog (i.e. edge computing). Emerging examples include NVIDIA's Jetson architecture that maintains NVIDIAs machine learning ecosystem, but targets low-power embedded devices. Over the last few years, there have been significant pushes from both industry and academia to integrate machine learning in resource-constrained environments, primarily by domain-specific architectures, but also by designing custom digital and analog CMOS circuits, as well as post-CMOS technologies such as memristors. Further, optimization techniques such as weight quantization, network pruning, autoencoders (Diabolo networks) [2], and other approaches are the current driving force in targeting near-sensor and edge scenarios.

This paper, presents an overview of the challenges and opportunities of mapping machine learning applications to resources

constrained environments, identifying several areas where current state-of-the-art approaches are still lagging. The paper identifies several challenges such as power and energy efficiency, performance under resource-constrained scenarios, reliability, safety and security, and also the social aspects attached with the integration of artificial intelligence within our daily activities. Further, the paper identifies several opportunities in both software and hardware approaches, addressing optimization opportunities such as quantization and the use of approximate computing paradigms, and also the opportunities derived from new technologies and architectures. Further, it presents our view of the ecosystem that will enable deep learning within the existing IoT infrastructure, both in terms of the design of such systems, but also for the tools and methodologies necessary to achieve a holistic intelligent IoT world. Lastly, the paper presents our anticipated roadmap of how all these challenges and opportunities blend with existing trends in deep learning architectures for the IoT domain.

## II. CHALLENGES

In the following, we highlight few of the most important challenges being faced by the system design community in proposing intelligent, reliable, and resource efficient systems for IoTs.

### A. Power and Energy Efficiency

The performance of Deep Convolutional Neural Networks in ImageNet [9] challenge triggered an avalanche of research in the area of deep neural networks. The key reason behind their success and widespread adaptation across the complete spectrum of AI applications is their ability to learn high level features directly from raw sensory data without being explicitly programmed [10].

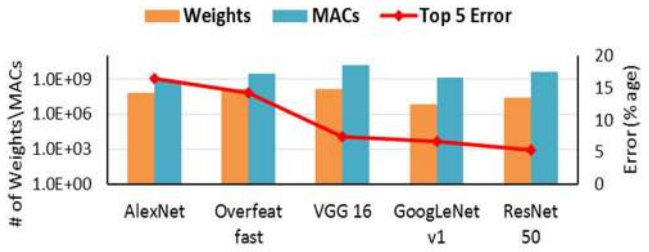


Figure 1: Number of computations and weight parameters for the popular state-of-the-art neural network architectures along with their respective accuracy on ImageNet dataset. The accuracy is reported in terms of Top 5 Error [10]-[14].

State-of-the-art neural networks have shown remarkable potential in almost all the AI applications. However, this enhanced performance usually comes at the cost of high computational and memory requirements. Figure **1** highlights the resource requirements of few of the state-of-the-art neural networks proposed for image classification. It can be observed from figure that for all the stated networks, the number of parameters required to represent the model (i.e., weights) are well over 1 million and the number of MAC computations are either around or well over the 1 billion mark. In order to highlight the effect of such huge computational load on the energy/power efficiency, it is worth elaborating that each MAC operation is usually coupled with a number of memory access, e.g., *considering no data re-use (i.e. the worst case scenario)*, each operation requires three memory reads and one memory write, which in case of even the simplest of the mentioned DNNs (i.e., AlexNet [10]), would result in 2172M memory read and 724M memory write operations. Note that all the aforementioned factors, i.e., memory storage, memory accesses, and computations consume a significant amount of energy and power and thereby render them unsuitable for resource constraint and battery driven devices.

**Computational efficiency gap:** In order to illustrate power and energy requirements of current IoT devices we take an example of a SoC used for capturing and analyzing images for people with low vision [54]. The device proposed in [54] is based on a quad core ARM processor and a Qualcomm Adreno GPU. It is programmed to capture an image every 30 seconds and analyze it using multiple deep learning models like object detection, face recognition, visual scene recognition, etc. For the assumed workload, the device is reported to maintain a battery life of nearly 17 hours. Although the specified frame rate is reasonable for a number of applications, it is not suitable for various other real-time applications like object tracking. Now, to emphasize on the energy requirements, assuming the same battery capacity and energy per computation, if we increase the number of frames that are required to be processed from 2 per min. to 2 per second the device will run only for 17 mins while in case if we increase the fps to 30 per second the battery will last only for 68 seconds

To further emphasize on the power and energy efficiency of the state-of-the-art, Figure 2 highlights few of the competitions where AI defeated professional humans at different strategic/IQ games. It can be observed from the figure that even though the competitions resulted in favor of the AI machines the amount of power consumed by the android systems is orders of magnitude larger than what is believed to be the peak operating power of a human brain (i.e., 20W). Therefore, one of the key challenges is to bridge the gap between the efficiency of intelligent devices and a human brain to enable the use of such devices in almost all aspects of the daily life.

