2020 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2020, ESPOO, FINLAND

SELF-COMPRESSION IN BAYESIAN NEURAL NETWORKS

Giuseppina Carannante, Dimah Dera, Ghulam Rasool and Nidhal C. Bouaynaya

Rowan University, Department of Electrical and Computer Engineering, Glassboro, NJ
carannang1@rowan.edu, derad6@rowan.edu, rasool@rowan.edu, bouaynaya@rowan.edu

ABSTRACT

Machine learning models have achieved human-level performance on various tasks. This success comes at a high cost of computation and storage overhead, which makes machine learning algorithms difficult to deploy on edge devices. Typically, one has to partially sacrifice accuracy in favor of an increased performance quantified in terms of reduced memory usage and energy consumption. Current methods compress the networks by reducing the precision of the parameters or by eliminating redundant ones. In this paper, we propose a new insight into network compression through the Bayesian framework. We show that Bayesian neural networks automatically discover redundancy in model parameters, thus enabling self-compression, which is linked to the propagation of uncertainty through the layers of the network. Our experimental results show that the network architecture can be successfully compressed by deleting parameters identified by the network itself while retaining the same level of accuracy.

***Index Terms*—** Bayesian deep learning, edge devices, self-compression.

1. INTRODUCTION

Deep Neural Networks (DNNs) have achieved outstanding results in a variety of domains, including computer vision and natural language processing [1, 2]. One of the key factors responsible for these successes is the ability of modern DNNs to learn a large number of parameters [2]. However, an increase in the number of parameters is directly linked to higher computational complexity, the requirement for substantial computational resources, and additional storage for the learned parameters. The availability of dedicated computational hardware (e.g., GPUs) and novel training techniques (e.g., distributed training) have stimulated and supported the growth of DNNs. However, resource-constrained systems, including mobile and edge devices (with limited storage and energy), cannot support these DNNs [3, 4].

Recently, research communities have focused on exploring compression techniques to reduce the memory usage and computational requirements of DNNs. Some well-known current strategies include *pruning* and *quantization* of neural networks. *Pruning* aims at eliminating (most) redundant connections from the network, while *quantization* is characterized by the use of low-precision representation for parameters and activations. Traditionally, most implementations of DNNs used single-precision floating-point format (FP32). However, recently, half-precision (FP16), and low-precision including INT8, INT4, and binary formats have been explored [5].

Various quantization techniques have been proposed to make DNNs perform faster and fit larger networks on edge devices with limited storage capacity and energy budget [6–8]. An unfortunate consequence of quantization is the reduced accuracy, which can be tackled by increasing the network size, performing quantization only on parameters (and not on activations), or fine-tuning and re-training the network. Although many real-world applications can benefit from various quantization techniques, the reduced accuracy offsets the benefits of low precision and significantly increasing training and testing time.

Pruning techniques reduce the total number of parameters and corresponding computations using an "importance" measure. The pruning process is based on the assumption that the trained networks may have redundant parameters that do not contribute to the output of the network and can be removed without compromising the performance [9]. Parameter pruning techniques are generally aimed at introducing sparsity during training or removing parameters or kernels from trained networks using a threshold [10–15]. The importance or threshold values are chosen heuristically. Moreover, inducing sparsity through approximate techniques, such as l_1-norm constraint, may not be effective. Therefore, the pruning process remains a challenging task due to the manual elimination of kernels or setting up a threshold value for the parameter importance [10–15].

In this paper, we propose an automated process of network compression by capitalizing on the recent work in Bayesian neural networks [16, 17]. We leverage uncertainty information available in Bayesian techniques to realize self-compression in convolutional neural networks (CNNs). Consequently, "less important" kernels are automatically identified by the Bayesian CNN and can be removed without compromising the performance accuracy. Our proposed Bayesian CNNs perform at the same accuracy level as vanilla CNNs,

