Device Non-Ideality Effects and Architecture-aware Training in RRAM In-Memory Computing Modules

Qiwen Wang, Yongmo Park, and Wei D. Lu*
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, Michigan 48109, USA
*Email: wluee@umich.edu

2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401307

***Abstract*—We studied factors that could degrade model performance in analog RRAM in-memory-computing (IMC) systems, including limited array size, ADC resolution, on/off ratio, and device conductance variations. Different levels of architecture-aware training methods were developed to mitigate these factors and allow the system to achieve accuracy comparable to floating-point baseline with realistic device parameters.**

I. Introduction

Machine learning algorithms represented by deep neural networks (DNN) have been employed for a wide range of applications. At the same time, traditional computing architectures are ill-suited for the high-memory-access and highly parallelized nature of DNN workloads. Thus, accelerators will be important for the further adoption of DNN networks, and RRAM-crossbar-array-based in-memory computing (IMC) architectures have gained popularity. RRAM arrays can perform vector-matrix multiplication in analog domain efficiently by accumulating the total current or charge at each column, while its high density, CMOS compatibility, and non-volatile nature make it possible to store entire networks with millions of weights on-chip, thus eliminating the memory bottleneck. However, potential issues when implementing large-scale models in practical RRAM arrays still need to be carefully addressed.

The possibility of mapping state-of-art networks on realistically sized RRAM arrays has been demonstrated using a tiled architecture [1][2][3]. However, accuracy loss still existed, and the RRAM device properties and ADC requirements are challenging to achieve. In this work, we studied the effects of training by incorporating different levels of architecture details. We found that through careful architecture-aware training, IMC systems can achieve accuracies similar to floating-point baseline with further relaxed requirements for both RRAM devices and ADC precision, suggesting RRAM-based IMCs can be applicable in the near term. Furthermore, device variations can have a significant impact on inference performance, particularly for more complex models, and require innovative solutions to address.

II. Analog Crossbar Array Implementation

A. Mapping DNN Model to RRAM Arrays

In this work, we focus on performance during inference operations, which is the primary workload in edge computing use cases. Pre-trained DNN models are programmed on to the RRAM arrays offline, and during inference, the RRAM cells are not modified. Even a single layer in today's DNN models is too

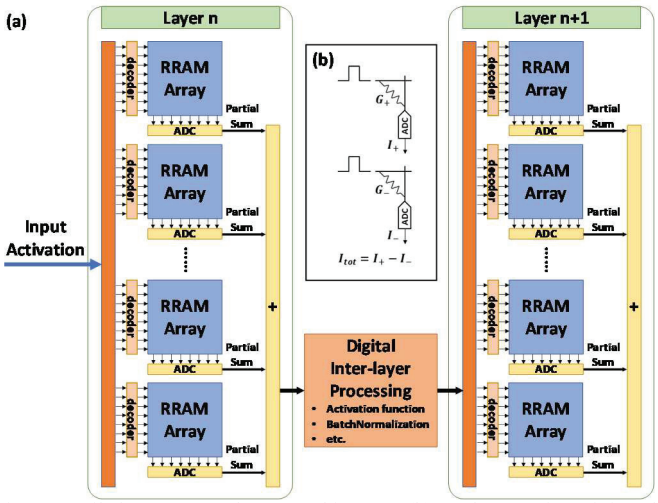

Fig. 1. (a)Tiled RRAM accelerator architecture. (b) Implementation of signed weights.

	Filter Shape	Input Size	Row Vector Length	Number of Arrays
CNN 1	3 x 3 x 3 x 32	32 x 32 x 3	27	1
CNN 2	3 x 3 x 32 x 32	32 x 32 x 32	288	2
MaxPooling	Pool 2 x 2	32 x 32 x 32		
CNN 3	3 x 3 x 32 x 64	16 x 16 x 32	288	2
CNN 4	3 x 3 x 64 x 64	16 x 16 x 64	576	3
MaxPooling	Pool 2 x 2	16 x 16 x 64		
CNN 5	3 x 3 x 64 x 128	8 x 8 x 64	576	6
CNN 6	3 x 3 x 128 x 128	8 x 8 x 128	1152	10
MaxPooling	Pool 2 x 2	8 x 8 x 128		
Flatten	Flatten	4 x 4 x 128		
FC 1	2048 x 128	2048	2048	16
FC 2	128 x 10	128	128	1
Total				41 Arrays

Fig. 2. The CIFAR-10 VGG Block Model as an example, and the number of arrays needed for each layer. Array size is 264x64.

