Smart Classrooms aided by Deep Neural Networks inference on Mobile Devices

Alberto Pacheco, Ever Flores, Raúl Sánchez
Intelligent Embedded Systems Laboratory
Instituto Tecnológico de Chihuahua, TecNM
Chihuahua, 31310, México
{ apacheco, eaflores, rgsanchez }@ itchihuahua.edu.mx

Salvador Almanza-García
IEEE Senior Member
Vector North America Inc.
Novi, Michigan, USA
salvador.almanza@vector.com

Abstract **-** Machine Learning over edge computing devices is levering up embedded and IoT intelligence and is expected to grow even more. Today, Machine Learning applications are mainly driven by Cloud Computing, but a recent trend towards on-device ML execution is taken over. We can expect that in coming years there will be smart homes and buildings populated by much more smart devices able to assist human activities more naturally. We present a work in progress, a mobile edge computing prototype built to explore and validate how ready are smartphones for on-device execution of deep neural networks (DNNs) using as a test bed a light control of a smart classroom via object recognition using three pre-trained non-optimized DNN models embedded in one mobile app. The prototype was successfully accomplished for recognition and control tasks using only smartphone's CPU/GPU processing units to run DNN models under 670ms. Finally, on-device DNN inference performance is discussed and some future work issues are presented.

Index Terms - Machine Learning; Deep Neural Networks; Mobile Edge Computing; Smart buildings.

I. Introduction

In recent years, deep learning (DL) has become a hot Computer Science and Engineering topic [1], offering an important big data analytic tool and methodology [2]. Cloud computing is the prevailing DL platform, supplying services through very large computing infrastructure, i.e. DL as a service (DLaaS). Edge computing is a more recent DL alternative. Constrained devices with limited processing, energy and memory resources offload computing tasks from the cloud. This paradigm potentially can: 1) reduce network traffic; 2) reduce cloud computing and storage resources; 3) reduce response time and network latency; 4) tolerate network intermittent service; 5) improve data safety and privacy [2, 3]. In the short term is expected that smart devices will play a prominent role in mobile, IoT, wearables, drones, VR, smart buildings and all kind of mobile and embedded devices [5]. Today mobile devices are equipped with multiple CPU/GPU/DSP cores and on-device sensors that may be useful for activity and context recognition [4], where machine learning algorithms may be applied for smart homes, smart buildings, as well as other interesting IoT application domains [7-9]. In summary, the extensive application of deep neural networks brings a new direction for the computer architecture research that is evolving rapidly [1], migrating from power-hungry cloud platforms (Cloud AI) towards new edge devices running DNNs (Mobile AI): mobiles, wearables, and IoT nodes within the milliwatt or microwatt power range [10].

This work is part of a larger smart classroom project, a simple scenario is presented in order to test on-device pre-trained DNNs for object recognition using a smartphone to control different classroom devices via an IoT micro-server. Several scenarios are considered, including voice recognition, object and handwritten instructions to activate diverse smart home devices. Also, a review of the state of the art DL on-device inference for constrained devices, and other related smart classroom works is presented. Finally, we provide some profile results from on-device non-optimized pre-trained DNNs, implemented on an iOS CoreML-based app capable of dynamically swap among different DNN models.

II. State of the Art Overview

On-device DNN inference (ODI) on constrained embedded devices is now a reality [5], but implies several technical challenges including limited memory, energy, and computational power. There are different alternatives to achieve an efficient and small DNN memory footprint suitable for mobile and embedded architectures (Table I). Those strategies rely on diverse methods and software frameworks to reduce the number of operations, algorithmic complexity or model size using pruning, compression algorithms, or parameter precision reduction (16-bits or less), among other methods. Some other solutions in Table I consider new DL optimized architectures for CPU, GPU, DSP, FPGA, ASIC DL co-processors or neural compute accelerators.

