Reducing Computational Complexity in CNNs: A Focus on VGG19 Pruning and Quantization

Md. Mijanur Rahman[1], Anik Datta[2], Md. Sabiruzzaman[3], Md Samim Ahmed Bin Hossain[4]
Assistant Professor, Dept. of CSE, Southeast University, Dhaka, Bangladesh[1]
Dept. of CSE, Southeast University, Dhaka, Bangladesh[2,3,4]

***Abstract*—The Convolutional Neural Network (CNN) models are effective in computer vision strategies and have gained popularity due to their strong performance in visual tasks. Nevertheless, models with architectures such as VGG19 are expensive in terms of computational resources and require huge memory, which limits their usage on low-end devices. The study examines how efficiency can be increased in the model VGG19 by using model compression techniques, like pruning (structured and unstructured) and quantization (8-bit and 4-bit Quantization-Aware Training - QAT). The efficiency of the individual compression approaches was tested by thoroughly exploring the VGG19 with the MNIST, CIFAR-10, and Oxford-IIIT Pet datasets. Each model was evaluated against the baseline based on measures of accuracy, model size, inference time, and complexities of the model, CPU usage, and memory usage. The applied QAT approach reduced the model size by 75% with a drop in computational cost across all methods. In addition, the 8-bit quantitative assessment allowed for substantial system compression alongside increased speed delivery with minimal impact on accuracy. The highest compression and sparsity achieved by 4-bit QAT was 48%, which was not effective as it reduced accuracy on complex datasets, with additional computational overhead on T4 GPU. Structured pruning resulted in faster inference, but unstructured pruning also demonstrated a good result in retaining accuracy and even improving it. To simplify the VGG19 structure, pruning and quantization mechanisms are suggested in order to simplify the architecture to implement the model on edge devices sufficiently, without compromising prediction performance.**

Keywords*—*VGG19 Model optimization; model compression; pruning, quantization; structured pruning; unstructured pruning; memory management; quantization-aware training; 8-bit; 4-bit

I. Introduction

CNNs have experienced significant advancements during the last few years to solve a wide range of computer vision [1] problems that include simple image classification [2] tasks alongside complex object detection [3] problems. The VGG19 CNN model stands among multiple models as a significant impactor because it significantly contributes to deep learning by accurately addressing challenging image classification tasks [4]. Although efficient, direct deployment of VGG19 on resource-limited devices, i.e., smartphones and edge computing setups, is infeasible because of its enormous parameter size and excessive computational demands in terms of floating-point operations per second (FLOPs) [5]. Model compression methods, which include pruning and quantization, have addressed these challenges effectively. Quantization achieves maximum savings in memory usage plus computation requirements through its usage of weight and activation bit widths, which makes it ideal for devices with minimal resources [6], [7]. Model performance enhancement through pruning strategies involves connection reduction for both size reduction and computational complexity reduction [8].

Although the advancement in model compression has been remarkable, most of the current research mainly concentrates on large-scale data and high-resource computing settings. However, their performance is hardly well-tested in environments where datasets are comparatively small and computer resources are modest. The testing of their performance needs more investigation in situations that use small datasets alongside minimal computer resources. Different pruning techniques like layer-wise pruning [9], structured pruning [10], activation-based pruning [11], and different quantization techniques like mixed-precision and post-training quantization [12], [13], have been studied separately on different models. There is limited modeling understanding of where the efficiency of these techniques is jointly implemented on the VGG19 model with constrained availability of resources. This study attempts to solve this gap by making a thorough evaluation of pruning and quantization techniques performed on VGG19.

To address this gap, the following research questions are asked:

- RQ1: What level of impact do various pruning methods (structured versus unstructured) have on the performance considering trade-offs of VGG19 on resource-limited datasets?
- RQ2: Which bit-width (8-bit QAT versus 4-bit QAT) of the encoded quantization is the best to keep the accuracy and simultaneously maximize the compression?
- RQ3: What are the performance of these optimization methods on data with different complexities (MNIST, CIFAR-10, Oxford-IIIT Pet)?

The gap is to be filled by comparing the accuracy, size, and processing costs of models, then evaluating the weight distribution and parameter sparsity norms [14]. To the best of our knowledge, such an in-depth, visual comparison of VGG19 with different reporting conditions, namely structured and unstructured pruning and 8-bit and 4-bit QAT.

This study compares and contrasts these approaches and studies their behavior under both isolated and combined settings and resource-limited data. The key contributions of this study are: i) an empirical evaluation of many pruning and quantization techniques on VGG19 and ii) their analysis of utility-by-constraints of usefulness under resource-constrained conditions, hence generalizing beyond the large-scale studies

done typically in the literature on model compression techniques.

The structural framework of the study is as follows: the literature review is given in Section II, methodology in Section III, the experimental results in Section IV, detailed discussion of the results in Section V, the future directions in Section VI, and conclusion with final remarks in Section VII.

II. Related Work

Research into deep learning model improvement seeks to enhance VGG19 and its contemporaries. Different research studies investigated pruning alongside quantization and transfer learning methods to minimize complex model sizes with no negative impact on accuracy levels. Resource-restricted devices such as mobile devices and embedded systems benefit most from optimization techniques because optimizing performance takes priority on such devices.

