

Analyzing the computational cost of EffBaGAN in data-scarce remote sensing scenarios

Nicolás Vilela-Pérez^{*†}, Dora B. Heras^{*†}, and Francisco Argüello[†]

^{*}Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, Spain

[†]Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, Spain

Abstract—Generative Adversarial Networks (GANs) applied to remote sensing classification have shown strong performance in scenarios with limited labeled data. However, their deployment on resource-constrained edge computing devices, such as autonomous aerial vehicles or satellites, is often limited by their high computational cost. This work presents a preliminary study on the impact of computational optimization techniques applied to EffBaGAN, aiming to facilitate the projection over edge computing devices. EffBaGAN is a neural architecture that integrates EfficientNet and a Balancing GAN (BAGAN) to tackle the classification of multispectral images under conditions of data scarcity and class imbalance. We evaluate the effect of several optimization techniques, including Automatic Mixed Precision (AMP), Post-Training Quantization (PTQ), Quantization-Aware Training (QAT), and structured magnitude-based pruning, on the computational efficiency of the EffBaGAN neural architecture when executing on a multicore CPU + GPU system. Experimental results on high-resolution multispectral datasets for biodiversity assessment show that the network can be significantly optimized without degrading classification accuracy. In particular, PTQ and QAT achieved a network size reduction of over 72%, while structured pruning successfully removed 30% of the parameters and floating-point operations. Moreover, we show that suitable combinations of optimization strategies, such as QAT combined with pruning, can substantially reduce computational overhead, thereby improving the feasibility of deploying EffBaGAN in resource-constrained environments.

Index Terms—High Performance Computing, edge computing, multispectral, classification, deep learning, data scarcity, Generative Adversarial Network, Automatic Mixed Precision, quantization, pruning

I. INTRODUCTION

Remote sensing multi- and hyperspectral image classification using deep learning techniques has proven highly effective [1][2]. Nevertheless, their use implies two important and correlated challenges: the high computational cost and the need for a large amount of labeled data for proper training [3]. Parallel computing platforms are typically used to reduce computational costs, particularly in heterogeneous systems [4].

Some classification problems in remote sensing are affected by data scarcity (low number of labeled samples) and class

imbalance. This is the case for applications such as small-scale land-cover mapping and biodiversity assessment [5][6]. In these cases, standard deep learning techniques fail to train the network correctly, resulting in overfitting, and adaptations or even new networks are necessary [7]. To mitigate data scarcity, Generative Adversarial Networks (GANs) [8] have emerged as a robust augmentation technique, capable of synthesizing novel samples from an estimated data distribution. In contexts of severe class imbalance, the Balancing GAN (BAGAN) [9] is particularly effective for classification, offering a specialized architecture that stabilizes training and restores equilibrium in small, imbalanced datasets.

This work studies the Efficient Balancing Generative Adversarial Network (EffBaGAN) [10], a method that integrates the efficiency of EfficientNet with a BAGAN framework for multispectral image classification under conditions of data scarcity and class imbalance. EffBaGAN incorporates a residual EfficientNet-based generator and a reduced EfficientNet-based discriminator, thereby lowering network complexity and computational cost. This classification method achieves high accuracy while maintaining a lower computational cost than other methods adapted to data scarcity and class imbalance. Its efficiency largely comes from the use of the EfficientNet core building block, the Mobile Inverted Bottleneck (MBConv), originally introduced in MobileNetV2 [11]. This block relies on depthwise separable convolutions to reduce computational cost by limiting expensive channel-wise interactions.

Recent advances highlight the importance of heterogeneous and hardware-aware optimization to achieve real-time performance on platforms such as autonomous aerial vehicles and on-orbit satellites [12][13]. Although EffBaGAN, as designed based on EfficientNet architecture [14], reducing the computational cost while maintaining accuracy, represents an improvement in terms of computational cost, its resource requirements may still be prohibitive depending on the target deployment platform. To further improve its suitability for resource-constrained environments, this work focuses on analyzing the computational cost of EffBaGAN and applying straightforward optimization strategies to enable efficient execution on a simple heterogeneous computing architecture. The considered setup consists of a multicore CPU with a single GPU. This analysis supports the exploration of EffBaGAN deployment on edge devices, where strict energy and memory constraints require highly efficient and lightweight neural networks.

