Edge Intelligence: Challenges and Opportunities of Near-Sensor Machine Learning Applications

Conference Paper · July 2018

DOI: 10.1109/ASAP.2018.8445118

George Plastiras
University of Cyprus

Christos Kyrkou
University of Cyprus

Edge Intelligence: Challenges and Opportunities of Near-Sensor Machine Learning Applications

George Plastiras[1,2], Maria Terzi[1], Christos Kyrkou[1], Theocharis Theocharides[1,2]
[1]KIOS Research and Innovation Center of Excellence
[2]Department of Electrical and Computer Engineering
University of Cyprus
Nicosia, Cyprus
{gplast01, terzi.maria, kyrkou.christos, ttheocharides}@ucy.ac.cy

***Abstract*—The number of connected IoT devices is expected to reach over 20 billion by 2020. These range from basic sensor nodes that log and report the data for cloud processing, to the ones on the edge, that are capable of processing and analyzing the incoming information and taking an action accordingly. Machine learning, and in particular deep learning, is the defacto processing paradigm for intelligently processing these immense volumes of data. However, the resource inhibited environment of edge devices, owing to their limited energy budget, and low compute capabilities, render them a challenging platform for deployment of desired data analytics, particularly in real-time applications. In this paper therefore, we argue that for a wide range of emerging applications edge intelligence is a necessary evolutionary need, and thus we provide a summary of the challenges and opportunities that arise from this need. We showcase through a case study regarding computer vision for commercial drones, how these opportunities can be taken advantage, and how some of the challenges can be potentially addressed.**

***Index Terms*—Edge Intelligence, Deep Learning, Machine Learning, Convolutional Neural Networks, Embedded Systems**

I. Introduction

Artificial intelligence is currently mostly facilitated using machine learning algorithms, such as Deep Neural Networks (DNNs), that is complex structures of hierarchical layers that build powerful representations from raw data. There are different types of DNNs such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). CNNs, for example, have emerged as the leading architecture for computer vision tasks, such as large scale image classification [1]. The key reason behind their success and widespread adaptation is their ability to learn high level features directly from raw sensory data without being explicitly programmed. This is applicable across a wide spectrum of Artificial Intelligence (AI) applications, with demonstrated remarkable results [2] at the cost of high computational and memory requirements. Due to the computational requirements of their learning, the CNNs are typically trained on GPUs [3]. As such, cloud computing has been used as a potential solution to mitigate the computation from the device to a remote computing infrastructure that has the necessary processing power. However, using the cloud as a centralized processing server increases the frequency of communication between user devices and the geographically distant data centers. This is limiting for applications that require real-time response, and therefore need computing resources and services very close to them, such as autonomous cars, for which delay or latency in transmitting vital information could be extremely dangerous. A connected car creates tones of megabytes of data per second including data regarding its mobility (such as routes), its operating conditions, images from its cameras, and data that describe its surrounding environment [4]. An autonomous car, will create even more data, estimated to be about one gigabyte per second [5]. Sending all the generated data to the cloud for processing will require prohibitively high network bandwidth and will increase response time.

Additionally, this is limiting for applications processing private data, such as wearable biomedical devices, or wearable cameras [6]. Processing private user data at the edge could protect user privacy and avoid exposing the data to security threats by sending them to the cloud. It is also worth mentioning that wearable devices do not guarantee connectivity, due to the location and movement of the wearer, and also require extremely low power operation emphasizing the need for intelligent processing on the edge. Hence, there has been a need for shifting/pushing the computation towards the edge of the network - processing data physically close to where the data is being produced, i.e., where the things and humans are, in the field area, homes, and remote offices- referred to as *edge computing* [7]. When the data is acquired, stored and processed with machine learning algorithms at the network edge (e.g embedded edge devices) we refer to this specifically as *edge intelligence* [8].

We argue, therefore, that edge intelligence improves time-to-action and reduces latency down to milliseconds while minimizing network bandwidth. Additionally, it can offer greater privacy and security than cloud computing and allow for greater control over data generated in foreign countries where laws may limit the use or permit unwanted governmental access to the data. Finally, edge intelligence lowers cost since more sensor-derived data are used locally and less data are transmitted remotely.