### B. Performance

Apart from the power and energy requirements, latency is another parameter which is vital for safety-critical applications like autonomous driving, where even a fractional delay in processing can trigger a catastrophic event. To highlight the performance issues being faced in IoT devices for simulating state-of-the-art deep convolutional neural networks, we consider an example of real-time object detection using NVIDIA Jetson TX1, which is considered to be a suitable candidate for edge processing, i.e., near sensor processing. Table 1 shows the frame rate achieved for different variants of YOLO algorithm (a state-of-the-art algorithm for real-time object detection) when simulated on NVIDIA Jetson TX1. It can be observed from the table that none of the illustrated models meet the real-time frame rate mark of 30 fps, when simulated on NVIDIA Jetson TX1 and even the most optimized of the illustrated, i.e., the Fast YOLO, is able to achieve only 17.85 fps. It is also worth highlighting that the algorithms which offers higher fps have slightly lower accuracy than other state-of-the-art. Therefore, there is significant need for improving the execution time of the state-of-the-art deep neural networks for resource constraint devices.

Table 1: Results of a few state-of-the-art object detection algorithms on Pascal VOC2007 dataset. Accuracy is mentioned in mean Average Precision (mAP)/Intersection of Union (IOU).

| Framework | fps (NVIDIA Jetson TXI) | IOU/mAP. |
|---|---|---|
| Fast YOLO | 17.85 [52] | 52.7 [53] |
| O-YOLOv2 | 11.8 [52] | 65.1 [52] |
| YOLOv2 | 5.4 [52] | 67.2 [52] |

Latency of a neural network is usually dependent upon a number of factors, e.g., the number of interdependent layers in a network, the total number of neurons/computations per layer, latency of memory accesses, number of memory accesses, and latency of the computing modules (i.e., adders and multipliers). In most of the cases the intra-layer independencies are exploited using multi-/many-core systems, which consumes more power and leads to thermal issues as well. Other factors like latency of memory accesses and latency of computational modules can be improved by using specialized application specific hardware. In the context of the aforementioned discussion, the following are the key aspects for improving the overall performance of a neural network:

1. Exploring alternate neural network structures that provide better accuracy while consuming lesser resources.

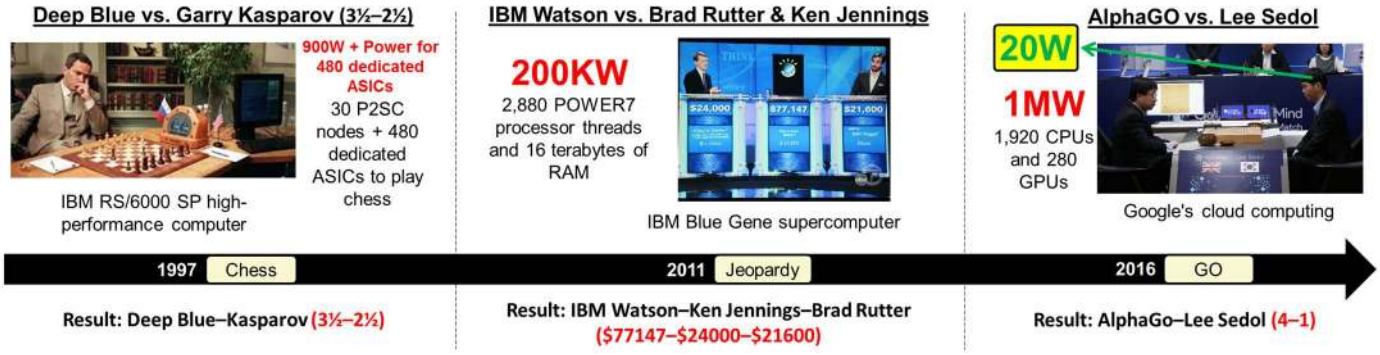


Figure 2: An illustration of the advancements in AI and the overall efficiency gap between a human brain and the computational devices [38]

2. Designing specialized hardware accelerators that can maximize latency by exploiting data reuse and the use of low-latency hardware modules.