Researchers adopted VGG19 for detecting defects on strip steel surfaces through successive solutions of restricted datasets, according to study [15]. The team obtained 97.8% accomplishment in accuracy through the implementation of new preprocessing approaches and model enhancement strategies. Until it conducted another training process, the detection model failed to detect recently introduced defect types and missed apparent defects in its evaluation. The authors conducted a comprehensive research study of diverse pruning and quantization methods in [16]. According to the research findings, model pruning results in an accurate yet significantly smaller model, approximately 13 times smaller compared to the original version, while quantization offers efficiency benefits, although this comes at a cost to precision levels [17].

The fusion filter and channel pruning approach from reference number [18] eliminated 83.8% of model parameters and 71% of computations but obtained slightly better accuracy results. The reinforcement learning-based pruning techniques [19] use automatic management for pruning speeds to achieve 73% maximum computation reduction before accuracy starts to degrade beyond traditional methods of kernel pruning along with guided pruning and soft filter pruning. Scientists across the board have conducted substantial research that led to the development of various quantization techniques. Many scientists have extensively researched and developed various techniques for quantization. The study in [20] analyzed VGG19 along with MobileNet when their parameters were quantized to 8 bits. VGG19 demonstrated higher quantization resilience since it maintained only an 8.58% decline in accuracy, whereas MobileNet experienced substantial accuracy degradation. Research indicates that combined quantization and pruning methods create effective solutions for the objective. Using a 4-bit quantization on the models, it has shown more than 30% variable reduction and as much as 50% memory reduction in CNN models, with minimal damage to accuracy on real-world edge devices [21].

The improvement of VGG19 and ResNet34 model accuracy and complexity comes from Minimax Concave Penalty (MCP)-based pruning techniques [22]. The technique adopted DeepPruningES by [23] along with its evolutionary strategy-based method to achieve up to 80.89% FLOPs reduction on ResNet56 through the generation of multiple model variants with various trade-off selections. A technique that merges weight and feature-level pruning with quantization methodology benefits MobileNetV2 by sustaining a 0.37% accuracy decrease after pruning 50% of the weights while improving operational performance [24].

VGG19 demonstrates its deployment capabilities in real-world applications apart from optimization purposes, thus showcasing its usable nature and versatility. Several improved forms of VGG19 demonstrate high performance levels when used for medical diagnosis applications. The proposed E-VGG19 system by Kandhro et al. detected skin cancer in real-time through pooling/dense layers and the combination of CNN features and SVM and KNN classifiers [25]. The collaboration between Faghihi and colleagues led to 98.18% skin lesion classification accuracy when using VGG19/VGG16 together with feature-modified AlexNet through transfer learning and dropout [26]. The combination of VGG19 along with morphological features and Otsu thresholding and watershed segmentation resulted in more than a 99% success rate for parasitic organism detection per Kumar et al. [27]. Tuned VGG19 models showed better performance than pipeline workflow combinations of previous architectural models for the detection of COVID-19 and pneumonia on X-ray dataset analysis [28], [1]. Research findings proved VGG19's superiority in medical imaging applications over newer structures, DenseNet and ResNet [29]. Enhancing the performance on imbalanced COVID-19 datasets greatly relied upon data augmentation along with hyperparameter optimization techniques [30].

The study by Ahmad et al. demonstrated how VGG19 successfully detected tomato leaf diseases by fine-tuning its last layers when trained on combined field and laboratory samples [31]. Remote sensing technology reached peak performance in small aerial target detection under class imbalance through the hybrid VGG19 and ResNet50 architecture combined with the Spiral Search Grasshopper Algorithm [32]. Accounting for road infrastructure monitoring involved the optimization of VGG19 through Q-learning applied on a two-layer optimizer that managed redundancy in filters and reached a 96.4% F1-score in drone-based road damage detection [33]. The VGG19 network underwent minor modifications to create a face anti-spoofing system, which achieved a 100% success rate in detecting spoofed attacks [34]. The Table Extraction Model (TEM) developed by Iqbal et al. used VGG19 as its base structure to exceed heuristic scanning methods in extracting document tables [35]. The VGG19 model functions as part of content-based image retrieval (CBIR) systems for image retrieval purposes. By adding texture and color features to its design, the model delivered better accuracy for images with semantic similarities [36].

Despite its dated design, VGG19 remains a flexible and effective baseline for optimization research. The model's basic design makes it an optimal choice for optimization research through pruning and quantization and transfer learning and hybrid approaches. The analyzed studies possess various areas that still need improvement. However, some of the biggest VGG19 struggles are in the sense that it has not worked on small datasets or subsets of datasets as appropriate, imbalances, compromises with precision and compression, and not being able to generalize outside of the domain without transfer learning, and there do not exist efficient compression frameworks.

Constructing accurate and highly deployable VGG19 versions requires the resolution of particular challenges to service data-scarce and resource-limited systems.