To better illustrate the difference between cloud- and edge-based systems we illustrate the example application of license plate recognition in Fig. 1. In the cloud paradigm the image

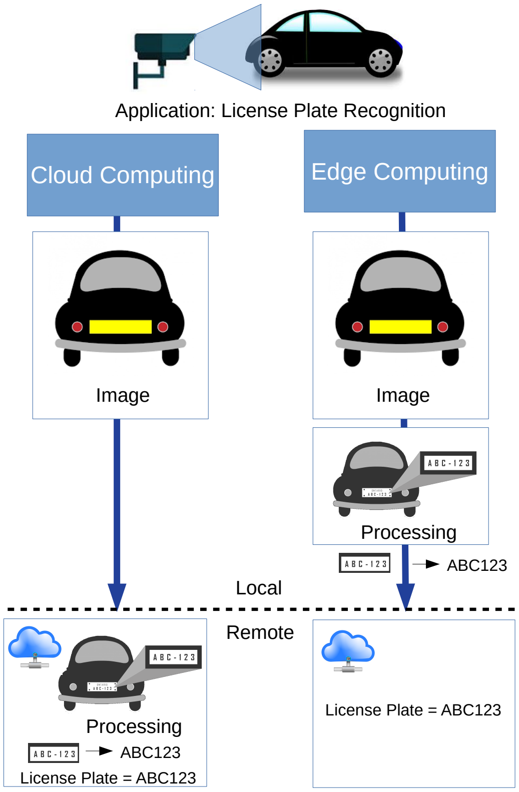

Fig. 1: Edge Vs Cloud Computing: Example Scenario of License Plate Recognition

captured by the camera needs to be transferred to the remote infrastructure for processing to first recognize the license plate is located in the image and then extract the number and store it in a database along with additional information such as time-stamp/location. In the edge paradigm all the aforementioned processing is done on the camera itself which only transmits the actual number and metadata information. Immediately, it is evident that the bandwidth requirements are reduced significantly and image data is not transmitted which alleviates privacy issues. Of course the challenge is on doing the whole processing efficiently on the camera.

We argue that edge intelligence is expected to arise in the up coming years and expand its reach into almost every domain. Therefore, in this paper, we attempt to identify and provide an overview of the challenges and opportunities it has to offer. We first identify several potential advantages that stem from pushing the intelligence to the cloud and also identify several challenges that need to be addressed such as power and energy efficiency, performance under resource-constrained scenarios, and storage. Further, we demonstrate how we addressed the above challenges in the adaption of a CNN detector, while maintaining the necessary performance for the particular application and utilising the opportunities edge intelligence has to offer.

II. Migrating Intelligence from Cloud to the Edge

In many applications the use of cloud computing is prohibitive due to various risks associated with the technical requirements or due to the type and nature of the information that is being processed. The nature of the risks of course, varies in different scenarios, depending among other things, on what type of cloud that is being employed whether that is a powerful workstation or a datacenter. However, in many cases these concerns are serious enough, that computation is mandatory to be shifted from the datacenter (cloud) to near the sensor (edge) [9]. In this section we discuss how important challenges of cloud computing can be addressed by shifting the acquiring, storing and processing of data at the edge.

A. Security/Privacy:

In any cloud computing paradigm, sensors which operate on specific devices collect data such as measurements, photos, videos, and location information. These data are transmitted from the sensor to the dedicated computing infrastructure with services that will perform the data analysis which can be either private or public [10]. These services can either be accessed through the internet or a dedicated communication infrastructure and the computing can range from a powerful workstation to thousands of servers.

Since the data leaves the device it can be exposed to various vulnerabilities and attacks such as penetration attacks (e.g., man-in-the-middle attack) resulting to theft of information or even Denial of Service attacks resulting to crashing servers or networks. Additionally, an attacker can not only access and intercept the sensor data, but also the application's processing outcome in transit to lead to a different action/scenario than the intended one (i.e. tampering). Additionally, the locality on the edge, as well as the potential proximity of system to end users can enable it to help address certain security challenges.