TABLE I
ODI mobile frameworks review (2015-2017)

Framework	Year	CPU	GPU	DSP	FPGA	Ref
DeepEar	2015			X		[11]
DeepX	2016	X	X	X		[12]
Caffeine	2016				X	[13]
DeepSense	2016		X			[14]
CNNdroid	2016		X			[15]
DeepMon	2017		X			[16]
MobiRNN	2017		X			[17]
RSTensorFlow	2017	X	X			[18]
SPARCNet	2017	X	X		X	[19]
fpgaConvNet	2017				X	[20]
DeepEye	2017	X	X			[21]

978-1-5386-5398-2/18/$31.00 ©2018 IEEE

A. *On-device DNN Inference (ODI) Frameworks*

Table I lists some ODI frameworks. DeepEar [11] was the first ODI for audio sensing and activity recognition tasks by reducing algorithmic complexity to deliver a smaller memory footprint model (8MB, 2.3 million of parameters) running on a low-power embedded Hexagon DSP of the Snapdragon 800. DeepEar could run continuously up to 24 hours, achieving 30% higher accuracy and exceeding the performance of other mobile audio classifiers [11]. DeepX improved the performance of DeepEar [5] by compressing a DNN model using singular value decomposition (SVD) running over the Snapdragon 800 by splitting computations across multiple co-processors (CPU, GPU, DSP) achieving up to 7 times more energy savings [12]. Caffeine [13] is an FPGA CNN accelerator that achieved 7x and 43x performance and energy gains over a double 6-core Intel Xeon CPU and a 1.5x better energy-efficiency over Nvidia K40 GPU on a medium-sized KU060 FPGA. DeepSense [14] is an OpenCL-based framework to run ODI on mobile GPUs that reduces inference time up to 74 times comparing to conventional mobile CPU implementations. Similarly, CNNdroid [15] and DeepMon [16] are open source ODI GPU-based accelerators for Android-based mobile devices: CNNdroid runs up to 60x faster on current mobile devices, and DeepMon runs almost twice faster than DeepX. MobiRNN [17] is a RenderScript-based ODI optimizer framework to run recurrent neural networks on mobile GPUs up to 4x faster. RSTensorFlow [18] extends the open source framework TensorFlow with the heterogeneous computing framework RenderScript to accelerate ODI execution on Android devices. The SPARCNet heterogeneous architecture accelerator [19] runs 11x faster and with 7x less energy on FPGAs than the equivalent Tegra TX1 CPU+GPU kit. fpgaConvNet [20] is a framework that automates the optimized mapping of CNNs on FPGAs achieving 5x speed out over Tegra X1. DeepEye improved over DeepX [21] providing an ODI wearable camera with a more advanced *layer-centric* execution framework able to execute multiple CNN models on a quad-core Snapdragon 410.

Other time/energy-efficient alternatives are the DNN co-processors, such as the Intel Movidius Neural Compute Stick (NCS) [20], a low-cost USB-drive-sized inference deep learning coprocessor. The NCS is a Myriad2 always-on vision processor with a high-performance Vision Processing Unit (VPU), which consumes only 1W of power. According to [22] a 545% speedup on GoogLeNet and a 396% speedup on SqueezeNet over Raspberry Pi 3 (RPi).

The new Apple iPhone 8/X multi-core A11 Bionic CPU has a *neural engine* coprocessor that can perform up to 600 GOPs in a more energy-efficient manner than using either CPU or GPU cores. Main applications are Face ID, Animoji, augmented reality object-detection and other ML system-level tasks. The *CoreML framework* adds a new iOS layer (Fig. 1) for deploying natural language processing, image analysis, computer vision techniques, and augmented reality experiences. To achieve optimized ODI performance, iOS provides the *Accelerate framework* basic neural network subroutines (BNNS) to optimize multicore CPU performance. Also below the *CoreML* layer, the *Metal Performance Shaders framework* adds low-level, high-performance data-parallel primitives, fine-tuned for optimal GPU performance (Fig. 1).

Unlike the proliferation of ODI smartphone-based with multicore architecture, powerful GPU units and Gigabytes of RAM, few examples of ML 16/32-bit ARM-based embedded systems exist [5]. Fortunately, there are promising results of deep binary architectures using 1-bit embedded binarized neural networks (eBNNs) and two-bit networks (TBNs) that achieve a 32x size reduction, making them suitable for deployment on IoT and embedded devices [23].

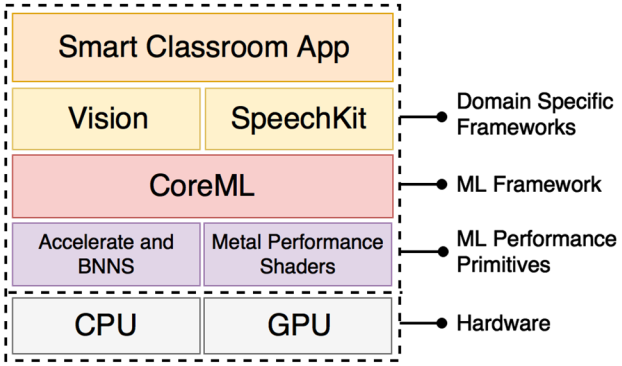

Fig. 1. iOS CoreML-based Machine Learning architecture.