III. Methodology

The research applied pruning and quantization processes to VGG19 architecture when assessing three benchmark datasets consisting of MNIST, CIFAR-10, and Oxford-IIIT Pet. The datasets contain visual data categories with original dimensions of 28×28 and 32×32 pixels, thus providing efficient computation while supporting complex image identification. Data augmentation procedures and preprocessing methods allowed all images to be uniformly resized to 224×224 pixels, even though VGG19 maintained its fixed input requirements. Among the available datasets, the MNIST provides 70,000 grayscale digits of handwriting distributed across 10 classes, and CIFAR-10 shows 60,000 color object pictures divided into 10 categories. A total of 7,349 images derived from the Oxford-IIIT Pet dataset with 2 classes were utilized, but training and evaluation occurred using 5,000 strategically chosen illustration samples from each collection. The data was split into training and experimental samples, where the data partitioned itself into training and experimental parts, keeping an 8:2 ratio, which ensured model training with diverse datasets while maintaining testing reliability on new data samples. The decision to split the data 80:20 was made by evaluating how training robustness relates to testing strength.

The torchvision library in PyTorch provides comprehensive image processing capabilities through data loading and preprocessing features that contain built-in functionality to control datasets alongside transform functionalities and augmentation options.

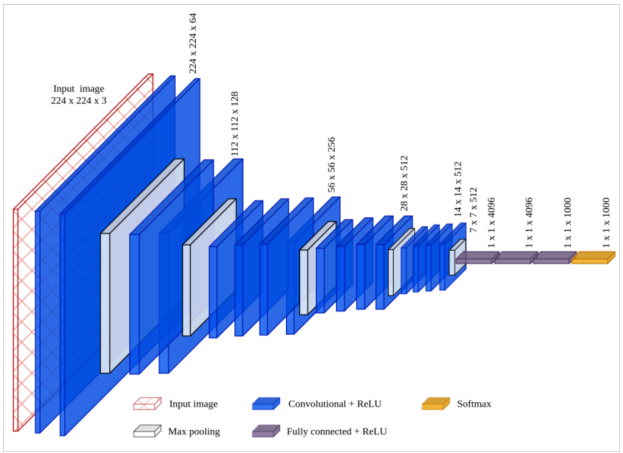

Fig. 1. VGG19 Architecture.

The VGG19 convolutional neural network developed by the Oxford Visual Geometry Group introduces a total of 19 layers composed of 16 convolutional layers together with 3 fully connected layers and 5 max-pooling layers (see Fig. 1) [37].

Through its 3x3 kernel with 1 stride, the model can master small-scale spatial features within each layer. After each convolutional layer, the model passes through ReLU non-linearity for introducing complexity. The max-pooling layers progressively shrink the spatial dimensions but keep the most important of the feature representations. To begin the multi-class classification process, the last fully connected (FC) layer will include the final softmax activation function, and binary classification will be applied to the Oxford-IIIT Pet dataset.

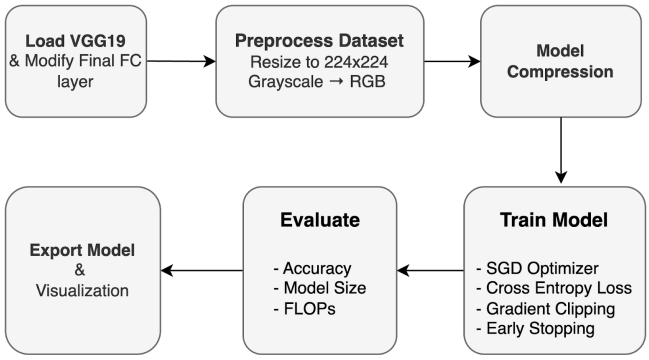

Fig. 2. VGG19 Model optimization and evaluation pipeline.

Fig. 2 represents the operational sequence of modifying, compressing, training, and evaluating the VGG19 model. Testing and training using optimization methods serve as intermediate steps before exporting the model using compression methods.

A. Pruning

The model simplification technique of pruning functions by removing insignificant weights and layers or neurons to maintain performance quality. This process substantially decreases both memory usage and computational expenses for the model. Deploying models on low-end devices makes for efficient performance and function use.

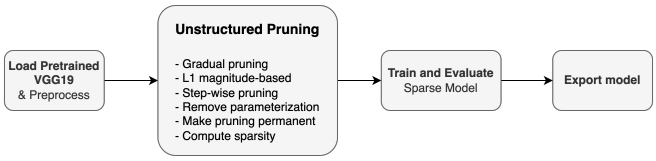

Fig. 3. Unstructured pruning training.

1) Unstructured pruning: In unstructured pruning, the importance of the weights is analyzed, and less important weights are removed. This is usually determined by a gradient-based method during backpropagation. Although this technique significantly reduces the size of the model, the sparse weight matrix has a limited impact on improving the efficiency of standard hardware implementations.

The removal of unimportant weights through unstructured pruning produces reduced models that preserve most of their operational accuracy. Fine-tuning post-pruning is used to recover any possible accuracy loss.