B. Performance

Performance, and latency in particular, is vital for time-critical applications where the signal roundtrip time can be very noticeable, such as self-driving cars. In such applications, the speed of decision making is of critical importance - even a fractional delay in processing can trigger a disastrous event. Therefore, migrating the computation to the edge provides a lot of opportunities to customization and specialization of the hardware of the device in order to reduce latency and power demands. In contrast to the datacenter which needs to handle a variety of workloads and the opportunities for customisation are limited. It is not a coincidence that in the last few years significant pushes have been made from both industry and academia towards to include domain-specific CMOS accelerators [11] for various machine learning algorithms in their Systems-on-chip. [12], [13], [14]

C. Bandwidth

Perhaps another important factor as to how edge intelligence can be beneficial is bandwidth savings. For instance, if you buy one surveillance camera, you could probably stream all of its footage to the cloud even at high resolutions. However, as the number of cameras increases the feasibility of this approach is questionable especially where the connectivity and bandwidth are limited. Furthermore, the cost of computation (e.g., both in renting and buying infrastructure) increases since the data also increases in volume with the same demands for processing time. On the other hand, if the cameras are able to process the video frames locally, detect targets and flag events, they can only save the important footage and discard the rest, leading to significant reduction in bandwidth demands.

D. Data Integrity

One final advantage of migrating the computation to the edge is that the data integrity is preserved due to its proximity to the source. Hence, there is no need to compress or change the data in any way which might have resulted in some loss of information (e.g., image/video compression). Also, any noise and signal degradation due to the communication infrastructure are alleviated.

III. Challenges and Opportunities for Edge-Computing

A. Power and Energy Efficiency

State-of-the-art AI techniques such as neural networks have shown remarkable promise in tackling a variety of different problems with impressive accuracy. However, this usually comes at the cost of high computational and memory requirements. Hence, in typical application scenarios today these neural networks run on powerful GPUs that dissipate a huge amount of power. On the other hand, embedded processors and DSPs offer an attractive low-power solution and benefit from fixed-point operations [15]. For practical deployment of neural networks on mobile devices, it is necessary to have low-complexity CNN models that can run on embedded processors. The algorithms need to leverage the fixed-point operations without compromising the accuracy. In addition, there is a significant need for improving not only the efficiency of the underlying operations performed by neural networks but also improve their structure and make them amicable for resource efficient systems.

B. Performance

Machine learning algorithms are characterized by extensive linear algebra operations as well as vector and matrix data processing. Traditional Von Neumann architectures are not optimized for such workloads and hence, specialized processing flow and parallel architectures are necessary to meet the low-latency requirements in safety-critical applications such as self-driving vehicles and Unmanned Aerial Vehicles (UAVs). Therefore, edge intelligence offers a clear opportunity for developing customized and specialized hardware to support the deployment of machine learning applications on the edge.

C. Memory Footprint

Edge intelligence devices have limited resources not only in terms of computation but also in terms of memory for storage and data access. Neural network and machine learning algorithms in general often require storing and accessing a number of parameters that describe the model architecture and weight values that form the classification model. Recent neural network architectures require accessing a vast amount of memory locations for every classification. Therefore, a significant challenge for deploying a machine learning algorithm on a resource constraint device is reducing the memory accesses and keeping the data local as to avoid costly reads and writes to the external memory modules (e.g. Data-reuse).