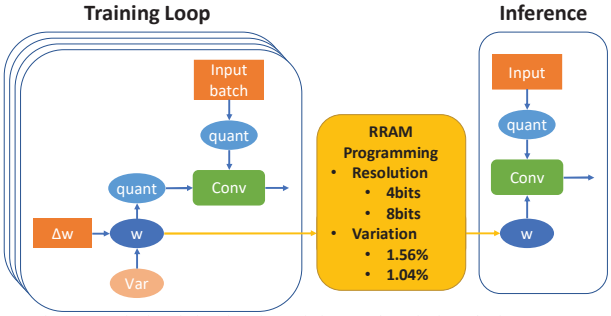

Fig. 3. Variation injection in training and variation during RRAM programming

978-1-7281-9201-7/21/$31.00 ©2021 IEEE

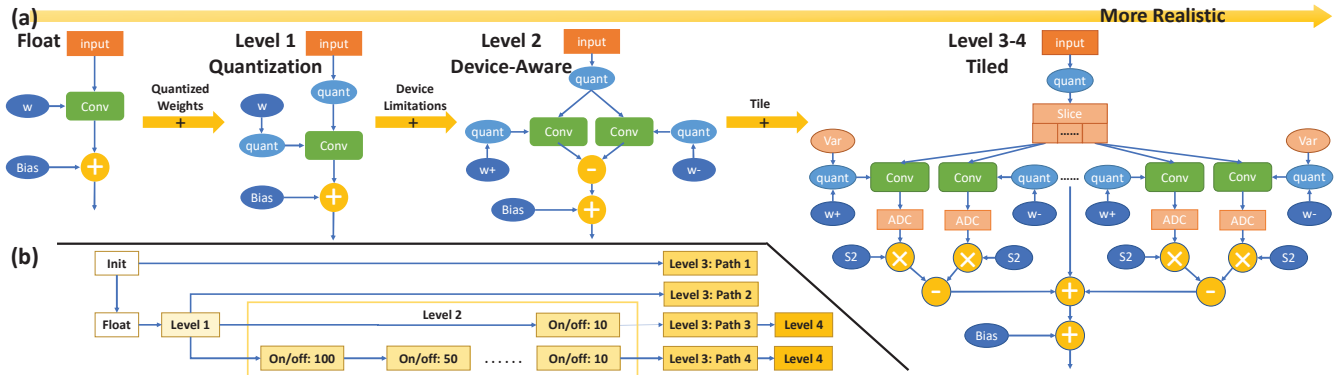

Fig. 4. (a) Training topologies corresponding to different levels of architecture-aware training. (b) Different training approaches.

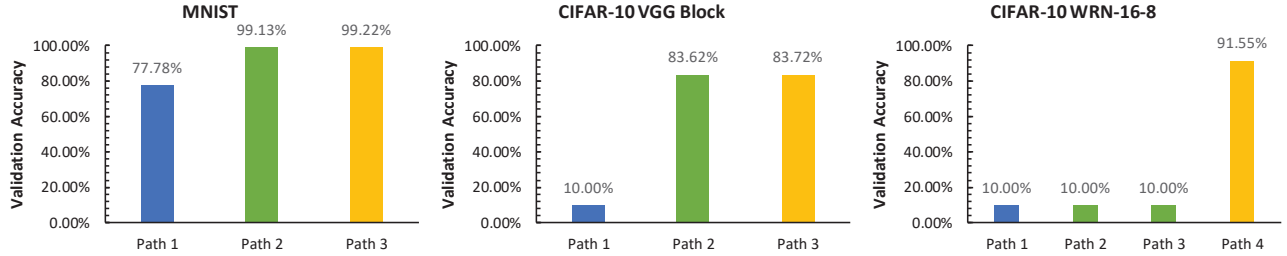

Fig. 5. Test accuracy of model trained in Level 3 topology, using different training paths described in Fig. 4b.