This work was supported by project PID2022-141623NB-I00 funded by MCIN/AEI/10.13039/501100011033, and by Xunta de Galicia – Consellería de Educación, Ciencia, Universidades e Formación Profesional (Centro de investigación de Galicia accreditation 2024-2027 ED431G-2023/04 and Reference Competitive Group accreditation ED431C-2022/16), both co-funded by ERDF/EU. The work of Nicolás Vilela-Pérez was supported by the Xunta de Galicia – Consellería de Educación, Ciencia, Universidades e Formación Profesional, under Grant ED481A-2025-015.

Overall, this work analyzes the computational cost of EffBaGAN for classification tasks under conditions of data scarcity and class imbalance. The main objective is to evaluate and apply optimization techniques that improve computational efficiency by reducing hardware resource utilization, thereby lowering overall computational cost. Specifically, this study evaluates the individual and combined impact of Automatic Mixed Precision (AMP), quantization, and structured pruning on the computational efficiency of EffBaGAN.

II. EFFBAGAN

EffBaGAN [10] is a generative classification method built upon the BAgAN architecture. It acts as both a data augmentation technique and a classifier. The core innovation of EffBaGAN lies in integrating the EfficientNet architecture [14] into both the generator and the discriminator, replacing standard convolutional blocks with MBConv (Mobile Inverted Bottleneck Convolution) blocks to perform convolutions with reduced computational cost.

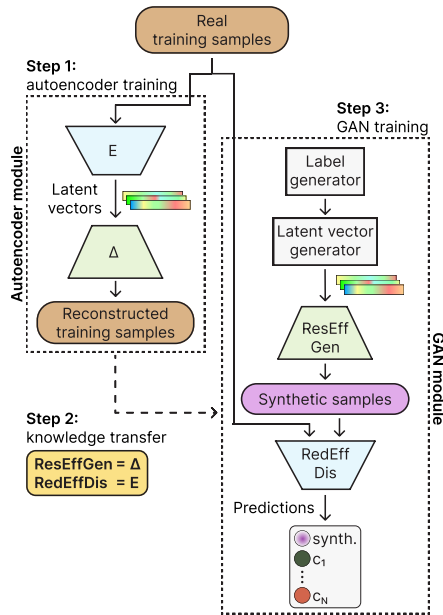


Fig. 1. Neural network architecture of EffBaGAN [10].

Its architecture, as shown in Figure 1, consists of two main components: ResEffGen, a generator with an EfficientNet-based residual path for synthesizing high-quality samples, and RedEffDis, a reduced EfficientNet-B0 discriminator used as the classifier. Training follows a three-step procedure: (1) autoencoder training to learn data distribution, (2) knowledge transfer to initialize the GAN, and (3) adversarial training. Input samples are obtained using Waterpixels superpixel segmentation to group pixels into homogeneous regions, considering all spectral components, thereby reducing computational cost.

III. REDUCING THE COMPUTATIONAL COST

In this section, different straightforward optimization strategies aimed at reducing the hardware resource utilization of EffBaGAN are presented.

A. Automatic Mixed Precision

AMP [15] combines single-precision (FP32) operations, required for numerical stability, with half-precision (FP16) to reduce memory bandwidth usage. This optimization dynamically selects the appropriate precision for each operation during training, preserving convergence and achieving final accuracy comparable to that of the baseline network without optimization.