B. *Smart Classrooms*

On this subsection, we present some smart classroom related works. The concept of smart classroom is quite broad and has evolved over time. Some proposals used interactive smart boards or amazing multimedia devices [24-27]. Others could control multimedia equipment depending on the activity being carried out by the instructor using voice commands or gestures [28, 29]. Some other group of works proposed to use ubiquitous and mobile computing, sensor networks, robotics, multimedia computing, middleware, agent-based software and AI improving the learning experience of students, lectures, and teaching [30-33]. Only a few of them applied deep learning algorithms to assist or improve the learning experience within the classroom as to recognize hand movements and gestures to draw on canvas [34] or made suggestions to an in-class presenter to adjust their non-verbal behavior based on emotion recognition [35].

III. Development and Validation Stages

In contrast to previous works, this prototype implements an on-device deep learning inference to assist the activation of diverse classroom devices (lights, door lock and multimedia projector) using the smartphone video camera to provide a simpler natural user interface, to easy device remote control and at the same time, to optimize energy consumption. This first prototype integrates the work progress of two graduate students working on their master's thesis. This initial stage implements an initial non-smart manual control which will help to optimize DNNs in the future; the user interacts and de/activates directly different classroom devices.

For this prototype, different DNNs were used for object, text and voice recognition using pictures directly of in-classroom objects, or instructor's written or spoken words (on this article only object recognition results are presented). For object recognition, different pre-trained non-optimized DNNs were used and tested. A mobile iOS app was implemented for hot-swapping [37] and run different DNNs for each object recognition task. The mobile app prototype performs ODI using the iOS CoreML framework on an iPhone 6. All input and processing tasks (taken picture, DNN inference, and control decision) were processed on this edge computing smartphone. However, each in-class device was activated or deactivated via wireless HTTP requests directed to a Raspberry Pi 3 micro-server (Fig. 2), which takes direct control of any classroom device (lighting and door lock).

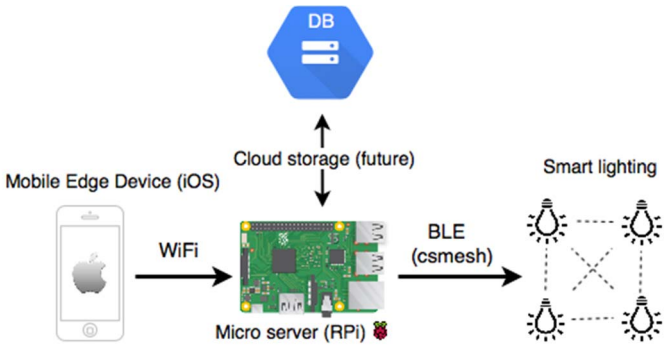

Fig. 2. Smart classroom micro-server and mobile app communication.

The prototype development was subdivided in: a) the iOS app with UI components, image acquisition, DNN inference, HTTP M2M protocol; b) micro-server implementation: Raspbian Flask-based server configuration, HTTP M2M protocol, Bluetooth *csrmesh* protocol [36]; c) validation phase: object recognition, wireless control tasks, on-device DNN memory and CPU/GPU profiling.

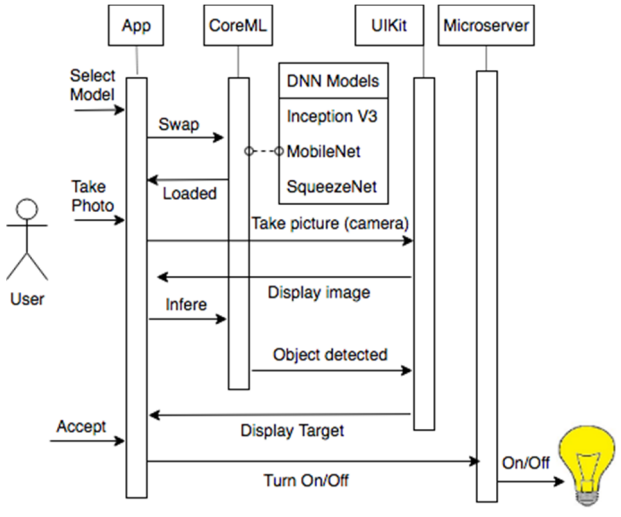

Fig. 3. Smart classroom UML sequence diagram.