### C. *Reliability*

Reliability is another core issue which has not been in the spotlight for neural networks. Although, neural networks themselves are not entirely accurate, there is still a dire need to use reliable platforms for sensitive applications, e.g., autonomous vehicles, forex predictions, cancer detection, etc. in order to ensure as much accuracy and reliability as possible. Alongside this, deep learning is readily being adapted in a wide range of industrial applications involving data mining & analytics, where it is usually required to meet certain industrial standards like IEC 61508 [31], which provides reliability specifications for a range of industrial applications. Therefore, it is essential to ensure reliability in deep learning based systems as well.

Conventional methods for ensuring reliability includes redundancy based approaches like TMR (Triple Modular Redundancy) and DMR (Double Modular Redundancy), which are not resource efficient. Therefore, in order to introduce reliability in a resource efficient manner following are the key challenges being faced by the system design community:

1. Identification of sensitive neurons/layers in which errors can significantly affect the accuracy of the system.
2. Identification of sensitive hardware modules of underlying hardware used for simulation.

### D. *Security Vulnerabilities in Machine Learning*

Due to rapid advancement in computational capability of hardware, machine learning tools, specially, neural networks are becoming more popular in handling the processing of large datasets. These tools perform exceptionally well as compared to the conventional ML algorithms but possess inherent security vulnerabilities which can be misused to manipulate them [16]. In machine learning, firstly, a model is trained and validated based on the available dataset, and then the obtained trained model is utilized for inference, as shown in Figure 3. As the process involves many data dependencies and complex computational process, it makes it vulnerable to several security attacks, i.e., data poisoning during training and inference stages. Thus, the above attacks can cause catastrophic impact on the performance, accuracy and reliability of the deployed machine learning algorithm. Typically, based on the attacker's goal, these security attacks can be divided into the following four categories [17]:

1. Confidence Reduction: This attack introduces the ambiguity in classification to reduce the confidence level of output classes.
2. Random Misclassification: This attack changes the output classification to a random output class different from original class.
3. Targeted Misclassification: This attack produces the inputs that force the output classification to a specific target class.
4. Source/Target Misclassification: This attack forcefully alters the output classification of a specific input class to specific target class.

However, the strengths and weaknesses of the above-mentioned security attacks depend upon the attacker's access and capability to

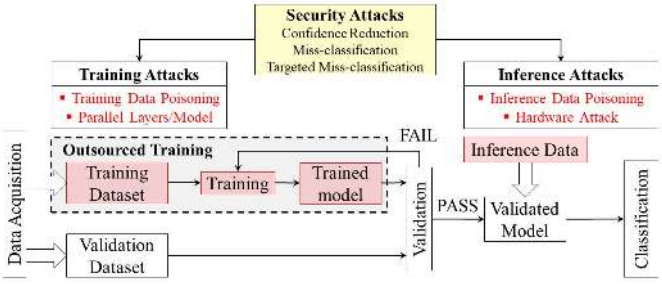


Figure 3: Security Vulnerabilities in Machine Learning (Neural Networks)

hardware implementation, which in combination with attack types can be referred as an attack surface. For example, the attack surfaces shown in Figure 4 and Figure 5, depict the strength and difficulty level of security attacks during the training and inference stages, respectively. Therefore, the following subsections discuss possible security vulnerabilities and state-of-the-art security attacks during the training and inference stages of a machine learning system design.

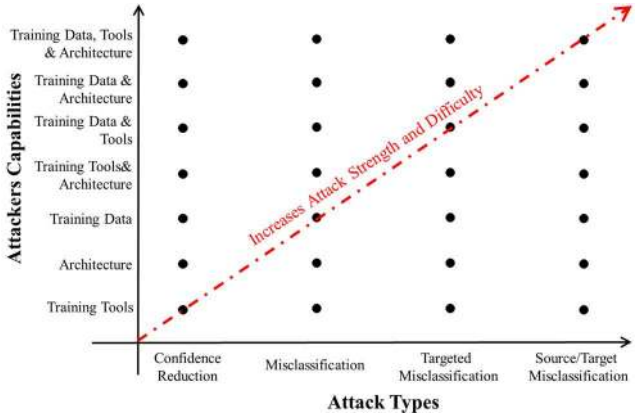


Figure 4: Attack Surface for Training Attacks

#### 1) *Training*

During the training stage, the model parameters are learned using an available dataset which is assumed to capture the input space of the system. However, for larger datasets, it is not always feasible to train a machine learning algorithm locally, especially neural network, because