978-1-7281-6662-9/20/$31.00 ©2020 IEEE

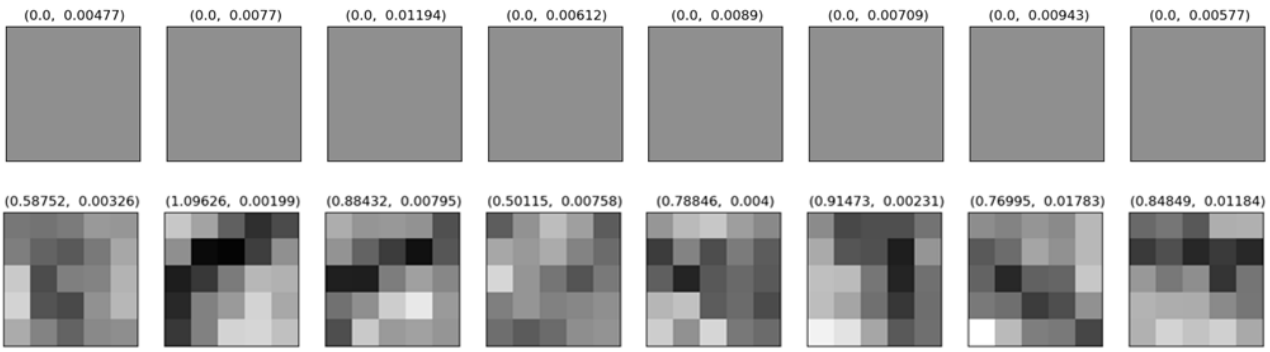

Fig. 1. Sixteen convolutional kernels (randomly selected out of total one hundred) for the Bayesian eVI CNN trained on the MNIST dataset are presented. On the top of each kernel, Frobenius norm and standard deviation obtained from the kernel's posterior distribution are shown. The top row shows eight kernels with Frobenius norm less than 10^{-7}, which can be removed without compromising the network's classification accuracy

.

however, at reduced storage and computational requirements.

2. BAYESIAN SELF-COMPRESSION

2.1. Extended Variational Inference

In the Bayesian framework, the unknown parameters Ω are treated as random variables, and a prior distribution $p(\Omega)$ is defined over these parameters. Using Bayes' Theorem and available dataset $\mathcal{D}$, the posterior distribution $p(\Omega|\mathcal{D})$ and subsequently the predictive distribution are found. Exact Bayesian inference on network parameters is mathematically intractable due to the functional form of these networks and large parameter space [18–20].

Variational Inference (VI) is scalable density approximation technique based on optimization which avoids intractable integration operations [18,20]. VI proposes an approximating distribution $q_{\boldsymbol{\theta}}(\Omega)$ and minimize the Kullback-Leibler (KL) divergence between the proposed distribution and the true posterior $p(\Omega|\mathcal{D})$:

$$\mathbf{KL}(q_{\boldsymbol{\theta}}(\Omega)\big|\big|p(\Omega\big|\mathcal{D})) = \int q_{\boldsymbol{\theta}}(\Omega) \log \frac{q_{\boldsymbol{\theta}}(\Omega)}{p(\Omega)p(\mathcal{D}|\Omega)} d\Omega. \quad (1)$$

Using (1), an optimization problem in the form of well-known Evidence Lower Bound (ELBO) [18] can be formulated as:

$$\mathbf{L}(\boldsymbol{\theta}) = \mathbf{KL}\left(q_{\boldsymbol{\theta}}(\Omega)\big|\big|p(\Omega)\right) - \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{\Omega})}\left[\log\left(p(\mathcal{D}|\Omega)\right)\right]. \quad (2)$$

Recently, Dera *et al.* proposed a VI framework, referred to as the extended Variational Inference (eVI), which propagates the first two moments of the variational posterior through layers of a CNN [16, 17]. Authors defined the tensor normal distribution over kernels as the prior distribution, and used (2) to approximate the posterior distribution of kernels $\mathcal{N}\left(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I}\right)$. The estimated σ^2 provided a measure of confidence or uncertainty attached to the learned kernels during and after the training [16, 17]. Bayesian eVI CNNs showed improved robustness to noise and adversarial attacks [16, 17]. In this work, we establish that the Bayesian eVI approach leads to self-compress and improves network performance.