B. Quantization

It reduces the precision of network parameters to lower-precision 8-bit integers (INT8) to decrease network size. This optimization is particularly effective during inference, reducing both network size and power consumption, enabling the deployment of complex networks on resource-constrained devices with minimal impact on accuracy. Two widely used quantization strategies are considered:

- Post-Training Quantization (PTQ) [16]: this approach applies quantization after the training process is completed. It is simple and fast to implement, but it may degrade accuracy and require an additional calibration step.
- Quantization-Aware Training (QAT) [17]: it simulates quantization effects during training by injecting quantization noise into the forward pass, allowing the network to adapt to reduced-precision representations. As a result, QAT typically increases training time.

In this work, both PTQ and QAT are evaluated to achieve a significant reduction in network size while preserving classification accuracy.

C. Pruning

it is the process of permanently removing redundant or non-informative connections from a neural network, effectively transforming a dense network into a more compact one by reducing network size, parameter count, and the number of floating-point operations (FLOPs). By eliminating unnecessary computations, pruning [18] reduces computational cost while preserving baseline accuracy.

Pruning methods can be categorized into unstructured pruning, which removes individual weights, and structured pruning, which removes entire groups of parameters, such as filters or channels. The effectiveness of pruning depends on the criterion used to identify removable components; common approaches include magnitude-based, sensitivity-based, and activation-based criteria.

In this work, structured filter pruning is applied to convolutional layers after training. An L1-norm magnitude-based criterion is used to estimate filter importance, and the least significant 25% of channels are removed under the assumption that they contribute minimally to the network's representational capacity. Unlike unstructured pruning, this approach physically

removes filters and their corresponding feature maps, leading to tangible reductions in network size, parameter count, and FLOPs. A fine-tuning stage is subsequently performed to mitigate potential accuracy loss.

IV. EXPERIMENTS

In this section, EffBaGAN-Small [10], a computationally efficient variant of EffBaGAN, is evaluated for classification performance and computational cost. EffBaGAN-Small baseline (without optimizations) is compared against multiple optimized configurations obtained by selectively enabling the proposed optimization techniques.

A. Experimental setup

Eight large, high-spatial resolution multispectral images of natural areas with dense vegetation in Galicia (Spain), previously used in [10], are used in the experiments. These scenes range in size from 6722×6689 to 22116×10718 pixels and provide between 1461 and 61393 labeled samples. The datasets have a spatial resolution of 10 cm/px, and the inputs to the network consist of 32×32 patches extracted from these scenes with 5 spectral bands corresponding to wavelengths of 475 nm (blue), 560 nm (green), 668 nm (red), 717 nm (red-edge), and 840 nm (near-infrared). In addition, all data were normalized to the range $[-1, 1]$, and the training and validation sets comprised 15% and 5% of the samples, respectively, leaving the remaining 80% for the test set.

Regarding evaluation metrics, classification performance is assessed using Overall Accuracy (OA), Average Accuracy (AA), and Cohen's kappa coefficient (κ). Additionally, training and test times are measured. Computational cost is evaluated in terms of peak GPU memory allocation during training, network size, number of parameters, and number of FLOPs required for a single forward pass with a batch size of 1.

All the experiments were conducted on a computer cluster using a node equipped with two AMD EPYC 7543 CPUs with 32 cores each, 256 GB of RAM, and two NVIDIA Ampere A100 GPUs, although only one CPU and GPU were used for all experiments. Execution times were measured end-to-end for training and inference, and peak GPU memory during training was captured via PyTorch's native maximum memory allocator tracking.

The training protocol and most hyperparameter settings follow those described in [10], with the SiLU activation function replaced by the Hardswish activation function [19]. SiLU is not currently supported by standard quantization backends due to the difficulty of efficiently mapping its sigmoid component to fixed-point arithmetic. Hardswish is a piecewise-linear approximation that preserves the non-linearity of SiLU while remaining compatible with hardware-accelerated integer inference. Hardswish is consistently used across all experiments. The network is trained for 600 epochs using a batch size of 64, a learning rate of approximately $5 \cdot 10^{-4}$, latent vectors of size 32, and a dropout probability of 5%. For PTQ, static calibration utilizes 64 batches from the training set. Regarding QAT, it inserts fake quantization nodes over the final 20%

of the total training epochs, using a three-phase schedule: standard training (first 80%), observer-enabled calibration (80-90%), and a final stabilization phase with frozen observers (last 10%). Structured pruning is applied one-shot, followed by fine-tuning for 30% of the original training epochs (using the 20% of the initial LR, and a StepLR scheduler decaying by $\gamma = 0.5$ every 25% of the fine-tuning epochs).