The methodology for unstructured pruning implementation on a pre-trained VGG19 model is illustrated in Fig. 3. L1 magnitude-based pruning uses step-wise methods to remove parameterization before enforcing permanent sparsity, followed by training and evaluation to export the model.

a) Mathematical representation: L1-norm Based Unstructured Pruning: For each weight tensor w in a module, the importance of each weight is computed as the L1-norm, as shown in Eq. (1).

$$\text{importance} = |W| \tag{1}$$

b) Gradual pruning strategy: In the implementation of this strategy, the pruning is applied in several iterations and does not remove a fixed percentage of the weights in one pass, as shown in Eq. (2). The pruning amount is gradually increased over a series of steps, starting from an initial amount and progressing to a target amount, as shown in Eq. (3).

$$\text{Step Increment: } A_{\text{steps}} = \frac{A_{\text{final}} - A_{\text{init}}}{S} \tag{2}$$

$$\text{Pruning Amount Update: } A_i = A_{\text{init}} + i \times A_{\text{steps}} \tag{3}$$

c) Sparsity calculation: After applying pruning, the sparsity of each module is calculated as Eq. (4) as the percentage of weights that have been set to zero.

$$\text{sparsity} = \left(\frac{\text{number of zero weights}}{\text{total number of weights}} \right) \times 100 \tag{4}$$

d) Overall pruning summary: Individual layers are then pruned, a summary is calculated overall, and parameters are aggregated across all layers, as given by Eq. (5).

$$\text{overall\_sparsity} = \left(\frac{T - R}{T} \right) \times 100 \tag{5}$$

where, T = total parameters and R = remaining parameters

2) Structured pruning: Structured pruning functions differently from unstructured pruning because it removes entire neurons, channels, or filters, which results in hardware-friendly and deployment-ready implementation. The implementation of this technique becomes more feasible in practical applications because hardware compatibility surpasses unstructured pruning methods. The method shows outstanding success in minimizing accuracy losses while reducing the convolutional layers of VGG19.

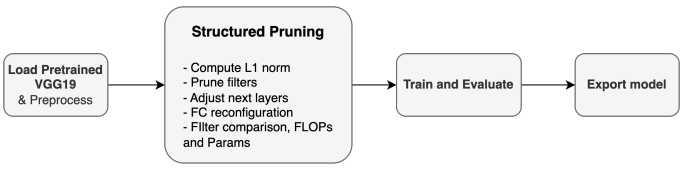

Fig. 4. Structured pruning training.

Fig 4 illustrates structured pruning on VGG19 pretraining. The system first calculates L1 norm values to locate insignificant filter elements for pruning while it readjusts subsequent layers, reorganizes the fully connected layers, measures FLOPs and parameter reduction, and then retrains and generates an exported model.

Structured pruning creates compatibility with hardware accelerators because it allows stable memory patterns that enhance inference speed significantly. The process cuts down the model size with computational complexity because it omits unnecessary channels and neurons to maintain both model performance levels and feature extraction capacity.

3) Mathematical formulation: Filter Importance Computation: For each convolutional filter, its importance is measured by the L1-norm of its weights, Eq. (6).

$$I_i = \sum_j |w_{ij}| \tag{6}$$

a) Pruning decision: For a convolutional layer with n_{filters} filters and a pruning ratio r, the number of filters to prune is given by Eq. (7).

$$n_{\text{prune}} = \lfloor r \times n_{\text{filters}} \rfloor \tag{7}$$

Filters are ranked by their L1-norm importance $I_i = \sum_j |w_{ij}|$, and the n_{prune} filters with the smallest I_i are removed.

b) Adjusting subsequent layers: The input channels for the next convolutional layer are modified according to the output channels that were previously trimmed, according to Eq. (8). This is done by re-indexing the weight tensor along the channel dimension:

$$\text{new input channels} = \text{previously kept indices} \tag{8}$$

B. Quantization

Quantization is a model compression technique that takes high-precision 32-bit floating-point weights and activations to low bit-value such as 8-bit or 4-bit integer. This heavily cuts down the memory usage and computational cost at the price of a little loss of accuracy in edge devices. So it is possible to guarantee good performance along with model functionality.

1) Precision-based quantization: Precision-based quantization, especially fixed-point quantization, is widely used to reduce the bit-width of weights and activations. When applied to the convolutional, activation, and fully connected layers of VGG19 [38]. The model implements 8-bit and 4-bit quantizations for all its convolutional layers in addition to ReLU activations and fully connected layers.

2) Quantization-aware training: QAT is a training procedure in which the quantization operation is applied during the forward operation of the model, i.e., the low-bit representation used for inference[39]. But the full precision of the parameters is saved in the backward pass, that is, the back-propagation, so that the model can learn and correct the quantization errors. This allows the model to be aware of quantization effects during the training phase, and it retains high performance and efficiency at the inference period. Mathematically, the quantized forward pass is represented as Eq. (9).

$$x_{\text{quant}} = q(x) \tag{9}$$

$q(x)$ is the quantization function. This way, inference-time quantization's accuracy is preserved, and QAT is one of the best methods for compressing deep learning models.

3) Module fusion: Efficient quantization, together with faster inference speed, needs module fusion to serve as a critical preparation step. The model gains advantages from reduced overhead, lower quantization error, and improved runtime efficiency through single fused modules retaining the same computed output resulting from combining convolution and ReLU layers.

4) Quantization and dequantization: These equations describe the dictionary defining the mapping of floating-point numbers to their integer representation: The quantized value was calculated with Eq. (10), and the dequantized float value was calculated with Eq. (11).

$$\text{Quantization: } q = \text{round}\left(\frac{x}{s}\right) + z \tag{10}$$

$$\text{Dequantization: } x = s \times (q - z) \tag{11}$$

5) 8-Bit QAT: The forward pass of quantized VGG19 models is composed of a sequence of quantized convolutions and fully connected layers, with in-between ReLUs and average pooling.