of limited resources and high non-recurring engineering (NRE) cost [18]. Therefore, in most of the cases, either training is outsourced or transfer learning is used, which increases the possibility of security attacks. Though, multiple security attacks are possible in both the inference and training stages, the training is the most vulnerable stage, especially for outsourced training. The security attacks during training are highly data dependent and their effectiveness is measured based on the attack types and knowledge of training data, tools and the network architecture. For example, in Figure 4 the most powerful attack is when attacker has all the information about training tools, data and algorithm architecture, thus can alter the output classification of a specific input class to target class. However, this attack is relatively difficult to realize: 1) in case of large datasets because of the required amount of data manipulation and 2) without having significant impact on the overall accuracy of the network.

Several attacks are proposed to report the vulnerabilities in machine learning tools, especially in neural networks. Most of them are focusing on training data poisoning to launch different types of attacks, e.g., confidence reduction [19], random [20], targeted [20][21] and source/target misclassifications [20][21]. Similarly, other attacks are based on the architectural modification that can perform targeted [22] and source/target misclassifications [23]. Although, these attacks are very effective but these attacks have some effects on the inference accuracy which reduces their effectiveness. Therefore, there are many alternatives which can be explored to generate an effective attack without reducing the inference accuracy. For example, if training data is modified based on the architecture knowledge or vice versa, then the effectiveness of such attacks can be improved. Similarly, if adversary has access to manipulate the training tools, dataset and the architecture then it can improve effectiveness and reduce its detection chances at inference stages. Therefore, in outsourced training several prevention techniques have been proposed and one of the most commonly used is encrypting the training data before outsourcing [24].

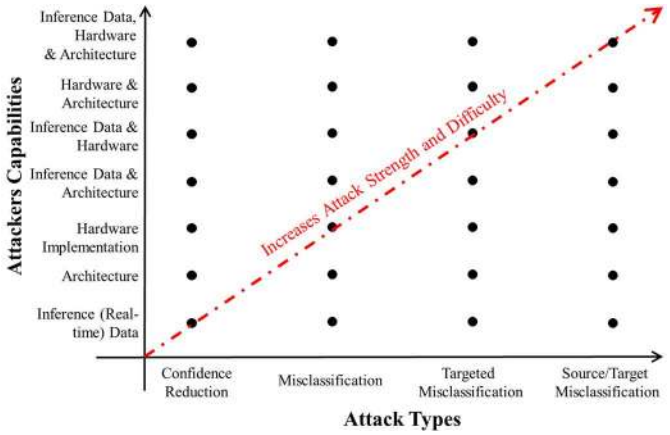


Figure 5: Attack Surface for Inference Attacks

### 2) *Inference*

The inference stage in machine learning algorithms is also vulnerable and data dependent. Unlike the training stage, the data poisoning attacks in inference stage are not effective because a sophisticated preprocessing stage can significantly reduce the perturbations in real time data. Thus, inference data poisoning based attacks are the weakest and easy to prevent at inference stages, as shown by the attack surface in Figure 5. However, highly correlated perturbations during the inference stage can be generated by combining the inference data poisoning with training stage poisoning. Therefore, the possibility of such attacks cannot be ignored because preprocessing stage can overlook very high correlated perturbations. For example, Sharif et al. proposed an attack which introduces a glasses shape perturbation in training dataset and then it uses this knowledge to launch a misclassification attacks on neural network based face recognition system. Alternatively, attacker can exploit the data acquisition block to add data of specific class at the periodic or at specific time intervals to misclassify the stream of inference dataset [25]. Similarly, the other possible attacks during the inference stage are to exploit the architecture or its hardware implementations based on the hardware or architecture access [26]. Although, such attacks are not easy but in most of the cases, trained models run at third part hardware which can have the multiple dormant or active intrusions. These intrusions can exploit the computations of trained model to launch multiple attacks [27].