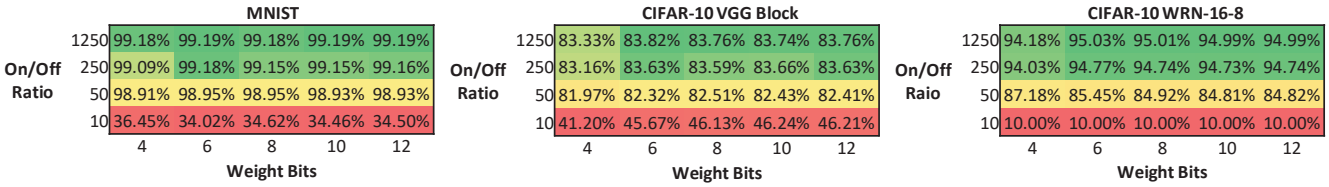

Fig. 6. IMC inference results by post-quantizing floating-point models, for different weight precision and On/Off. Effects of tiles and mapping variations are neglected.

large to fit in a single practical RRAM array and must be mapped onto multiple arrays. ADCs are used to readout currents in each array to produce partial sums (Psum) that are then added together in digital domain to produce the final layer outputs (Fig. 1a). This process was discussed in detail in [1]. Additionally, since RRAM cells can only represent positive conductance values, we store signed weight in two cells (Fig 1b). An example of a DNN model used in this study and mapping to the tiled architecture is shown in Fig. 2.

B. Realistic RRAM and Circuit Characteristics

Several RRAM cell non-ideality factors have been considered, including conductance resolution down to 4bits, on/off ratio down to 10, and device conductance variation up to 1.56% of the dynamic range, corresponding to only 2σ separation between conductance levels at 4bits. The cell max read current was assumed to be 3μA during inference. Realistic circuit parameters were also used. We considered an 8bit ADC, which is a reasonable compromise between precision, power, and area. Effects of bit-serial operations where each bit needs to be quantized in the ADC were also considered during our analysis.

In RRAM analog computing, the weight is represented by the analog conductance level thus the programmed weights inevitably deviate from the targets (Fig. 3). The deviation leads to a mapped model that is slightly different from the trained model, and this deviation does not change during the inference process. This fixed deviation is different from noise sources that vary between operations. It can have a large effect on IMC inference accuracy but is uncommon in digital inference systems, and standard training processes may not be effective in mitigating it since its effects occur during weight transfer (instead of weight updates during training) and are not well captured during training processes.

III. Architecture-Aware Training

A. Training Topologies

We propose that the architecture-aware training approach can be generally considered in 4 levels (Fig. 4a). Quantization-aware training (Level 1) has already been used by many to deploy DNN models with integer operations [4]. In quantized inference, weights are stored in low precision (e.g. 4-8bits), but multiplication and accumulation are carried out in high precision (e.g. 32bits) then quantized to low precision activation afterward (e.g. 8bits). This is similar to computation in analog crossbar arrays where weights represented by RRAM devices are low precision, but multiplication and accumulation are completed in analog domain without quantization loss to preserve higher precision. This is the first level we implement to train the IMC system. Built on top of this, we propose three more levels that incrementally add more hardware architecture details to improve inference accuracy in analog crossbar arrays.