2.2. Self-Compression with Bayesian eVI

The proposed method can be considered as a "self-aware" compression process for neural networks. During training, a set of kernels is identified as redundant by the Bayesian eVI framework. The redundant kernels are removed through an iterative process, and a compressed network is realized. The Frobenius norm is used to rank each kernel's redundancy or equivalently its importance. The Frobenius norm of a kernel K of size $d \times d$ is defined as:

$$\|K\|_F = \sqrt{\mathrm{Tr}\left(KK^T\right)}, \quad (3)$$

where Tr is the trace operator and ^{T} represents transpose. The Frobenius norm helps us understand the contribution of a kernel to the performance of the Bayesian eVI CNN.

The MNIST handwritten digits dataset and Fashion-MNIST (F-MNIST) dataset were used to train multiple Bayesian and vanilla CNNs [21, 22]. The architecture of all CNNs included one convolution layer followed by the rectified linear unit (ReLU) activation, one max-pooling layer, and one fully-connected layer [23]. The kernel size was set to 5×5 in the convolutional layer of all CNNs. The test accuracy served as the metric to evaluate and compare the performance of Bayesian eVI and vanilla CNNs. In the case of Bayesian CNN, the variances of the predictive distribution provided a measure of the confidence in the classification decision.

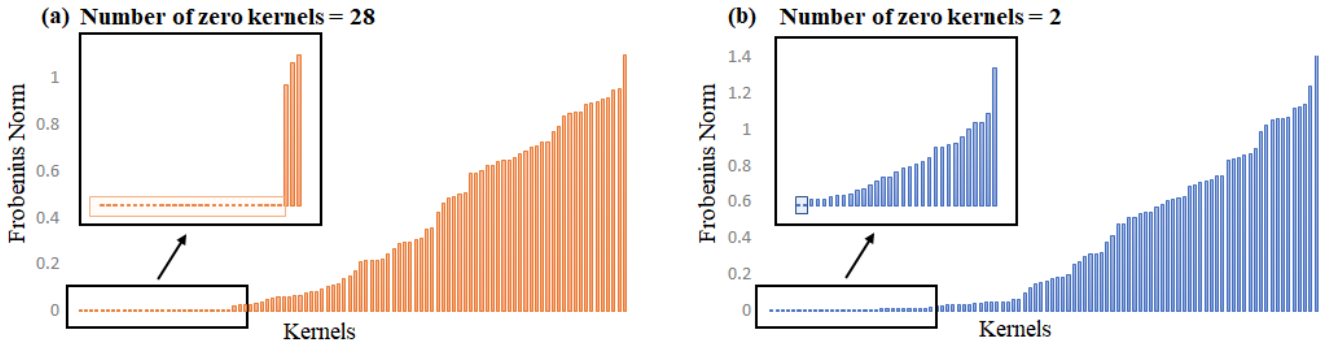

Fig. 2. Frobenius norm of 100 kernels for (a) Bayesian eVI CNN and (b) vanilla CNN trained on the MNIST dataset are presented. The zoomed sub-figures show zero kernels ($\|.\|_F \leq 10^{-7}$). The test accuracy of both CNN was 97%. We consider that a Bayesian CNN can perform at the same accuracy level with 28 zero kernels, which are a candidate for removal in the next training cycle.

We used a different number of kernels in the convolutional layer of both Bayesian and vanilla CNNs. For the MNIST dataset, our first set of CNNs was trained using $N = 100$ kernels. For the F-MNIST, our experiments started with $N = 128$ kernels. The kernels are then reduced based on the kernel's importance metric computed from the Frobenius norm while the variance information is used to evaluate the network's confidence in the values obtained during training for each kernel.

3. RESULTS AND DISCUSSION

3.1. MNIST Dataset

The first set of experiments included training both Bayesian and vanilla CNNs using $N = 100$ kernels. Both CNNs achieved 97% test accuracy. In Fig. 1, we present sixteen randomly selected kernels of a trained Bayesian eVI CNN on MNIST dataset. Fig. 2 shows Frobenius norms of all hundred kernels for both CNNs. We note that for the Bayesian CNN, there are 28 kernels with Frobenius norm equal to zero ($\|K\|_F \leq 10^{-7}$). However, for vanilla CNN, the number is only 2. We refer to the kernels that have $\|K\|_F \leq 10^{-7}$ as *zero kernels*.