Therefore, based on the above discussion the following challenges are extracted to ensure security and privacy in machine learning based classification systems.

1. How to ensure protection of the training dataset and its corresponding labels, especially for outsourced training and transfer learning?
2. How to the secure the data acquisition during the inference stage?
3. How to detect and prevent the highly correlated data perturbations in pre-processing stage.
4. How to detect and prevent the dormant and active intrusions in third party hardware accelerators of machine learning algorithm?

## E. *Social Aspects and Social Integrations*

While artificial intelligence is already revolutionizing our daily lives, it is still considered a taboo item in our society, and in fact, several scholars and pioneers advocated against its use[51]. We firmly however believe that the main reasons for this, are the aforementioned challenges in reliability and security, as well as the associated ethics issues. Given that reliability and security are within our grasp as academics and practitioners, we need to ensure that such technology meets both the industrial standards as well as governmental regulations, in terms of security, privacy, safety and reliability. We therefore are obliged to maintain the highest standards in designing all such devices, tools, algorithms and methodologies to ensure that society benefits from the augmented services and quality of life that this revolution is accompanied with.

# III. Current Trends

Most, if not all, state-of-the-art Neural Networks are over-parameterized. To alleviate the overall computational requirements of NNs for inference stage a number of optimization techniques have been proposed which span all the way from software to hardware level. Figure 6 shows the set of optimization techniques that can be employed while Figure 8 illustrates the general sequence in which they are currently being adapted [32][33]. In the forthcoming subsections, we provide an overview of the state-of-the-art optimization techniques and the hardware accelerators in section III-A and III-B, respectively.

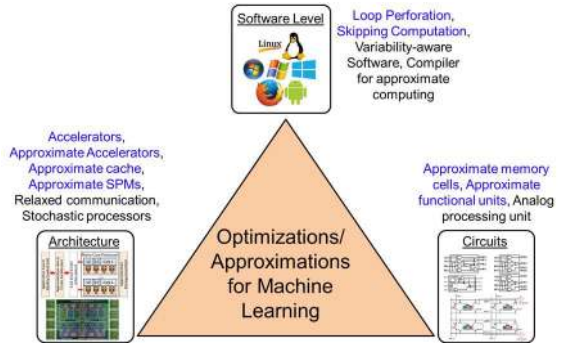


Figure 6: Possible optimization/approximation knobs for improving power, energy, performance, and area of underlying ML hardware.

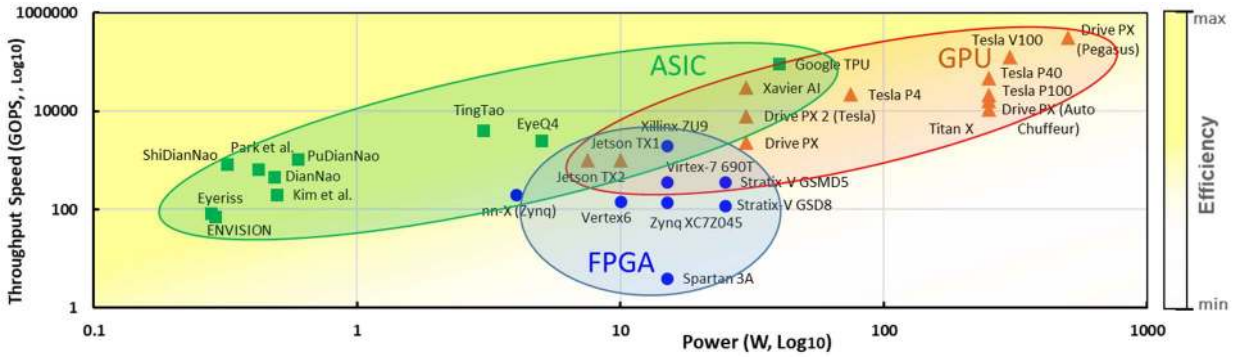


Figure 7: Performance comparison between possible hardware platforms for realizing machine learning algorithms. [7][40]-[50]

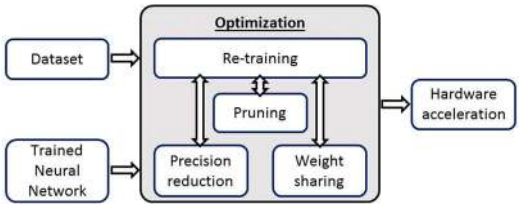


*Figure 8: Available software and co-design optimization strategies for improving power and performance efficiency of ML architectures.*