B. Experimental results

Table I reports OA, AA, κ (accuracy metrics), along with training and test times, and peak GPU memory allocation during training, network size, number of parameters, and number of FLOPs for a single forward pass (computational metrics) for the EffBaGAN-Small baseline (without optimizations) and for the different optimization techniques described in Section III, evaluated both individually and in selected combinations. Results are averaged over five runs across eight datasets (described in Section IV-A). Mean and standard deviation are reported for accuracy metrics and times, while the computational cost metrics are constant across runs for a given optimization configuration.

The table shows that applying AMP reduces peak GPU memory allocation by 10% during training by storing a significant portion of computations in FP16, thereby lowering memory bandwidth requirements. However, a slight increase in training time is also observed due to the overhead introduced by dynamic loss scaling and runtime casting operations.

Regarding quantization techniques (columns PTQ and QAT in the table), both achieved a network size reduction exceeding 72% by compressing weights to INT8. Nevertheless, QAT introduces a notable training overhead for this small network due to the insertion of fake quantization nodes into the computational graph, increasing both training time and peak GPU memory allocation during training. In contrast, PTQ introduces no training overhead, as it relies on post-training calibration using a representative subset of the data to map activation ranges to INT8 without modifying the learned weights or degrading accuracy.

The results for pruning show a successful reduction of more than 30% in the number of parameters and FLOPs without degrading accuracy metrics. This robustness is attributed to the fine-tuning stage performed after pruning, which enables the remaining filters to adapt and compensate for the removed ones.

Finally, when combining optimization techniques, PTQ with pruning results in greater accuracy degradation than QAT combined with pruning. The degradation observed with PTQ and pruning can be attributed to the aggressive nature of both techniques: PTQ lacks a fine-tuning phase to compensate for the structural changes it introduces, and pruning inherently reduces the network's representational capacity. In contrast, the combination of QAT and pruning benefits from training-aware quantization and subsequent fine-tuning, enabling the network to better adapt and converge. A per-class analysis for the imbalanced datasets used in this paper shows that for some spectrally overlapping classes, PTQ is unable to separate

TABLE I
RECORDED METRICS FOR ALL THE DATASETS WITH THE DIFFERENT OPTIMIZATION TECHNIQUES APPLIED TO EFFBaGAN-SMALL.

Metrics	Baseline	AMP	PTQ	QAT	Pruning	PTQ + Pruning	QAT + Pruning
OA (%)	95.92 ± 0.49	95.95 ± 0.46	95.35 ± 1.23	95.85 ± 0.40	96.50 ± 0.10	74.34 ± 14.77	89.36 ± 3.86
AA (%)	87.49 ± 1.29	87.03 ± 2.28	86.62 ± 2.27	87.12 ± 2.15	88.40 ± 0.83	65.97 ± 8.87	73.82 ± 5.88
κ (%)	93.26 ± 0.77	93.28 ± 0.78	92.31 ± 1.97	93.14 ± 0.66	94.19 ± 0.17	61.81 ± 17.20	81.77 ± 6.07
Training time (s)	3 862.95 ± 146.80	3 984.05 ± 51.81	3 750.86 ± 94.97	6 581.87 ± 809.12	4 294.92 ± 159.69	4 795.22 ± 221.09	4 809.46 ± 208.45
Test time (s)	893.63 ± 14.22	952.73 ± 15.21	964.41 ± 28.24	1 005.63 ± 32.48	1 035.85 ± 38.93	1 112.05 ± 69.60	1 167.69 ± 42.45
Peak mem. alloc. (MiB)	1 320.91	1 183.48 (↓ 10.40%)	1 320.91 (0.00%)	1 431.44 (↑ 8.37%)	1 320.91 (0.00%)	1 320.91 (0.00%)	1 320.91 (0.00%)
Network size (MiB)	3.78	3.78	1.03 (↓ 72.85%)	0.95 (↓ 74.91%)	3.22 (↓ 14.99%)	1.27 (↓ 66.32%)	1.26 (↓ 66.81%)
Parameters (#)	972 235	972 235	972 235 (0.00%)	966 475 (↓ 0.59%)	677 315 (↓ 30.33%)	675 443 (↓ 30.53%)	674 219 (↓ 30.65%)
FLOPs (#)	27 104 608	27 104 608	26 619 232 (↓ 1.79%)	26 619 232 (↓ 1.79%)	18 339 720 (↓ 32.34%)	17 939 592 (↓ 33.81%)	17 939 592 (↓ 33.81%)