In the case of Bayesian eVI CNN, the average variance σ^2 was 0.006 for zero kernels (28 in total) and 0.007 for non-zero kernels. These low variance values for zero and non-zero kernels indicate the network's high level of confidence in these kernels' values. However, no kernel variance information is available for vanilla CNNs. Through experiments and empirical evidence, we found that the reduction of 28 kernels in the Bayesian eVI CNN and two kernels in the vanilla CNN did not affect the test accuracy.

In the second set of experiments using the MNIST dataset, we trained both the Bayesian eVI and vanilla CNNs with a different number of kernels, i.e., $N = 64, 32,$ and, 25. Frobenius norms of kernels for all three cases and both CNNs are presented in Fig. 3. In all three cases, Bayesian CNNs have more zero kernels as compared to vanilla CNNs with comparable or higher accuracy. For the case of $N = 64$, both CNNs have 97% test accuracy. For the Bayesian CNN, 11 zero kernels were identified, however, for the vanilla CNN, the number was only 5. Similarly, for the case of N=25, Bayesian CNN has 96% test accuracy, and vanilla CNN has 94% test accuracy. We identified 4 zero kernels for the Bayesian CNN and none for the vanilla CNN.

3.2. Fashion MNIST Dataset

In Fig. 4, the Frobenius norms of kernels for both CNNs with $N = 128, 64,$ and, 46 are presented. For all three cases, Bayesian eVI CNN achieved the same or better test accuracy; however, only Bayesian eVI CNN was able to identify a set of kernels that could be potentially removed without adversely affecting the test accuracy.

3.3. Iterative Compression of CNNs

Based on experiments with MNIST and F-MNIST datasets and Bayesian eVI and vanilla CNNs, we developed an iterative self-compression technique. We used the number of zero kernels identified by the Bayesian eVI CNN as our guide to remove these kernel in an iterative process. The number of identified zero kernels were removed in the next training cycle. Fig. 5 shows the test accuracy of both Bayesian eVI and vanilla CNNs for a range of the number of kernels N for both datasets. We note that Bayesian eVI CNN retains high accuracy when zero kernels are removed; however, the accuracy of vanilla CNNs starts to drops at a higher rate.