### *A. Optimizations*

Early Convolutional Neural Networks such as AlexNet and VGG exhibited uniform layer connectivity and a large number of parameters (i.e. weights). Over time, deeper NN architectures such as GoogLeNet, ResNet, and DenseNet provided improved accuracy results by employing a larger number of layers and introducing novel computational structures departing from the uniform layer connectivity of the early DNNs. At the same time, in an effort to reduce the memory footprint and computational workload of the NN models, significant effort was placed on the pruning of the networks, and the quantization of weights and activation functions. Current results show that a significant reduction in both computational load and memory footprint can be achieved with a small impact on the accuracy of the classification for even up to 8 bits of quantization in certain applications. Currently, research effort is placed into ternary and even binary NN for improved performance with limited impact on the quality of the algorithm.

### *B. Current Architectural Trends*

The architecture evolution of the devices for Deep Learning was coupled with the algorithmic changes introduced in the Machine Learning community. Early works focused on the utilization of GPGPUs as an accelerator for the main computational part of the CNNs, as their SIMD structure and floating-point support was matching well the computational requirements of the early CNN structures. As the CNNs became less regular, and optimizations were introduced in the employed number representation, the research effort was moved to the design of custom devices tailored on the new requirements.

In the server-graded space, the GPGPU devices were evolved and began to contain hardened blocks for Machine Learning applications such as the Nvidia Tesla V100 with Tensor Core technology for the acceleration of AI workloads. Moreover, new ASICs tailored for Machine Learning workloads started to appear such as the Tensor Processing Unit (TPU) from Google, providing inherent support for fixed-point calculations and later also support for floating point calculations, and the Lake Crest chip from intel.

In the embedded systems space, specialized ASICs have appeared with emphasis on low power such as the Myriad 2 chip from Movidius, and a number of projects from academia such as Eyeriss [7], and Cnvlutin [8]. In parallel to the above approaches, effort has also focused on Field-Programmable Gate Arrays, which are reprogrammable ASICs that can be customized for a specific CNN instance providing high performance with respect to the CPU and GPU counterparts under a power budget.

Towards the easy programmability of the above systems, a number of SW tools have been developed that can help researchers and practitioners to map CNNs on the above devices without extensive knowledge of the underlying architecture, and at the same time to obtain highly-performing systems. Examples include Intel's Deep Learning Framework for targeting Intel devices, the cuDNN library from Nvidia for targeting cuda-enabled GPUs, the reVision stack from Xilinx for targeting FPGAs, where a number of tools have also appeared from academia such as fpgaConvNet [28], FP-DNN [29], and FINN [30].

Figure 7 provides a summary of efficiency and the computational capability of various state-of-the-art hardware platforms and machine learning accelerators. It can be observed from the figure that application specific hardware provides the best efficiency in terms of computations per watt. However, the state-of-the-art high-end GPUs provides highest computation capability which when coupled with their easy programmability makes them a highly suitable candidate for the development stage. The region of efficiency, in terms of computations per watt, is also highlighted in the figure, which is expected to be covered by future neuromorphic devices.

## IV. Future Directions and Roadmap of Deep Learning For IoTs

### *A. Alternate Neural Network Structures*

As recently highlighted by Sabour et al. in [34], the state-of-the-art neural networks have less levels of structure. However, systems that possess the capability of solving complex problems are known to have a well-defined hierarchy. Such system are also expected to be robust to small changes. For example, in case of image classification, changes in the orientation of an image like tilt and rotation usually leads to misclassification in conventional CNN based systems. However, using nested sets of layers [34], it is possible to learn more complex patterns and to design highly robust systems. These alternate neural network architectures are also expected to have lesser number of parameters as compared to the state-of the-art. This can be highlighted using the fact that GoogLeNet v1, which is constructed using relatively well-structured inception modules, uses lesser number of parameters than VGG for providing better classification accuracy. Thus, in the light of the above discussion, there is a significant need for improving the structure of neural networks in order to build highly intelligent and resource efficient systems.

### *B. Approximate and Near Threshold Computing*

Approximate and near threshold computing are the emerging computing paradigms for designing ultra-low power computing devices. The core principle of these paradigms lies within improving area, power, performance, and/or energy at the cost of reduced quality. Although these paradigms have shown remarkable potential in

designing resource efficient systems for multimedia applications, they have yet to show their potential in designing highly resource efficient ML systems for in-/near-sensor computing in IoTs.