Accuracy metrics and execution times are presented as the mean and standard deviation computed across all experiments. All reported computational cost metrics correspond to the inference phase; therefore, only the discriminator network is considered when computing these metrics. The percentage values reported below each computational cost metric indicate the relative difference with respect to the baseline configuration.

them. The network struggles to distinguish some classes. For example, the minority class concrete has low accuracy and is often confused with asphalt, while similar vegetation classes are not well separated. Conversely, QAT's training-aware adaptation allows the remaining parameters to readjust to the structural capacity loss. QAT resolves failures across all flora classes by restricting remaining misclassifications to minor intra-vegetation confusions.

Overall, QAT with pruning achieved the largest reductions in network size, parameter count, and FLOPs (66.81%, 30.65%, and 33.81%, respectively). The observed accuracy decline stems from the cumulative impact of restricted numerical precision (quantization) and reduced representational capacity (pruning), creating compounded constraints that fine-tuning could partially mitigate. However, this approach successfully avoided the severe accuracy degradation observed with PTQ pruning, remaining effective even under severe class imbalance, with some datasets containing classes that represent as little as 0.30% of the labeled samples, while others account for up to 69.75%.

V. CONCLUSIONS

This work optimizes the computational cost of EffBaGAN, a generative classification method designed for multispectral image classification in data-scarce scenarios. The study evaluates the effectiveness of several widely used deep learning optimization techniques when applied to this specialized generative architecture. In particular, we examined:

- The impact of Automatic Mixed Precision (AMP) on training efficiency.
- The effect of quantization strategies, comparing Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT).
- The application of structured network pruning.
- The interaction between these techniques, identifying combinations that provide cumulative efficiency gains and those that lead to performance degradation.

Experimental results demonstrated that EffBaGAN can be significantly compressed and optimized without compromising

its classification performance on imbalanced datasets. Specifically, AMP reduces peak GPU memory allocation by approximately 10%, alleviating memory bandwidth requirements during training with negligible impact on convergence. Regarding storage, quantization is highly effective at reducing network size, with both PTQ and QAT achieving reductions of more than 72%. However, PTQ is better suited to this architecture, as QAT introduces substantial training overhead due to the insertion of fake quantization operations. Finally, structured magnitude-based pruning successfully reduces the network's parameter count and FLOPs by 30%, with subsequent fine-tuning playing a key role in preserving the network's representational capacity and classification accuracy.

When combining strategies, QAT with pruning yields the largest reductions in network size, parameters, and FLOPs. Although the compounded constraints of precision and reduced capacity led to a drop in accuracy, the integrated fine-tuning phase successfully mitigated the severe degradation, highlighting the importance of training-aware adaptation when aggressive compression techniques are applied. Overall, this study confirms that standard deep learning optimization techniques can be effectively applied to specialized generative networks such as EffBaGAN.