The prototype was implemented and tested with three different object recognition DNN classification models: Inception v3, MobileNet, and SqueezeNet. All of them are ImageNet pre-trained models based on convolutional neural networks that can recognize up to 1,000 categories. Once a picture is taken, and depending the object being recognized by the selected DNN model, the app sends HTTP requests to the micro-server (RPi) to activate the corresponding in-classroom wireless device: led lights, door lock or multimedia projector (Fig. 2). The micro-server stores transactions and status on an external database in the cloud.

Fig. 3 represents the UML sequence diagram of the operational model prototype currently implemented. First, the app's user selects one of the DNN models, swapping the CoreML model to Inception, MobileNet or SqueezeNet [37]. When the user takes a picture of a target object, the app runs the neural network model, which returns the classification category label with the highest percentage. The mobile app decides then what is the corresponding device action for the target device and sends a micro-server HTTP request to activate or deactivate the corresponding in-class device.

TABLE II
DNN MODEL CHARACTERIZATION

Model	# layers	Size (MB)	Top 1 ImageNet accuracy
InceptionV3	48	94.7	77 %
MobileNet	28	17.1	70.6 %
SqueezeNet	18	4.8	57.5 %

IV. Profiling Results

All ODI DNN models were tested and profiled on an iPhone 6 with iOS 11 using XCode 9.3, obtaining memory and timing profiling reports. During the app startup, DNN models are initialized; this process consumes some time and memory resources (it seems that CoreML initialize models ordered from heaviest to lightest model). Also, the first DNN inference takes the longest latency and memory consumption, regardless of the model used [38].

TABLE III
MEMORY PROFILE

Model	Persistent heap allocated memory		Dynamic and destroyed heap allocated memory	
	Init	*Prediction*	*Init*	*Prediction*
InceptionV3	1.5 MB	1.5 MB	133 MB	3.5 MB
MobileNet	212 KB	706 KB	25 MB	1 MB
SqueezeNet	170 KB	688 KB	8.5 MB	914 KB

Table II describes each DNN: number of layers, memory size, and standardized accuracy tests results. Table III shows average memory resources allocated while DNN inferences are performed (static and dynamic memory allocations). The deepest DNN architecture consumes more memory resources.

For example, for the 95 MB Inception V3 model, initialization and loading consumes less than 2MB, and it requires other 5MB for running this model (DNN inference). It appears that the iOS paging mechanism achieved some runtime optimization, consuming an average of only 7% of the DNN model size when DNN inferences were performed.

DNNs initialization and inference time profiles are shown in Table IV. Those times roughly correspond to their model size and complexity (number of layers); so again, the biggest InceptionV3 model takes the longest time to execute.

TABLE IV
TIMIMNG PROFILE

Model	Time	
	Init	***Prediction***
InceptionV3	800-1200 ms	610-670 ms
MobileNet	62-190 ms	170- 175 ms
SqueezeNet	66-87 ms	80 - 90 ms

Table V reports processing capacity in frames per second using the Vision Framework, where each frame corresponds to an image processed, i.e. DNN inference. Inception V3 had the highest latency (1.5 fps). The fastest model was SqueezeNet (11x faster than Inception V3), but had the lowest accurancy (57%). Between both extremes, MobileNet (5x lighter than Inception V3) had a medium latency and accuracy (70%).

TABLE V
PERFORMANCE PROFILE (VISION FRAMEWORK)

Model	***FPS***	***CPU usage (%)***	***CPU***	***GPU***
InceptionV3	1.5	15-18	680 ms	658 ms
MobileNet	9.8-10	41-51	102 ms	87 ms
SqueezeNet	15-16	60-71	70 ms	58 ms

V. CONCLUSIONS AND FUTURE WORK

This article provides a smartphone profiling study for on-device deep learning inference applied for a smart classroom prototype. A use case for object recognition aimed to control different classroom wireless devices through a mobile edge device (smartphone) controlled by an IoT microserver is described. The literature review performed found few smart classroom use cases that applied deep learning algorithms to assist or improve the learning experience inside a classroom. The prototype demonstrates it is feasible to dynamically swap and perform on-device DNN inferences of pre-trained non-optimized DNN models using iOS CoreML. It was found that most complex and heavy models performed slower, consumed more memory, but also had better accuracy (Inception V3). The next step will be to optimize CNN models, perform a more detailed profiling test including battery consumption, and extends the mobile app to support an educational activity called “Programming my Smart Home” to promote STEM and computational thinking for kids.