D. Current trends for Designing Neural Networks for Edge Intelligence

Recent techniques for designing and optimizing DNNs for edge intelligencce focus on reducing the computational and storage requirements by reducing the bitwidth precision of the network parameters and the processed data within the neural network [16]. Some works have even proposed using only binary values [17]. In an orthogonal approach to quantization aiming at reducing the computations needed by a DNN, other works propose to decompose the tensor operations of a neural network through low-rank approximations [18] or through separable filters [19], [20]. It is worth noting that the memory demands of a neural network is not only determined by the weight values and parameters but also by the working memory needed to store the intermediate results and values produced by the computations. Large networks may need hundreds of MBs of working memory [15]. Hence, some works have proposed to remove redundant information propagating through the network layers to reduce the on-chip memory needed for the produced results [21]. This can be beneficial for many edge intelligence applications such as video surveillance from static cameras where frame by frame changes are limited. Distillation, is another technique that trains a much smaller DNN to produce the outcomes of a resource hungry DNN to simultaneously reduce the computational cost while achieving the same classification accuracy [22]. Model compression is another method to reduce the network demands for memory and storage [23]. By using compression techniques such as Huffman encoding combined with quantization and pruning, it is possible to achieve significant reduction in the storage demands making it possible to store a model in the on-chip memory thus reducing the overhead of external memory accesses. From the analysis in the literature it is evident that the main benefactor towards more efficient operation of machine learning and neural networks in particular is the optimization of the model and structure. We show one such example on how to optimize a CNN specialized for aerial object detection that is suitable for on-board processing on a UAV using different embedded devices.

IV. Case Study

In this section we demonstrate how the edge intelligence challenges have been addressed for implementing a CNN-based object detector on an embedded edge device for Unmanned Aerial Vehicle (UAV) applications. This is particularly useful to applications such as search and rescue, emergency/disaster management, and intelligent transportation systems where there is a need for rapid deployment, fast response time, and in many cases limited connectivity and available infrastructure.

CNN-based object detectors have emerged as the leading architecture for many computer vision tasks [1], and therefore constitute a representative example for this case study. Through the case study we demonstrate how to select the best system architecture for given application constraints. The methodology, presented in this case study, is based on our previous work [24] which demonstrated that it is feasible to design an optimized CNN for object detection that can perform real-time on embedded devices and at the same time maintain high accuracy.

A. Deep-Learning based object detection

State-of-the-art methods for object detection cast the problem into a combination of regression and classification where they simultaneously predict the bounding box and class probabilities of the objects in an image [25]. YOLO [26] is an example of such a framework that was optimized for real-time processing achieving 30 frames-per-second(FPS) on a GPU such as Pascal Titan X. Existing detectors utilizing frameworks such as YOLO, primarily target high accuracy and run on power hungry devices, with large memories and storage capacity [25]. The underlying neural network structure of such detectors, consist of multiple layers and large number of filters in each layer which imposes significant challenges when using low-end embedded hardware. Thus, the feasibility of edge intelligence seems extremely challenging under these circumstances for resource-limited devices with power and size requirements, such as UAVs.

B. DroNet

DroNet [24] was proposed as an efficient Convolutional Neural Network(CNN) detector for real-time UAV Applications that provides real-time performance with minimal storage requirements on low-end embedded hardware without accuracy loss. The design of DroNet was based on Tiny-YOLO detector: a smaller model of YOLO detection that was optimized for fast inference on a GPU [27]. We explored the impact of changing the structure of that network (number of layers, filters, image size and the number of convolution and pooling layers), in order to achieve higher performance on a CPU rather than a GPU platform with minimal decrease in accuracy. DroNet can perform 35 frames-per-second and on an i5-8250U CPU processor around 60 frames-per-second on an Nvidia MX150, a top-end GPU for laptops.