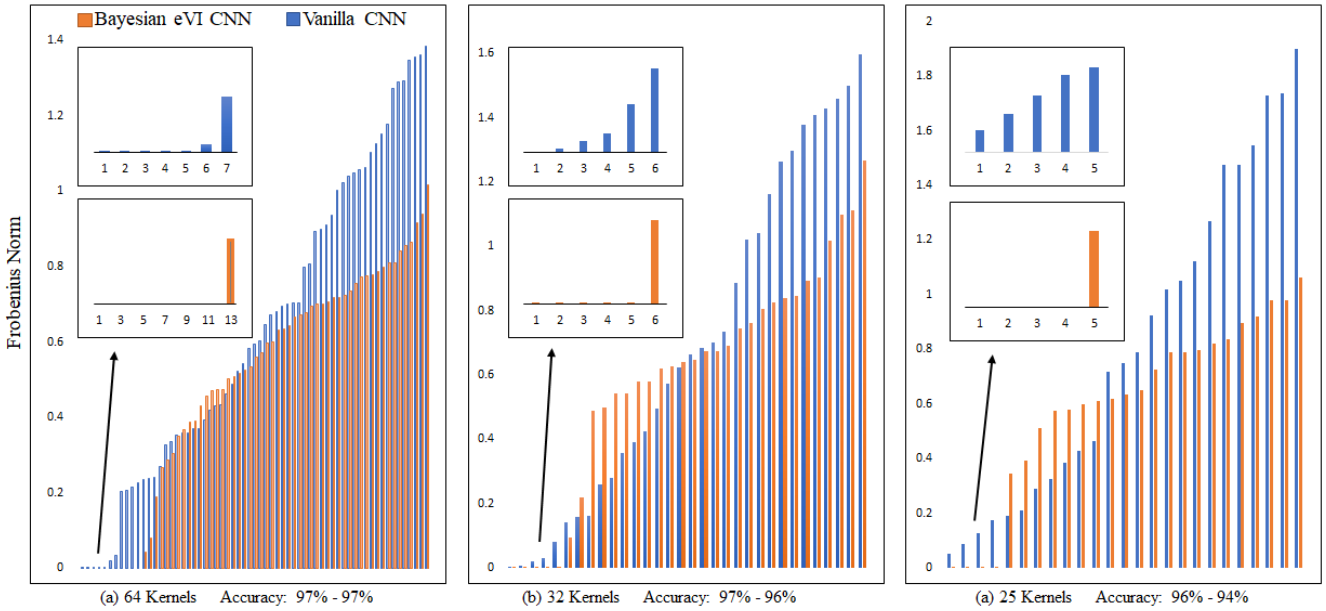

Fig. 3. The Frobenius norms of convolutional kernels for the Bayesian eVI and Vanilla CNNs with $N = 64, 32,$ and, 25 for the MNIST dataset are presented. The zoomed sub-figures show kernels with small Frobenius norms. In all cases, Bayesian eVI CNN can discover zero kernels ($\|.\|_F \leq 10^{-7}$). We consider that a Bayesian eVI CNNs provide a principled way to identify and iteratively remove zero kernels, i.e., self-compression.

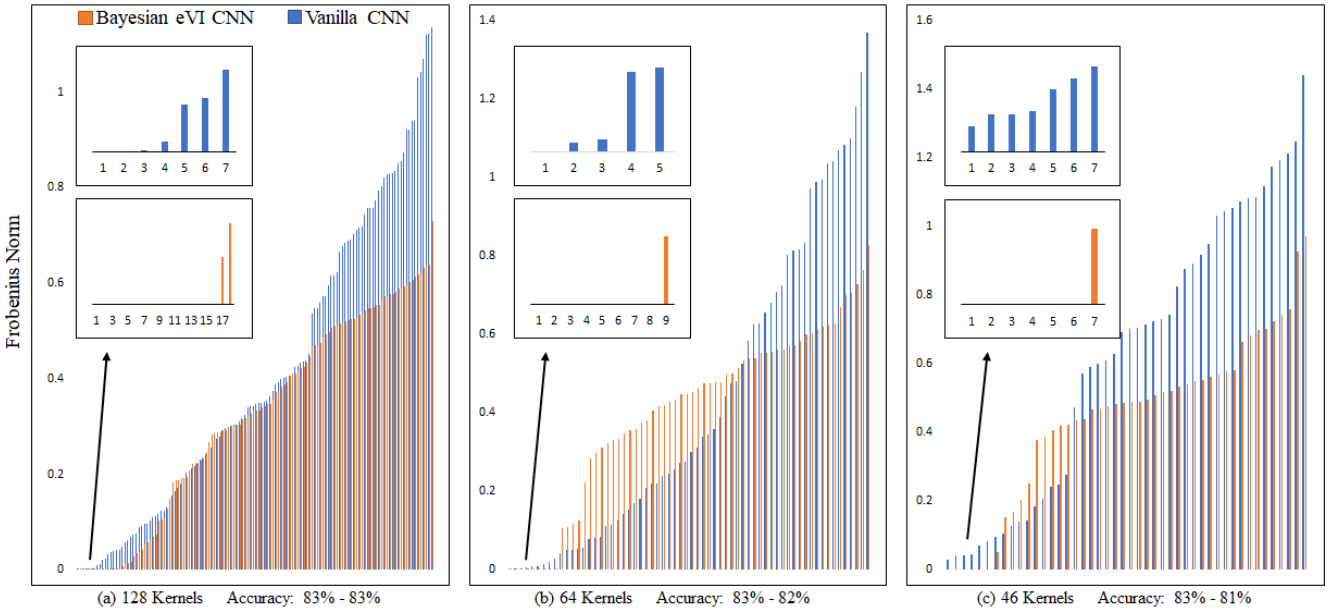

Fig. 4. The Frobenius norms of convolutional kernels for Bayesian eVI and Vanilla CNNs with $N = 128, 64,$ and, 46 for the F-MNIST dataset are presented. The zoomed sub-figures show kernels with small Frobenius norm values. We note that for all cases, Bayesian eVI CNN can discover zero kernels and leads to self-compression.