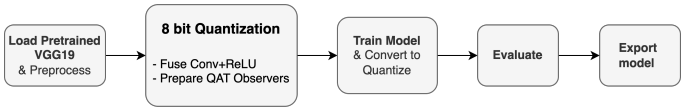

Fig. 5. 8-bit Quantization aware training.

Fig. 5 shows the pipeline for the quantization of 8-bit VGG19 models during preprocessing. The quantification method starts with convolutions and ReLUs before preparing QAT observers, then conducts training followed by quantized model conversion and evaluation and final export processing.

6) Kaiming normal initialization for weights: This method makes the weights have the appropriate value such that the variance of the activations remains roughly the same across layers. It is also good at signalling in deep nets using ReLU activations, helping to deal with issues with the vanishing or exploding gradients. The weights w are sampled from a normal distribution defined as Eq. (12).

$$w \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right) \tag{12}$$

7) 4-Bit QAT: A structure similar to the 8-bit configuration was followed, but all quantization functions and weight representations were replaced with their 4-bit counterparts.

The workflow for 4-bit QAT to apply on the VGG19 model appears in Fig. 6. The procedure starts with importing the model while performing the required input preparation.

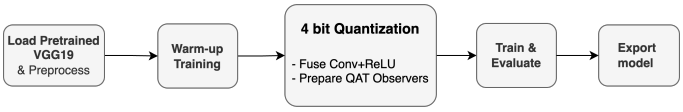

Fig. 6. 4-bit Quantization Aware Training

After weight stabilization, the short warm-up training begins. The implementation of QAT begins when 4-bit quantization combines convolution and ReLU layers and then adds QAT observers for all the layers. The model proceeds to fine-tuning through a training and evaluation stage, after which it becomes exportable for deployment.

The choice of pruning rates and quantization bits was done under the premise of preliminary testing that optimized the trade-offs. It was found that a pruning rate of 30% yields the best trade-off between compressing and maintaining the accuracy gained from the preliminary experiments. Training the hyperparameters was performed based on known model compression methods. In the process, it was found that cosine annealing learning rate is an absolute requirement for achieving stable QAT convergence.

8) Cosine annealing learning rate equation: Eq. (13) gives the dynamical learning rate during the period of training epochs by implementing a cosine that provides a progressive decay within the process of learning.

$$n_t = n_{\min} + \frac{1}{2}(n_{\max} - n_{\min})\left(1 + \cos\left(\frac{T_{\text{cur}}}{T_{\max}} \times \pi\right)\right) \tag{13}$$

C. Combined Metrics for Performance Evaluation

1) Model size: Eq. (14) was used in calculating the model size.

$$\text{Model Size (MB)} = \sum_{p \in \text{parameters}} \frac{\text{numel}(p) \times \text{element\_size}(p)}{1024^2} \tag{14}$$

2) Inference Time (T): Eq.(15) defines the inference time.

$$T_{\text{batch,avg}} = \frac{\sum_{i=1}^{N} T_{\text{batch}}^{(i)}}{N} \tag{15}$$

3) Parameter Reduction (R): The percentage decrease in model size due to compression [see Eq. (16)]:

$$R = \left(1 - \frac{S_{\text{compressed}}}{S_{\text{original}}}\right) \times 100 \tag{16}$$

4) Incremental memory usage: Eq. (17) defines incremental memory usage.

$$\Delta M = M_{\text{current}} - M_{\text{baseline}} \tag{17}$$

5) FLOPs counting: Eq. (18) defines FLOPs counting.

$$\text{FLOPs} = 2 \times \sum_{p \in \text{parameters}} \prod_{i=1}^{\text{ndim}(p)} \dim(p) \tag{18}$$

6) Early stopping: The early stopping mechanism continuously monitors the validation loss $L_{\text{val},t}$. If, for a given epoch t, the loss difference

$$L_{\text{val},t} - L_{\text{val},t+\Delta t} \tag{19}$$

Eq. (19) is less than min_delta for patience consecutive epochs, the algorithm concludes that further training is unlikely to yield significant improvements and stops the training process.

IV. Experimental Results

The evaluation in this part utilizes experimental tests on VGG19 optimizations to study structured and unstructured pruning together with quantization effects on performance outcomes. Three popular benchmark datasets—MNIST, CIFAR-10, and Oxford-IIIT Pet Dataset—are used for the evaluation to measure the actual performance of these optimizations. The tests are mainly conducted based on several important indicators, including model size, FLOPs, classification accuracy, inference latency, CPU utilization, and memory consumption.

In this study, the training and evaluations are run on Kaggle P100 GPUs to maximize computational efficiency. After the training phase, the developed models were exported and subsequently subjected to evaluation. The experimental benchmarking of the models was conducted using CPUs on Google Colab. The 4-bit model inference overhead was specifically measured on Google Colab CPUs during this evaluation phase. Although 4-bit quantization significantly reduces model size and memory consumption, the overhead arises because standard CPUs are optimized for 8-bit or higher precision operations. In the absence of native 4-bit hardware support, additional computational cost is incurred due to packing, unpacking, and emulating low-bit arithmetic during inference.