Although EffBaGAN was originally designed for small-scale remote sensing tasks, the proposed optimizations further reduce computational requirements, enabling deployment on resource-constrained edge computing platforms where memory and energy constraints are critical. Deployment strategies require carefully balancing hardware and training constraints: we recommend QAT combined with pruning for strictly storage-limited edge environments, given its maximal size reduction. However, because this combination adds training overhead and degrades accuracy, either standalone PTQ or pruning is optimal for preserving maximum classification accuracy, or when operating under strict computational or time constraints during training. Future work will focus on evaluating these techniques on hardware-specific inference engines and embedded platforms to further improve real-time performance and deployment efficiency.

REFERENCES

- [1] M. Paoletti, J. Haut, J. Plaza, and A. Plaza, "Deep learning classifiers for hyperspectral imaging: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 158, pp. 279–317, 2019.
- [2] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, and J. A. Benediktsson, "Deep learning for hyperspectral image classification: An overview," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 9, pp. 6690–6709, Sep. 2019.
- [3] N. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The Computational Limits of Deep Learning," in *Ninth Computing within Limits 2023*. LIMITS, jun 14 2023, <https://limits.pubpub.org/pub/wm1lwjce>.
- [4] D. Tuia, K. Schindler, B. Demir, X. X. Zhu, M. Kochupillai, S. Džeroski, J. N. van Rijn, H. H. Hoos, F. D. Frate, M. Datcu, V. Markl, B. L. Saux, R. Schneider, and G. Camps-Valls, "Artificial intelligence to Advance Earth Observation: A review of models, recent trends, and pathways forward," *IEEE Geoscience and Remote Sensing Magazine*, vol. 13, no. 4, pp. 119–141, 2025.
- [5] Á. G. Dieste, F. Argüello, and D. B. Heras, "ResBaGAN: A residual balancing GAN with data augmentation for forest mapping," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 16, pp. 6428–6447, 2023.
- [6] V. Barreiro, D. B. Heras, and F. Argüello, "EMViT-DDPM: An equilibrium-based ViT diffusion framework for data augmentation in multispectral land cover classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 18, pp. 25 923–25 938, 2025.
- [7] A. Dash, J. Ye, and G. Wang, "A review of generative adversarial networks (GANs) and its applications in a wide variety of disciplines: From medical to remote sensing," *IEEE Access*, vol. 12, pp. 18 330–18 357, 2024.
- [8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
- [9] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi, "BAGAN: Data augmentation with balancing GAN," in *International Conference on Machine Learning*, 2018. [Online]. Available: <https://research.ibm.com/publications/bagan-data-augmentation-with-balancing-gan>
- [10] N. Vilela-Pérez, D. B. Heras, and F. Argüello, "EffBaGAN: An efficient balancing GAN for Earth observation in data scarcity scenarios," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 18, pp. 2477–2496, 2025.
- [11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [12] J. López-Fandiño, D. B. Heras, and F. Argüello, "Using heterogeneous computing and edge computing to accelerate anomaly detection in remotely sensed multispectral images," *The Journal of Supercomputing*, vol. 80, no. 9, pp. 12 543–12 563, jun 2024. [Online]. Available: <https://doi.org/10.1007/s11227-024-05918-z>
- [13] R. Del Prete, P. K. Thind, A. Mazzeo, M. Whitley, L. Papa, N. Longépé, and G. Meoni, "Optimizing deep learning models for on-orbit deployment through neural architecture search," *Scientific Reports*, vol. 15, no. 1, p. 37783, oct 2025. [Online]. Available: <https://doi.org/10.1038/s41598-025-21467-8>
- [14] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6105–6114. [Online]. Available: <https://proceedings.mlr.press/v97/tan19a.html>
- [15] S. Narang, G. Diamos, E. Elsen, P. Micikevicius, J. Alben, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh *et al.*, "Mixed precision training," in *Int. Conf. on Learning Representation*, 2017.
- [16] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *CoRR*, vol. abs/1806.08342, 2018. [Online]. Available: <http://arxiv.org/abs/1806.08342>
- [17] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.
- [18] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.
- [19] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for MobileNetV3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.