Bayesian eVI CNNs are able to identity zero kernels (i.e., kernels with $\|.\|_F \leq 10^{-7}$) and provide a principled mechanism for the self-compression, i.e., iteratively removing convolutional kernels that are irrelevant. It is important to highlight that the identified zero kernels had low variance values, i.e., the network showed a high level of confidence in these kernels. The capability to identify zero kernels (with high confidence) is exhibited by Bayesian eVI CNNs and is referred to as the *self-compression*.

The self-compression ability of the Bayesian eVI CNNs stems from propagating variational posterior distribution across layers of a neural network during the training process. The availability of the variance information through the variational posterior during the training process allows Bayesian eVI networks to discover important and unimportant (zero) kernels. Vanilla CNNs, on the other hand, owing to lack of information about uncertainty (or variance) tend to use all available kernels to achieve a high accuracy.

Fig. 5 shows that at the same level of test accuracy, Bayesian eVI CNNs require significantly less number of kernels as compared to vanilla CNNs. The Bayesian eVI CNN performed at the same accuracy level as a vanilla CNN using only half kernels (32 vs. 64 kernels) for the MNIST dataset. Similarly, for the F-MNIST dataset, both Bayesian eVI and vanilla CNNs achieved 83% accuracy. However, the former used only 40 kernels as compared to 72 for the vanilla CNN. Moreover, the process of discovery of unimportant (zero) kernels is inherently part of Bayesian eVI CNNs. On the other hand, the process of compression or pruning for vanilla CNNs requires manual selection of threshold and possibly a compromise on the test accuracy and performance.

3.4. The Size of the Parameter Space

Previously proposed Bayesian approaches for neural networks resulted in a larger parameter space (double the number of parameters as compared to vanilla network) due to the probability distribution defined over the parameters [16, 19]. However, the recently proposed Bayesian eVI CNNs require only one additional variance parameter σ^2 for each convolutional kernel owing to the use of diagonal tensor normal distribution [16, 17]. A Bayesian eVI CNN has $(d^2 + 1)N$ parameters as compared to a vanilla CNN that has d^2N parameters, where d is the size of a kernel and N represents the total number of convolutional kernels.

Storage-wise, the additional number of parameters does not appear to be an issue given the *self-compression* capability. For example, in our implementation (with kernel size $d = 5$), the storage for the convolution parameters after compression for both models is shown in table 1.

Table 1. Storage (KB) after Compression

Dataset	Bayesian eVI CNN	Vanilla CNN
MNIST	3.25	6.25
F-MNIST	4.06	7.03

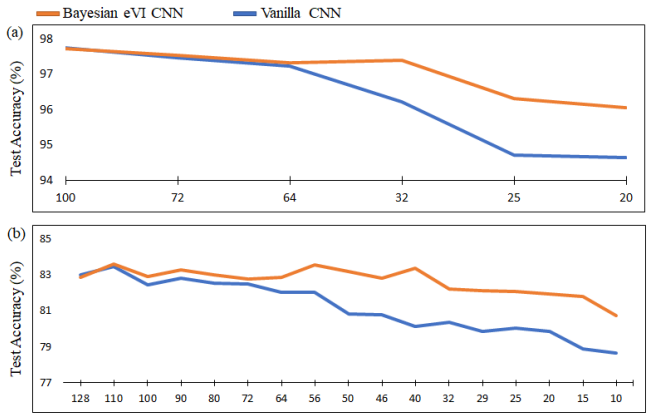

Fig. 5. Test accuracy vs. the number of convolutional kernels N for the Bayesian eVI CNN (orange) and a vanilla CNN (blue) are presented. We note that the Bayesian eVI CNN is able to perform at the same accuracy level as that of vanilla CNN using almost half the number of convolutional kernels (32 vs. 64 for the MNIST dataset and 40 vs. 72 for the Fashion-MNIST dataset). (a) Test accuracy for the MNIST Dataset. (b) Test accuracy for the F-MNIST dataset.