The optimization process of VGG19 baseline models starts through standard training interventions on each dataset. The model then undergoes both structured and unstructured pruning procedures, which enhance resource utilization. The efficiency of parameter utilization in the model has improved since unnecessary parameters are now being removed. Tests of both 4-bit and 8-bit quantization techniques seek to enhance the model compression efficiency.

The collected findings help to determine the impact of these optimization methods on system performance. Model efficiency faces an opposing force against inferential performance when implementing these optimization techniques. The findings from the research demonstrate valuable information regarding the realistic implementation prospects of the optimized VGG19 model.

1) Model efficiency summary: Examination of the advantages of pruning and quantization on the VGG19 architecture was conducted via inspection of performance metrics, including decreased parameters, increased FLOPs, and overall sparse network. Eq. (14) was used in calculating the model sizes of each configuration. The performance metrics of the baseline model and its structured and unstructured pruning implementations appear in Table I, Eq. (16), and Eq. (17).

TABLE I. Model Efficiency Comparison

Metric	Original	Structured	Unstructured	8-bit	4-bit
Parameters (M)	139.58	98.70	80.27	139.58	139.58
Pruning Rate (%)	–	30	30	–	–
FLOPs (G)	0.28	9.77	0.28	0.28	0.28
Sparsity (%)	0	0	42.50	1.59	48.10
Model Size (MB)	532.57	376.64	532.57	133.42	133.20
Size Reduction (%)	–	29.28	0	74.95	74.99

The VGG19 system requires 139.58 million model parameters and computation of 0.28 GFLOPs Eq. (18). A structured pruning technique with a 30% rate allows our model to maintain 98.70 million parameters while lowering FLOPs to 9.77 GFLOPs. In contrast, the unstructured pruning presents a higher pruning ratio of 42.50%; thus, the total parameter in the compressed model is reduced to 80.27 million. Overall sparsification of the pruned network is 42.50%, greatly reducing computational complexity. For 8-bit QAT and 4-bit QAT, the observed model size reduction is almost 75% [38].

For training efficiency, the pruned model was trained over 30-45 epochs with early stopping with a batch size of 32. The optimization method decreases algorithm processing complexity without affecting accuracy performance[40].

2) Best weight distribution after model quantization: These experiments show that this approach effectively can compress the model, reduce the number of FLOPs, and keep a balance between efficiency and performance, potentially demonstrating the most effective technique to compress deep neural networks with a slight loss in accuracy.

3) Best layer comparison after model compression: In summary, the results from Fig. 14 and Fig. 13 show that, although it offers compression benefits, structured pruning keeps accuracy. Excessive quantization methods, like using 4-bit, need exact calibration precision when working with challenging dataset materials.

4) Inference performance and norm-based metrics: Table III shows the effect of the comparison depending on inference time [Eq. (15)], CPU consumption, memory consumption, norm-based weight measures (L1, L2), and sparsity.

The best way for real-time inference with a minimal loss of model goodness was 8-bit quantization, which reliably delivered the fastest inference time and less memory usage across all datasets. 4-bit quantization provided the highest sparsity (up to 48%) and considerable model compression along with competitive inference speed with overhead.

The structured pruning method decreased inference time and memory consumption, though it did not introduce weight sparsity because it primarily modified network architecture instead of parameter numbers.

TABLE II. CLASSIFICATION PERFORMANCE

Dataset	Approach	Epochs	Accuracy	Top-1 Acc.	Top-5 Acc.
MNIST	Baseline Model	13	98.00%	98.00%	100.00%
	Structured Pruning	30	97.60%	97.60%	100.00%
	Unstructured Pruning	16	98.10%	98.10%	100.00%
	8-bit Quantization	17	97.50%	97.50%	100.00%
	4-bit Quantization	9	98.30%	98.30%	100.00%
CIFAR-10	Baseline Model	11	86.30%	86.30%	99.80%
	Structured Pruning	30	71.80%	71.80%	98.10%
	Unstructured Pruning	19	86.50%	86.50%	99.70%
	8-bit Quantization	21	85.70%	85.70%	99.60%
	4-bit Quantization	11	71.80%	71.80%	84.80%
Oxford-IIIT Pet	Baseline Model	9	96.74%	96.74%	–
	Structured Pruning	18	94.02%	94.02%	–
	Unstructured Pruning	15	97.96%	97.96%	–
	8-bit Quantization	13	97.42%	97.42%	–
	4-bit Quantization	10	89.67%	89.67%	–

Note: Top-5 accuracy was not computed for the Oxford-IIIT Pet dataset due to its binary classification nature.