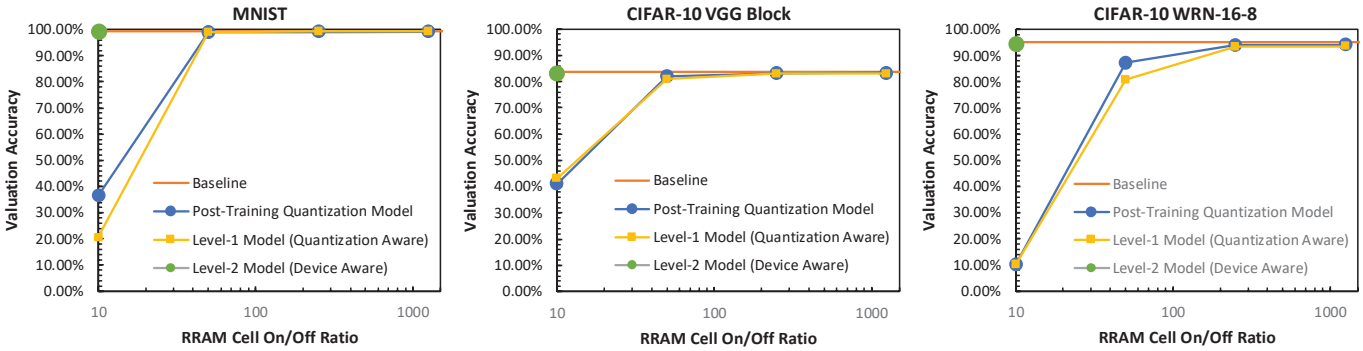

Fig. 7. Effects of device-aware training, for different On/Off ratios and weight precision of 4bit during inference. Effects of tiles and mapping variations are neglected.

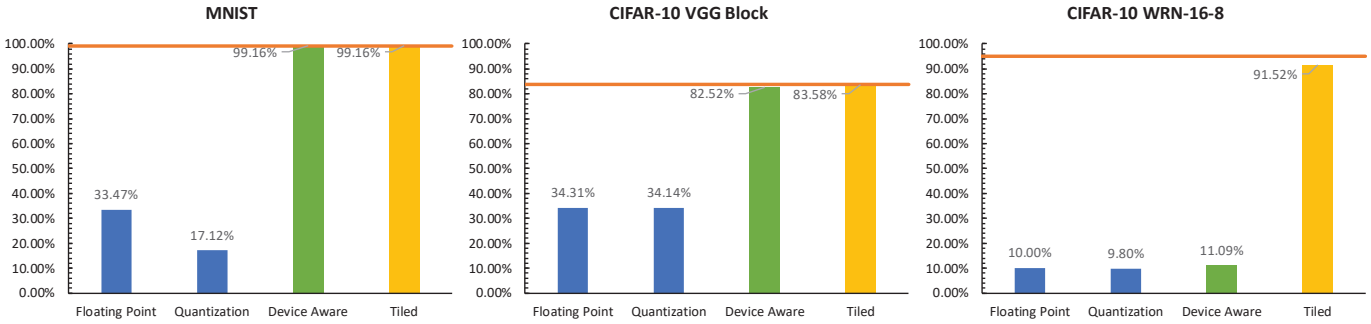

Fig. 8. IMC inference performance when realistic tile sizes are considered. Orange line: floating point baseline. Mapping variations are neglected.

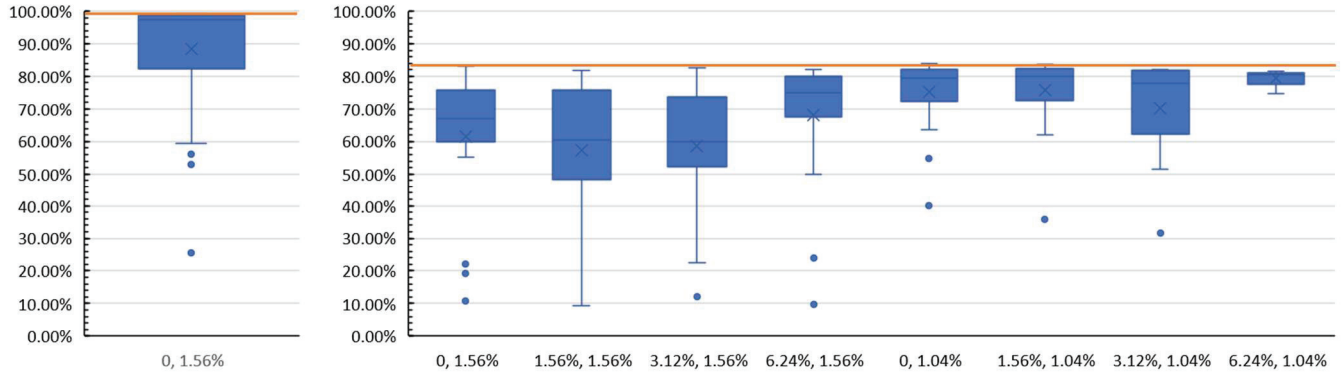

Fig. 9. Effects of different device variations for (a) MNIST and (b) CIFAR-10 VGG Block models, trained using Level 4, approach A. The first number represents noise injected during training and the 2nd number represents actual device variation during weight mapping. Orange line: floating point baseline

First, we consider realistic device properties such as the limited device on/off ratio and the effect of implementing signed weights in two cells . At this "device-aware" level (Level 2) we neglect the finite array size effects and note the limited weight precision is already addressed in Level 1 during quantization aware training.