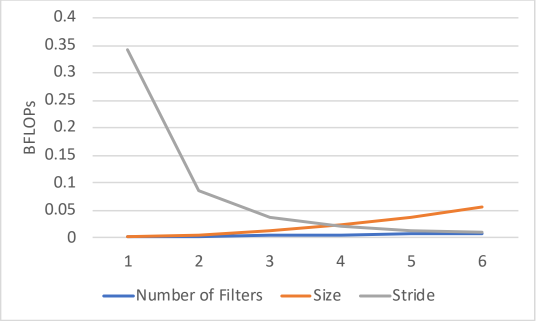

Fig. 2: Comparison with different number of layers, stride, size

1) Layer Structure: CNN-based object detectors consist of three types of layers, a convolutional layer, a pooling layer and a fully-connected layer. Each layer has a different number of filters, filter size and stride. Each layer has different requirements with respect to number of operations, measured in billion floating operations per second (BFLOPs). Figure 2 demonstrates how the number of filters, filter size and stride affects the number of BFLOPS produced for the first layer of the DroNet network. As shown in Fig. 2 by increasing the number of filters from one to six, we observe a small increase on the number of BFLOPs from $0.002(1)$ BFLOPs to $0.009(6)$ BFLOPs. By increasing the size of each filter from one to six we observe an increase from $0.002(1)$ BFLOPs to $0.057(6)$ BFLOPs. This shows a significant impact on the performance. On the other hand, stride can reduce the number of BFLOPs but also can lead to a higher reduction in accuracy. To find out the impact of stride on BFLOPs, we increased the number of filters and the size of each filter to $6(0.341$ BFLOPs) in order to have the maximum number of BFLOPs and increased the size of stride from 1 to 6 (Fig. 2). Increasing the size of stride lead to a significant reduction on the number of BFLOPs, around $34\times$.

Through the above discussion it is evident that by appropriately selecting the number of parameters and reducing the CNN network structure, we can reduce the computational requirements (e.g., power consumption) and the memory footprint, two important challenges of edge intelligence.

2) Input Size: Edge intelligence must address the computational, power and memory challenges while maintaining the accuracy of the implemented machine learning application. In this section, we demonstrate that the input image resolution affects both the accuracy and the performance of the CNN detector. Particularly, Fig. 3 shows the impact of the input size on performance (in terms of FPS) and on the accuracy of the CNN detector. As demonstrated in Fig. 3 the input size affects positively the accuracy and negatively the performance of the CNN detector - the larger input size on the network the higher the accuracy of the CNN detector and at the same time the lower the performance of the CNN detector. Moreover, larger input size, can also increase both memory and power requirements of the detector. Therefore, to maintain the good accuracy and performance of a machine learning application on an edge device one should take into account the input size and the impact on accuracy, FPS, memory and power consumption in order to choose the appropriate value.

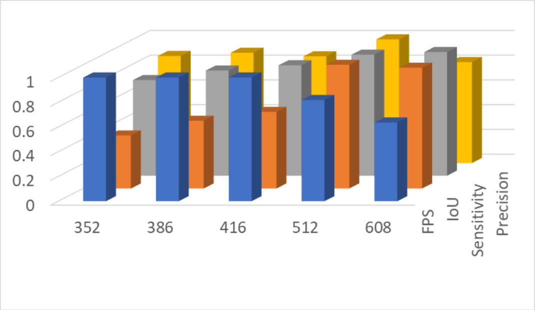

Fig. 3: DroNet Performance with different input sizes

3) Object Size: Another parameter that can affect the accuracy of the CNN detector is the size of the object that the detector was trained to detect. Fig. 4 shows both accuracy and performance denoted as sensitivity and average processing time respectively on a vehicle dataset. In that case the accuracy of the detector was between 75% and 95% with input sizes between 352 and 544. Detecting large object will not be affected by the resizing of the image, thus the input of the network can be decreased to a certain point in order to increase the performance and maintain the accuracy. On the other hand, Fig. 5 shows the accuracy and the performance, using the same network, on a pedestrian dataset. The accuracy of the detector in that case, was between 25% and 75% which shows a reduction of 50%. Considering the resolution that cameras operate nowadays (e.g. 4K), this can lead to problems especially in cases where the resolution of the objects of interest becomes small due to the distance from the camera or for small objects. It is clear that the input size has a large impact on the accuracy for small objects and minimal impact on larger objects. This indicates that given the object size it might be efficient to reduce or decrease the input size in order to increase the performance or the accuracy of the detector accordingly. Embedded devices pose restrictions to the power consumption and memory footprint of machine learning applications. Therefore, there is a limit on how much the input size can be increased to improve the accuracy. Thus, there is a need of implementing efficient technique that keeps the input size small and at the same time maintains the high accuracy of the detector.