### C. *Emerging Technologies*

With Moore's law reaching its end at a rapid pace, emerging technologies like Nano-wires, Memristors, Spintonic devices [37], etc. are anticipated to be the beginning of a new era of computing. These technologies are expected to offer highly resource efficient solutions which can be useful for improving the computational capability of edge-devices under a defined energy/power or performance constraint. Nano-wires based 3D-stacked architectures [36] are an example of such futuristic devices, where a sea of accelerators are connected with a high-speed dense memory through high-bandwidth nano-wires. These technologies are particularly helpful for designing neuromorphic devices that are based on digital, analog, or analog and digital mixed circuits.

### D. *Distributed Learning and Inference*

Distributed computing is also getting popular for simulating neural networks in IoTs based systems [35]. The main concept behind distributed computing in IoTs is to enable fine-grained energy-/latency-aware distribution of the computations across the complete stack of the IoT system, i.e., starting from the edge all the way to the cloud. This is usually done in order to optimize the overall processing time or the energy consumption of in-/near-sensor devices. Such algorithms when coupled with the aforementioned developments can significantly be beneficial for achieving optimal/near-optimal improvements in the overall performance and efficiency of the systems.

## REFERENCES


[1] Gartner Inc., https://www.gartner.com/newsroom/id/3598917, Available Online, November 2017.

[2] LeCun, Y., et al. (2015). Deep learning. Nature, 521(7553), 436-444.

[3] Chen, Y., et al. (2017). Hardware for machine learning: Challenges and opportunities. 2017 IEEE Custom Integrated Circuits Conference , 1-8.

[4] David A. Ferrucci. 2011. IBM's Watson/DeepQA. In Proceedings of the 38th annual international symposium on Computer architecture. ACM, USA

[5] Caulfield, A. M., et al. (2016, October). A cloud-scale acceleration architecture. In Microarchitecture (MICRO), (pp. 1-13). IEEE.

[6] K. Sato, et al., "An in-depth look at Google's first Tensor Processing Unit (TPU)", available online, https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu, November 2017.

[7] Y.-H. Chen et al., "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in ISCA, 2016.

[8] J. Albericio et al., "Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing," in ISCA, 2016.

[9] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International Journal of Computer Vision 115.3 (2015): 211-252.

[10] Krizhevsky, Alex, et al.. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[11] Sze, Vivienne, et al. "Efficient processing of deep neural networks: A tutorial and survey." arXiv preprint arXiv:1703.09039 (2017).

[12] K. He, et al., "Deep Residual Learning for Image Recognition," in CVPR, 2016.

[13] K. Simonyan et al., "Very Deep Convolutional Networks for Large-Scale Image Recognition," in ICLR, 2015

[14] C. Szegedy et al., "Going Deeper With Convolutions," in CVPR, 2015.

[15] (Accessed on 13th Nov 2017.). URL https://github.com/jcjohnson/cnn-benchmarksAccessedon13thNov2017.

[16] Melis, Marco, et al. "Is deep learning safe for robot vision? adversarial examples against the icub humanoid." arXiv preprint arXiv:1708.06939 (2017).

[17] Papernot, Nicolas, et al. "The limitations of deep learning in adversarial settings." Security and Privacy (EuroS&P), IEEE European Symposium on. IEEE, 2016.

[18] Zhang, Xinyang et al. "Modular Learning Component Attacks: Today's Reality, Tomorrow's Challenge." arXiv preprint arXiv:1708.07807 (2017).

[19] Papernot, Nicolas, et al. "Practical black-box attacks against machine learning." Asia Conference on Computer and Communications Security. ACM, 2017.

[20] Gu, Tianyu, et al.. "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain." arXiv preprint arXiv:1708.06733 (2017).

[21] Sharif, Mahmood, et al. "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition." Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016.

[22] Collobert, Ronan, et al. "A unified architecture for natural language processing: Deep neural networks with multitask learning." International conference on Machine learning. ACM, 2008.

[23] Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013).

[24] Gilad-Bachrach, Ran, et al. "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy." ICML. 2016.

[25] Hosseini, Hossein, et al. "Attacking Automatic Video Analysis Algorithms: A Case Study of Google Cloud Video Intelligence API." arXiv preprint arXiv:1708.04301 (2017).