Building on the device-aware topology, the tiled topology is introduced at Level 3 (tile-aware). During training, a single layer is sliced into multiple tiles corresponding to the size of RRAM arrays used, then quantization functions are applied after each tile to reflect the limited resolution and dynamic range of the ADCs. Different ADC implementations were also analyzed at this level.

Finally, we studied two approaches to address RRAM cell conductance variation during weight mapping (Level 4). In Approach A, during training, weights are quantized to the same number of bits the RRAM cell can represent (e.g. 4bits), then variation is injected. This is the common approach. In Approach B, we allow higher target resolution than the cells can achieve. During training and mapping, weights are quantized to a higher number of bits than the cells can represent (e.g. 8bits), with the same injected variations as Approach A. This is somewhat counterintuitive since the higher precision model cannot be reliably programmed due to the large variation (larger than the level spacing between the levels), but this approach was found to lead to better results.

IV. Results

A. *Inference Evaluation Setup*

We consider an RRAM-based IMC system with array size of 256x64 with properties described in Section 2b. We used three networks as examples. The first is a simple MNIST toy model: CNN1 3x3x1x22, MaxPool 2x2, CNN2 3x3x22x27, MaxPool 2x2, CNN3 3x3x27x64 Stride 2x2, MaxPool 4x4,

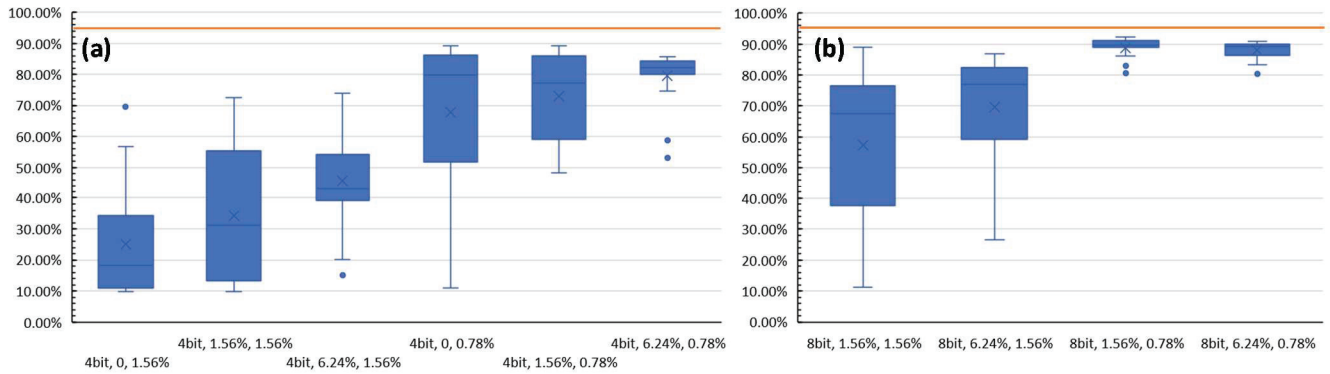

Fig. 10. WRN-16-8 performance for (a) Training and inference with 4bit target at different variations (Approach A), and (b) Training and inference with 8bit target at the same level of variations (Approach B). Orange line: floating point baseline

Dense 64x10. In this model, all layers are small enough to fit onto a single array. The 2nd is a VGG-block-based [5] network trained on the CIFAR-10 dataset (shown in Fig. 2). The 3rd model is Wide ResNet WRN-16-8 [6], which is a more complex and well-established benchmark network. Both simple 8bit ADC and bit-serial ADC were examined. All results shown are based on simple 8bit ADC, although we found the added per-bit quantization in bit-serial operation does not have significant effects.

B. *Training Process*

We first train each of the models in floating point. For MNIST and the VGG block model, we use SGD optimizer with learning rate of 0.0001, momentum of 0.9. For the WRN model, we follow parameters described in [6]. The Level 1 model is fine-tuned from the floating-point model, and each new level is fine-tuned from the model in the previous level. We found this approach leads to better results while training Level 3 or 4 directly from random initialization fails to converge in many cases or leads to degraded accuracy (Fig. 5).