TABLE III. INFERENCE PERFORMANCE AND NORM-BASED METRICS

Dataset	Approach	Inference Time (ms)	CPU Usage (%)	Peak Memory (MB)	L1 Norm	L2 Norm	Sparsity
MNIST	Baseline Model	849.286	98.530	41.762	772058	100.57	0%
	Structured Pruning	483.952	99.700	35.641	536877	83.23	0%
	Unstructured Pruning	829.155	99.547	24.812	665095	98.75	7.93%
	8-bit Quantization	472.216	99.827	0	771630	100.09	1.59%
	4-bit Quantization	489.674	99.352	0	622668	96.62	48.10%
CIFAR-10	Baseline Model	842.537	99.675	30.719	772053	100.57	0%
	Structured Pruning	473.458	99.819	23.977	536882	83.23	0%
	Unstructured Pruning	837.766	99.562	30.699	666228	98.75	0.12%
	8-bit Quantization	480.096	99.626	0	771622	100.09	1.59%
	4-bit Quantization	485.351	99.577	0	616684	95.94	48.42%
Oxford-IIIT Pet	Baseline Model	837.163	99.610	30.594	771776	100.55	0%
	Structured Pruning	475.043	99.717	36.094	536555	83.20	0%
	Unstructured Pruning	836.408	99.533	36.816	664427	98.67	0.66%
	8-bit Quantization	471.534	99.866	0.258	771024	100.01	1.59%
	4-bit Quantization	488.039	99.521	0	648955	100.61	48.11%

The results from unstructured pruning methods were inconsistent because memory consumption improved in certain use cases, yet inference speed boosts, along with sparsity degrees remained dependent on the dataset and generally low. Considering smaller and less redundant models, all optimization methods lowered L1 and L2 norms from the baseline.

V. DISCUSSION

This research demonstrates that both pruning and quantization methods are effective in improving the VGG19 model efficiency for deep learning functions. The experimental figures make visible the different trade-offs and benefits produced by each optimization method implemented in this study.

Fig. 7 demonstrates. Both quantization techniques reduced the model size extensively, although they left the actual parameter count unchanged because quantization modifies computational bit representation instead of parameter number. Structured pruning maintained a higher parameter count at

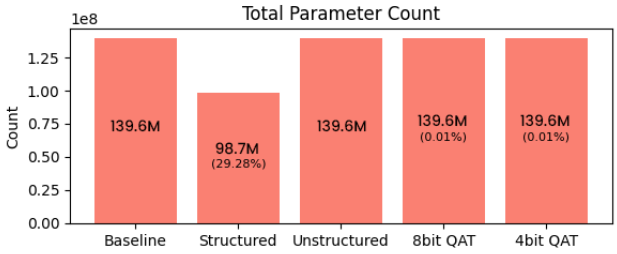

Fig. 7. Total parameter count of VGG19 variants.

98.7 million, but the parameter total in unstructured pruning dropped to 80.27 million.

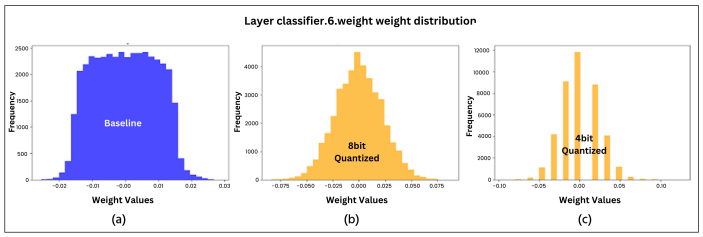

Fig. 8. Best QAT weight distribution.

The effectiveness of QAT was depicted in Fig. 8 by its weight distribution. The baseline distribution is plotted as shown in Fig. 8(a), and the 8-bit quantization, which is plotted in Fig. 8(b), still has a relatively smooth distribution. Fig. 8(c), however, shows that the use of 4-bit quantization leads to the doubling of weights around the zero value, which means that the model weights itself to the larger quantization error and reduced precision.

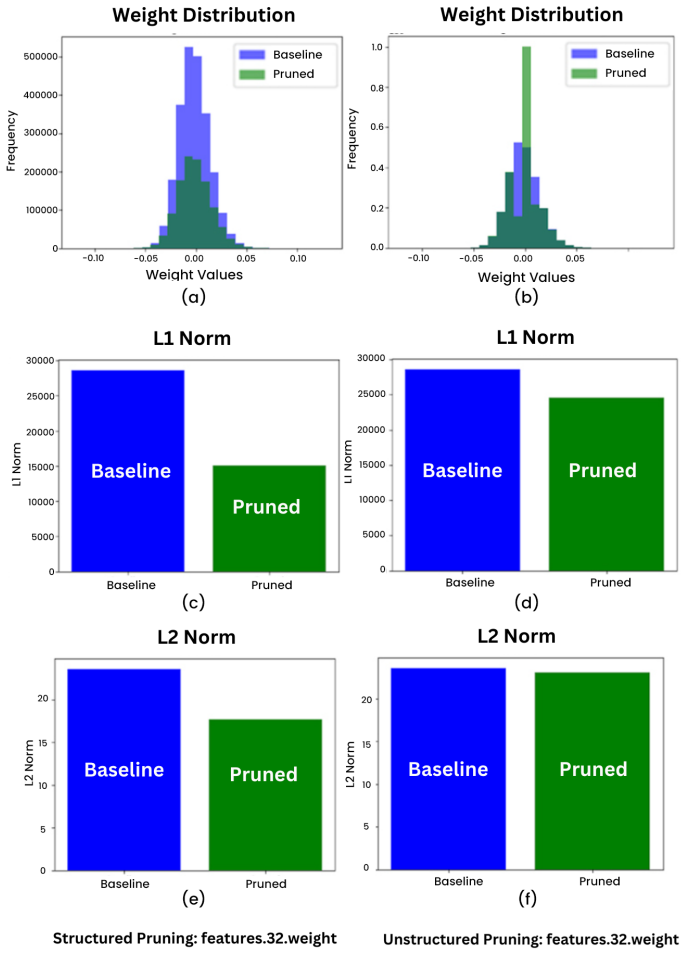

Fig. 9. Best pruned layer compression.