C. Selective Tile Processing

In order to make object detection more effective but at the same time maintain overall performance at similar levels a tiling strategy can be apply on high resolution images. This strategy separates the larger input image into smaller images with size equal to that of the CNN called *tiles* and selectively process only a subset of them using different selection criteria. At the same time, using a memory buffer, tracks the activity in other tiles in order to estimate the position of the object on the non-selected tiles. The assumption for using this simple approach is that the relative position of an object is not going to change significantly from the time that the same tile will be processed again due to the high performance of the detector. Tiling is an efficient way to increase the accuracy of the detector without increasing the memory requirements of the detector. The problem that arises with the tiling approach is when and how many times each tile must be selected for processing. This selection metric must be chosen by taking into consideration both the movement of the detected object and the camera. Moreover, for the selection of each tile one can choose different selection criteria based on his system requirements such as performance, accuracy, low-power consumption, reduction of memory. In the extreme case, it might be efficient to select one tile on each frame which will increase the performance and at the same time reduce the power consumption, due to the reduction of the computational requirements.

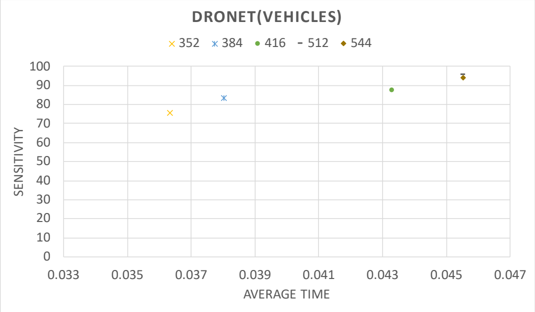

Fig. 4: DroNet Performance on vehicle dataset

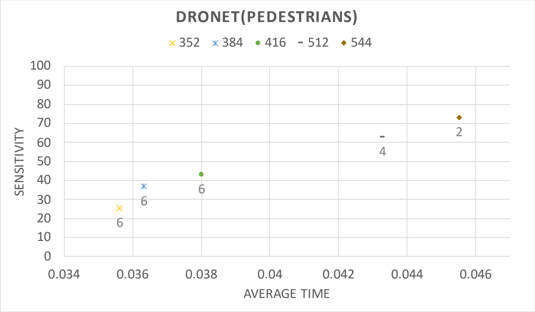

Fig. 5: DroNet Performance on pedestrian dataset

1) DroNet^{TA} - Processing of all Tiles: This approach process all the tiles before moving to the next frame. This will result in improving the accuracy but lead to a large performance degradation on the FPS of the detector.

2) DroNet^{T1} - Single tile processing: In the extreme case it might even be sufficient to process a single tile. Of course this method is agnostic to the activity in each tile while some image regions may need to processed more frequently than others.

3) DroNet^{TSM} - Process tiles based on selection criteria: A rather more solid approach is to use the detection information in order to steer the detector to select the top N tiles

Approach	APT (*sec*)	FPS	SEN (%)
$Tiny-YOLO$	1.64	0.61	63.51
$DroNet$	0.20	4.83	19.86
$DroNet^{\boldsymbol{TA}}$	0.96	1.03	92.37
$DroNet^{\boldsymbol{TI}}$	0.10	9.36	80.10
$DroNet^{\boldsymbol{TSM}}$	0.39	2.54	91.10

TABLE I:
Performance on Odroid XU4

for further processing instead of selecting all or one tile. A combined metric can be used to select one or more tiles for processing by selecting the most promising ones.

D. Odroid XU4

Odroid XU4 is an embedded device running Linux-based operating system with an octacore Samsung Exynos-5422 CPU at 2GHz with 2 GB of RAM. Table I shows the performance for all tiling approaches. It is clear that for the specific device $DroNet^{\boldsymbol{TI}}$ is by far better on FPS and accuracy for all the approaches. $DroNet^{\boldsymbol{TI}}$ achieved $16\times$ higher performance than the $Tiny-YoloV2$ and $2\times$ higher performance than the $DroNet$.

In our case, we carefully designed a structure that was efficient to detect both vehicles and pedestrians by keeping the number of filters relatively small (256), using only 1×1 and 3×3 size of filters and the stride between $1-2$, in order to increase the accuracy due to the reduction of the other values. Moreover, the tiling approach maintains the structure of the network, and at the same time enhances the network for high resolution images with minimal increase on the computational requirements and a significant increase on the accuracy. This shows that by using the aforementioned techniques and by carefully designing the structure of a network, it is feasible to push the intelligence to the edge.