C. *Device-Aware Training Results (Level 1-2)*

Effects of device properties such as the device conductance precision, on/off ratio, and signed weight representation are first examined, neglecting tile effects and mapping variations. As shown in Figs. 6,7, at high on/off ratios, floating-point models and Level 1 models can achieve good accuracy even at 4bit weights. This is consistent with earlier results, where many models can retain accuracy with simple post-training quantization. However, when the device on/off ratio decreases even moderately, the model accuracies drop to unacceptable levels even for the MNIST toy model. After taking into account the device factors in the training topology (Level 2), inference accuracy of all three models recovered to levels similar to that of the floating-point baselines, even at on/off ratio of 10 and weight precision of 4bits (Fig. 7).

D. *Tiled Architecture Results (Level 3)*

Effects of the tiled architecture with realistic RRAM array size and ADC characteristics were then analyzed. As shown in Fig. 8, both the floating-point model and the Level 1 quantization-aware trained models produce very poor accuracy for all 3 models during inference when implemented in the tiled architecture. For the MNIST model, because every layer can fit in a single array, the device-aware topology (Level 2) is sufficient to recover the baseline accuracy (Fig. 8). For the relatively simple CIFAR-10 VGG block model, the device-aware topology trained model also produces reasonable accuracy with a 1.21% drop. This drop is reduced to only 0.15% with the tiled topology trained model (Level 3). For the WRN model, the device-aware trained models produced accuracy barely better than chance, highlighting the effects of ADC limitation on Psums in the tiled implementation. By considering the Psum quantization effect (Level 3), accuracies of all modes can recover to the baseline during inference.

E. *Effect of RRAM Programming Variations (Level 4)*

Effects of RRAM cell programming variations are then added. Due to the stochastic nature of conductance variations, the mapping was run 20 times to characterize the distribution of the inference performance. For the MNIST model, even with a large standard deviation of 1.56% (2σ separation between 4bit levels), most of the runs are within 1% of the baseline (Fig 9a). However, the two more complex models exhibited a large spread in accuracy. By using Approach A described in Section 3b, with 1.56% RRAM variation the CIFAR-10 VGG Block model accuracy is improved to acceptable levels, and larger noise injected during training helps to reduce the spread of the model performance (Fig. 9b). However, injecting higher levels of noise during training does not improve the WRN model performance and often leads to larger spread during inference instead (Fig. 10a). Better results can be obtained with tighter control of precision to 0.78%, but overall accuracy is still significantly lower than the baseline.

Since programming variations lead to weights slightly different from the target, inference on the RRAM arrays is thus performed on essentially slightly different models than the trained one, so injecting noise during training may not fully account for this effect. Device variation during weight mapping can thus pose a serious challenge for inference applications, particularly for more complex models.

By using Approach B, improved performance can be obtained at the same training and inference variations. Tighter distributions are observed, and accuracy can approach the baseline at 0.78% variation (Fig. 10b). We hypothesize that the higher target RRAM resolution during training produces more stable models with a wider local minimum, which can then better tolerate mapping variations.

ACKNOWLEDGMENT

This work was supported in part by SRC and DARPA through the Applications Driving Architectures (ADA) Research Center, and by Applied Materials.

REFERENCES

[1] Q. Wang, X. Wang, S. H. Lee, F.-H. Meng, and W. D. Lu, "A Deep Neural Network Accelerator Based on Tiled RRAM Architecture," *IEEE International Electron Devices Meeting (IEDM)*, pp. 14.4.1-14.4.4, 2019.

[2] V. Joshi et al., "Accurate deep neural network inference using computational phase-change memory," *Nature Communications*, vol. 11, no. 1, pp. 1–13, 2020.

[3] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile Device Technologies," *IEEE International Electron Devices Meeting (IEDM)*, pp. 32.5.1-32.5.4, 2019.

[4] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

[5] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Proceeding of International Conference on Learning Representations*, pp. 1–14, 2014.

[6] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," *Proceding of the British Machine Vision Conference*, pp. 87.1-87.12, 2016.