ACKNOWLEDGMENT

This work is supported by the TecNM via "Aula Inteligente con interfaz basada en realidad aumentada y aprendizaje automático" project (6417.18-P). We also thank Salvador Almanza from Vector North America Inc., and IEEE Senior Member from the Computer Chapter of the IEEE Southeastern Michigan Section for his advice and contributions to this project.

REFERENCES

[1] Q. Li, Q. Xiao, and Y. Liang, “Enabling high performance deep learning networks on embedded systems,” on IECON 2017 - 43rd Annual Conf. IEEE Ind. Electronics Soc., 2017, pp. 8405–8410.

[2] H. Li, K. Ota, and M. Dong, “Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing”, IEEE Network, vol. 32, no. 1, pp. 96–101, Jan. 2018.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge Computing: Vision and Challenges”, IEEE Internet of Things Journal, vol. 3, núm. 5, pp. 637–646, oct. 2016.

[4] P. Georgiev, S. Bhattacharya, N. D. Lane, and C. Mascolo, “Low-resource Multi-task Audio Sensing for Mobile and Embedded Devices via Shared Deep Neural Network Representations,” Proc. ACM Interact. Mob. Wearable Ubiquitous Technol., vol. 1, no. 3, pp. 1–19, Sep. 2017.

[5] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar, “Squeezing Deep Learning into Mobile and Embedded Devices,” IEEE Pervasive Computing, vol. 16, no. 3, pp. 82–88, 2017.

[6] V. Radu et al., “Multimodal Deep Learning for Activity and Context Recognition,” Proc. ACM Interact. Mob. Wearable Ubiquitous Technol., vol. 1, no. 4, pp. 157:1–157:27, Jan. 2018.

[7] M. Manic, K. Amarasinghe, J. J. Rodriguez-Andina, and C. Rieger, “Intelligent Buildings of the Future: Cyberaware, Deep Learning Powered, and Human Interacting,” IEEE Industrial Electronics Magazine, vol. 10, no. 4, pp. 32–49, 2016.

[8] Y. He, F. R. Yu, N. Zhao, VC Leung, and H. Yin, “Software- defined Networks with Mobile Edge Computing and Caching for Smart Cities: A Big Data Deep Reinforcement Learning Approach,” *IEEE Comm. Magazine*, vol. 55, no. 12, pp. 31–37, Dec. 2017.

[9] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and JS Oh, “Semi-supervised Deep Reinforcement Learning in Support of IoT and Smart City Services,” *IEEE IoT Journal*, no. 99, p. 1, 2017.

[10] M. Verhelst and B. Moons, “Embedded Deep Neural Network Processing: Algorithmic and Processor Techniques Bring Deep Learning to IoT and Edge Devices,” IEEE Solid-State Circuits Magazine, vol. 9, no. 4, pp. 55–65, Fall 2017.

[11] N. Lane, et al., “Deepear: Robust smartphone audio sensing in unconstrained acoustic environments using deep learning,” on UbiComp ’15.

[12] N. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar, “Accelerating Embedded Deep Learning Using DeepX: Demonstration Abstract,” on Proc. 15th Int. Conf. Information Processing in Sensor Networks, NJ, USA, 2016, p. 61:1–61:2.

[13] C. Zhang, Z. Fang, P. Zhou, P. Pan, and J. Cong, “Caffeine: towards uniformed representation and acceleration for deep convolutional neural networks,” on ICCAD ’16, Austin, TX, 2016, pp. 1–8.

[14] L. N. Huynh, R. K. Balan, and Y. Lee, “DeepSense: A GPU-based Deep Convolutional Neural Network Framework on Commodity Mobile Devices”, on Proc. 2016 Workshop on Wearable Systems and Applications, New York, NY, USA, 2016, pp. 25–30.

[15] S. S. Latifi Oskouei, H. Golestani, M. Hashemi, and S. Ghiasi, “Ddroid: GPU-Accelerated Execution of Trained Deep Convolutional Neural Networks on Android”, on Proc. 2016 ACM Multimedia Conf., New York, NY, USA, 2016, pp. 1201–1205.

[16] L. Huynh, Y. Lee, and R. Balan, “DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications”, Proc. 15th Annual Int. Conf. on Mobile Systems, Applications, and Services, 2017, pp. 82–95.