4. CONCLUSION

The high performance of current machine learning algorithms has associated high computational costs and storage requirements. We exploit the uncertainty information inherent in Bayesian neural networks and propose a method for self-compression of convolutional neural networks. We used the recently proposed extended Variational Inference (eVI) framework that offers advantages of Bayesian neural networks, albeit at a minimal increase in the number of parameters. We showed that Bayesian eVI CNNs were able to identify redundant kernel during training, which can be removed in an iterative process to realize self-compression. The Bayesian eVI CNNs were able to perform at the same accuracy level as that of vanilla CNNs using almost half the number of convolutional kernels. The proposed process of self-compression is part of the training, and resulting Bayesian eVI CNNs are ready for deployment on the edge devices.

5. ACKNOWLEDGEMENT

This work was supported by the National Science Foundation Awards NSF ECCS-1903466. NSF CCF-1527822, NSF OAC-2008690. Giuseppina Carannante is supported by the US Department of Education through a Graduate Assistance in Areas of National Need (GAANN) program Award Number P200A180055. We are also grateful to UK EPSRC support through EP/T013265/1 project NSF-EPSRC: ShiRAS. Towards Safe and Reliable Autonomy in Sensor Driven Systems.

6. REFERENCES

[1] Tero Karras, Samuli Laine, and Timo Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever, "Language Models are Unsupervised Multitask Learners," *OpenAI Blog*, vol. 1, no. 8, pp. 9, 2019.

[3] Rajat Raina, Anand Madhavan, and Andrew Y Ng, "Large-Scale Deep Unsupervised Learning using Graphics Processors," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 873–880.

[4] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al., "Large Scale Distributed Deep Networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[5] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David, "Binaryconnect: Training Deep Neural Networks with Binary Weights during Propagations," in *Advances in neural information processing systems*, 2015, pp. 3123–3131.

[6] Alexander Goncharenko, Andrey Denisov, Sergey Alyamkin, and Evgeny Terentev, "Fast Adjustable Threshold for Uniform Neural Network Quantization," *arXiv preprint arXiv:1812.07872*, 2018.

[7] Yoni Choukroun, Eli Kravchik, and Pavel Kisilev, "Low-Bit Quantization of Neural Networks for Efficient Inference," *arXiv preprint arXiv:1902.06822*, 2019.

[8] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, "Xnor-Net: Imagenet Classification using Binary Convolutional Neural Networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[9] Song Han, Huizi Mao, and William J Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv preprint arXiv:1510.00149*, 2015.

[10] Vadim Lebedev and Victor Lempitsky, "Fast Convnets using Group-Wise Brain Iamage," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2554–2564.

[11] Hao Zhou, Jose M Alvarez, and Fatih Porikli, "Less is More: Towards Compact CNNs," in *European Conference on Computer Vision*. Springer, 2016, pp. 662–677.

[12] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li, "Learning Structured Sparsity in Deep Neural Networks," in *Advances in neural information processing systems*, 2016, pp. 2074–2082.

[13] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung, "Structured Pruning of Deep Convolutional Neural Networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, pp. 32, 2017.

[14] Adam Polyak and Lior Wolf, "Channel-Level Acceleration of Deep Face Representations," *IEEE Access*, vol. 3, pp. 2163–2175, 2015.

[15] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf, "Pruning Filters for Efficient Convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[16] Dimah Dera, Ghulam Rasool, Nidhal C. Bouaynaya, Adam Eichen, Stephen Shanko, Jeff Cammerata, and Sanipa Arnold, "Bayes-SAR Net: Robust SAR Image Classification with Uncertainty Estimation using Bayesian Convolutional Neural Network," in *Proceedings of the IEEE International Radar Conference*, 2020.

[17] Dimah Dera, Ghulam Rasool, and Nidhal Bouaynaya, "Extended Variational Inference for Propagating Uncertainty in Convolutional Neural Networks," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2019, pp. 1–6.

[18] Alex Graves, "Practical Variational Inference for Neural Networks," in *Advances in neural information processing systems*, 2011, pp. 2348–2356.

[19] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra, "Weight Uncertainty in Neural Networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning, (ICML)*, 2015, vol. 37, pp. 1613–1622.

[20] Kumar Shridhar, Felix Laumann, Adrian Llopart Maurin, and Marcus Liwicki, "Bayesian Convolutional Neural Networks," *arXiv preprint arXiv:1806.05978*, 2018.

[21] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-mnist: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[22] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[23] Vinod Nair and Geoffrey E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010, pp. 807–814.