[26] Liu, Yingqi, et al. "Trojaning Attack on Neural Networks." (2017). Department of Computer Science Technical Reports. Paper 1781.

[27] Liu, Yuntao, et al. "Neural Trojans." arXiv preprint arXiv:1710.00942 (2017).

[28] Stylianos I. Venieris et al. "fpgaConvNet: A Framework for Mapping Convolutional Neural Networks" on FPGAs. FCCM 2016: 40-47

[29] Y. Guan et al. "FP-DNN: An Automated Framework for Mapping Deep Neural Networks onto FPGAs with RTL-HLS Hybrid Templates," in FCCM, 2017.

[30] Y. Umuroglu et al. "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference," in FPGA, 2017.

[31] IEC 61508 2016. Functional Safety and IEC 61508. (2016). Retrieved Oct. 2016 from http://www.iec.ch/functionalsafety/

[32] Han, Song, et al. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." arXiv preprint arXiv:1510.00149 (2015).

[33] Han, Song, et al. "EIE: efficient inference engine on compressed deep neural network." Proceedings of the 43rd International Symposium on Computer Architecture. IEEE Press, 2016.

[34] Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic Routing Between Capsules." Advances in Neural Information Processing Systems. 2017.

[35] Eduardo Cuervo et al., "Maui: making smartphones last longer with code offload". Conference on Mobile systems, applications, and services, ACM, 2010.

[36] M. M. Sabry et al., "Energy-Efficient Abundant-Data Computing: The N3XT 1,000x," in IEEE Computer, vol. 48, no. 12, pp. 24-33, Dec. 2015.

[37] C-SPIN: http://cspin.umn.edu/ Accessed on 7th December 2017.

[38] Another Way Of Looking At Lee Sedol vs AlphaGo: https://jacquesmattheij.com/another-way-of-looking-at-lee-sedol-vs-alphago: Accessed on 7th December 2017.

[39] Du, Zidong, et al. "ShiDianNao: Shifting vision processing closer to the sensor." ACM SIGARCH Computer Architecture News. Vol. 43. No. 3. ACM, 2015.

[40] B. Moons, et al., "Envision: A 0.26-to-10 tops/w subwordparallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi," International Solid-State Circuits Conference (ISSCC), 2017, pp. 246–257.

[41] Chen, Tianshi, et al. "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning." ACM Sigplan Notices. Vol. 49. No. 4. ACM, 2014.

[42] K. He, et al. "Deep Residual Learning for Image Recognition," in CVPR, 2016.

[43] K. Simonyan et al., "Very Deep Convolutional Networks for Large-Scale Image Recognition," in ICLR, 2015

[44] C. Szegedy, W et al., "Going Deeper With Convolutions," in CVPR, 2015."

[45] Park, Seong-Wook, et al. "An energy-efficient and scalable deep learning/inference processor with tetra-parallel MIMD architecture for big data applications." IEEE transactions on biomedical circuits and systems 9.6 (2015): 838-848.

[46] Kim, Joo-Young, et al. "A 201.4 GOPS 496 mW real-time multi-object recognition processor with bio-inspired neural perception engine." IEEE Journal of Solid-State Circuits 45.1 (2010): 32-45.

[47] Nvidia Infrence Accelerator: http://www.nvidia.com/object/accelerate-inference.html Accessed on 7th December 2017.

[48] Liu, Daofu, et al. "Pudiannao: A polyvalent machine learning accelerator." ACM SIGARCH Computer Architecture News. Vol. 43. No. 1. ACM, 2015.

[49] Jouppi, Norman P., et al. "In-datacenter performance analysis of a tensor processing unit." International Symposium on Computer Architecture. ACM, 2017.

[50] Startix V FPGAs: https://www.altera.com/products/fpga/stratix-series/stratix-v/overview.html, Accessed on 7th December 2017

[51] Stephen Hawking, BBC News, http://www.bbc.com/news/technology-30290540, Accessed on 7th December 2017

[52] Shafiee, Mohammad Javad, et al. "Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video." arXiv preprint arXiv:1709.05943 (2017).

[53] [2] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[54] Mathur, Akhil, et al. "DeepEye: Resource Efficient Local Execution of Multiple Deep Vision Models using Wearable Commodity Hardware." (2017).