[17]Q. Cao, N. Balasubramanian, and A. Balasubramanian, “MobiRNN: Efficient Recurrent Neural Network Execution on Mobile GPU”, on EMDL’17, NY, USA, 2017, pp. 1–6.
[18]M. Alzantot, Y. Wang, Z. Ren, and M. B. Srivastava, “RSTensorFlow: GPU Enabled TensorFlow for Deep Learning on Commodity Android Devices”, on Proc. of the 1st Int. Workshop on Deep Learning for Mobile Systems and Applications, New York, NY, USA, 2017, pp. 7–12.
[19]A. Page, A. Jafari, C. Shea, and T. Mohsenin, “SPARCNet: A hardware accelerator for efficient deployment of sparse Convolutional Networks,” J. Emerg. Tech. Comp. Sys., vol. 13, núm. 3, p. 31:1–32, may 2017.
[20]S. I. Venieris and C.-S. Bouganis, “fpgaConvNet: A Toolflow for Mapping Diverse Convolutional Neural Networks on Embedded FPGAs,” on NIPS 2017, Long Beach, CA, 2017.
[21]A. Mathur, N. D. Lane, S. Bhattacharya, A. Boran, C. Forlivesi, and F. Kawsar, “DeepEye: Resource Efficient Local Execution of Multiple Deep Vision Models Using Wearable Commodity Hardware”, on Proc. 15th Annual International Conf. on Mobile Systems, Applications, and Services, NY, USA, 2017, pp. 68–81.
[22]A. Rosebrock, “Getting started with the Intel Movidius Neural Compute Stick,” PyImageSearch blog, 12-Feb-2018.
[23]W. Meng, Z. Gu, M. Zhang, and Z. Wu, “Two-Bit Networks for Deep Learning on Resource-Constrained Embedded Devices”, arXiv:1701.00485 preprint, Jan. 2017.
[24]SmartBoards, “Smart Boards | Are you sure which smart board is right for you?,” 2018. Available: goo.gl/9AmrwF [Accessed: 30-Jan-2018].
[25]SmartTech, “SMART Classroom Suite,” *SMART Technologies*, 2011. Available:https://goo.gl/66WFVH [Accessed: 30-Jan-2018].
[26]SmartWay, “Smart Classroom,” *SmartWay*, 18-Feb-2015. Available:https://goo.gl/tUYSkm [Accessed: 30-Jan-2018].
[27]M. Alderton, “Smart classrooms give tech boost to learning,” USA Today, 2016. Available: https://goo.gl/6RK8nB [Accessed: 30-Jan-2018].
[28]L. R. Winer and J. Cooperstock, “The ‘intelligent classroom’: Changing teaching and learning with an evolving technological environment,” *Computers & Education*, vol. 38, no. 1–3, pp. 253–266, 2002.
[29]D. Franklin, J. Flachsbart, and K. Hammond, “The intelligent classroom,” *IEEE Intelligent Sys. and their Applications*, vol. 14, no. 5, pp. 2–5, 1999.
[30]M. Antona, A. Leonidis, G. Margetis, M. Korozi, S. Ntoa, and C. Stephanidis, “A Student-Centric Intelligent Classroom,” in *Ambient Intelligence*, 2011, pp. 248–252.
[31]T. V. Ark, “8 Ways Machine Learning Will Improve Education,” *Getting Smart*, 26-Nov-2015.
[32]B. Dickson, “How Artificial Intelligence enhances education,” *The Next Web*, 2017. Available: https://goo.gl/YreYQZ [Accessed: 30-Jan-2018].
[33]V.K. Maheshwari, “THE CONCEPT OF SMART CLASSROOM,” 2016.
[34]A. Dash *et al.*, “AirScript - Creating Documents in Air,” in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017, vol. 01, pp. 908–913.
[35]Y. Kim, T. Soyata, and R. F. Behnagh, “Towards Emotionally Aware AI Smart Classroom: Current Issues and Directions for Engineering and Education,” *IEEE Access*, vol. 6, pp. 5308–5331, 2018.
[36]Qualcomm, “CSRmesh technology for Bluetooth-based networks”. Available:https://goo.gl/sKJLoU [Accessed: Feb. 25, 2018].
[37]C. Dahlstrom, “Hot swapping Core ML models on the iPhone,” *Zedge*, 2017. Available:https://goo.gl/ziX77j [Accessed: 18-Feb-2018].
[38]M. Hollemans, “A peek inside Core ML”, August 2017. Available:https://goo.gl/8MMWMj [Accessed: Jan. 13, 2018].