V. Conclusions and Future Directions

Edge intelligence is recognized as the future for enabling artificial intelligence on embedded edge devices at mass. It has the potential to offer increasing security and reduced costs while maintaining the performance compared to processing on the cloud, and has the potential to enable new capabilities for industry and consumers alike. Thus, in the light of the above discussion, there is a significant need for improving the capabilities not only of the computing infrastructure and underlying processor architectures but also the efficiency of the algorithms used of machine learning.

In this paper we have discussed the challenges and the oportunities edge intelligence has to offer. In addition, we have presented a use-case that shows how the careful design of a CNN for object detection can lead to real-time performance on embedded edge devices. We anticipate that the co-optimization of algorithm and hardware architecture can provide the basis in order to build highly intelligent and resource efficient systems realizing the vision of edge intelligence.

Acknowledgment

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 739551 (KIOS CoE) and from the Government of the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

References

[1] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[4] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[5] L. Mearian, "Self-driving cars could create 1gb of data a second," *Computerworld*, vol. 23, 2013.

[6] L. Esterle, P. R. Lewis, R. McBride, and X. Yao, "The future of camera networks: Staying smart in a chaotic world," in *Proceedings of the 11th International Conference on Distributed Smart Cameras*, ser. ICDSC 2017. New York, NY, USA: ACM, 2017, pp. 163–168. [Online]. Available: http://doi.acm.org/10.1145/3131885.3131931

[7] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2831347.2831354

[8] Y. C. Hu *et al.*, "Edge Intelligence," IEC Market Strategy Board, Tech. Rep., 2017.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.

[10] S. Carlin and K. Curran, "Cloud computing security," in *Pervasive and Ubiquitous Technology Innovations for Ambient Intelligence Environments*. IGI Global, 2013, pp. 12–17.

[11] M. Margala and R. Lin, "Highly efficient digital cmos accelerator for image and graphics processing," in *ASIC/SOC Conference, 2002. 15th Annual IEEE International*. IEEE, 2002, pp. 127–132.

[12] D. A. Ferrucci, "Ibm's watson/deepqa," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ser. ISCA '11. New York, NY, USA: ACM, 2011, pp. –. [Online]. Available: http://dl.acm.org/citation.cfm?id=2000064.2019525

[13] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, pp. 1–12, Jun. 2017. [Online]. Available: http://doi.acm.org/10.1145/3140659.3080246

[14] "Movidius neural compute stick," Movidius, Tech. Rep., 01 2017.

[15] M. Shafique, T. Theocharides, C. S. Bouganis, M. A. Hanif, F. Khalid, R. Hafz, and S. Rehman, "An overview of next-generation architectures for machine learning: Roadmap, opportunities and challenges in the iot era," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 827–832.

[16] T. B. Preuer, G. Gambardella, N. Fraser, and M. Blott, "Inference of quantized neural networks on heterogeneous all-programmable devices," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 833–838.

[17] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," *CoRR*, vol. abs/1603.05279, 2016. [Online]. Available: http://arxiv.org/abs/1603.05279

[18] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," *CoRR*, vol. abs/1511.06530, 2015.

[19] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *CoRR*, vol. abs/1610.02357, 2016. [Online]. Available: http://arxiv.org/abs/1610.02357

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[21] L. Cavigelli, P. Degen, and L. Benini, “Cbinfer: Change-based inference for convolutional neural networks on video data,” in *Proceedings of the 11th International Conference on Distributed Smart Cameras*, ser. ICDSC 2017. New York, NY, USA: ACM, 2017, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/3131885.3131906

[22] G. E. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *CoRR*, vol. abs/1503.02531, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531

[23] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding,” *CoRR*, vol. abs/1510.00149, 2015. [Online]. Available: http://arxiv.org/abs/1510.00149

[24] C. Kyrkou, G. Plastiras, T. Theocharides, S. I. Venieris, and C. S. Bouganis, “Dronet: Efficient convolutional neural network detector for real-time uav applications,” in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 967–972.

[25] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, “Speed/accuracy trade-offs for modern convolutional object detectors,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 3296–3297.

[26] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6517–6525.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.