Fig. 9 displays particular impacts of pruning, since this model compression comparison acts at the level of a layer. Structured pruning [Fig. 9(a),Fig. 9(c),Fig. 9(e)] preserved the functionality of convolutional layers by removing the most redundant parts of deeper fully connected layers. In particular, Fig. 9(a) is the distribution of weight before and after pruning, Fig. 9(e) and Fig. 9(c) show the decreasing L2 and L1 norms, which means sparsity of parameters. The effect of unstructured pruning is shown in Fig. 9(b),Fig. 9(d) and Fig. 9(f), with the more focused post-pruning weight distribution shown in Fig. 9(b) and the L2 and L1 norm comparison seen in Fig. 9(d) and Fig. 9(e), respectively. The pruning was used to maintain the feature extraction capability of the model and to trim its structure.

As it is presented in Table II, five different architectures of VGG19 were assessed on three different datasets: MNIST, CIFAR-10, and Oxford-IIIT Pet Dataset. The models will be denoted as Baseline, Structured Pruning, Unstructured Pruning, 8-bit Quantization, and 4-bit Quantization. All tested models in the MNIST evaluation exceeded 97.5% accuracy levels. With a 4-bit quantized model, the accuracy level reached 98.30% while exceeding baseline standards at 98.00%. The 9-epoch training duration marked the minimum needed for this model. Unstructured pruning added slightly to the baseline with the accuracy of 98.10%, indicating sparsity-based pruning also works for lesser complexity of datasets. Despite being marginally reduced (97.60% and 97.50% in structured trimming and 8-bit quantization, respectively), they maintained complete top 5 accuracy. Using unstructured pruning proved to be a slightly better option when it was applied to CIFAR-10 as multi-class (86.50% vs. 86.30%). This was because it could keep the accuracy high for the more complicated dataset. 8-bit quantization hit a good accuracy (85.70%) with just small losses. The performance of the models deteriorates extensively to 71.80% when applying structured pruning and 4-bit quantization techniques to the dataset. The Oxford-IIIT Pet dataset by binary classification proved the substantial performance improvement brought up through unstructured pruning, achieving 97.96%, better than the basic 96.74%. Both 8-bit quantization and structured pruning resulted in comparable results to the baseline, with performances of 97.42% and 94.02%, respectively. However, 4-bit quantization decreased the most.

1) Results: Training figures using MNIST, Oxford-IIIT Pet, and CIFAR-10 datasets appear in Fig. 10(a), Fig. 11(a), Fig. 12(a) to show training behavior and validation and corresponding training and validation accuracies are presented in Fig. 10(b), Fig. 11(b), Fig. 12(b) to show training and validation accuracies. The training curves show that unstructured pruning techniques, together with other pruning methods, brought stability to training operations while achieving accurate test results throughout all epochs. The results in Fig. 12 show that structured pruning and 4-bit QAT [41] struggle on the complex CIFAR-10 dataset because they produce slow convergence and an enlarged generalization gap during aggressive compression efforts.

Devices that operate at edge points need maximum performance throughout inference procedures to qualify for deployment. The optimization algorithms in Fig. 14 reveal lower norm values than basic model protocols for L1 and L2 norm metrics. Their functionality depends on parameter sets, which must be minimal and simple. The structured pruning approach

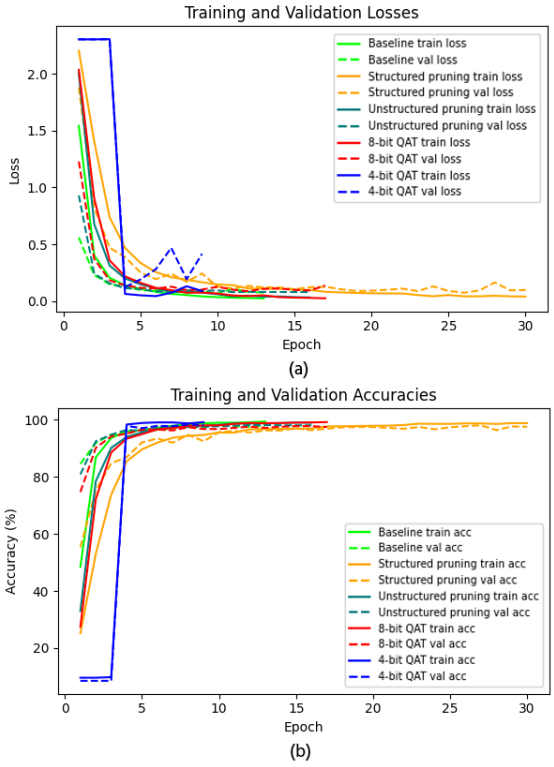

Fig. 10. MNIST training and validation.

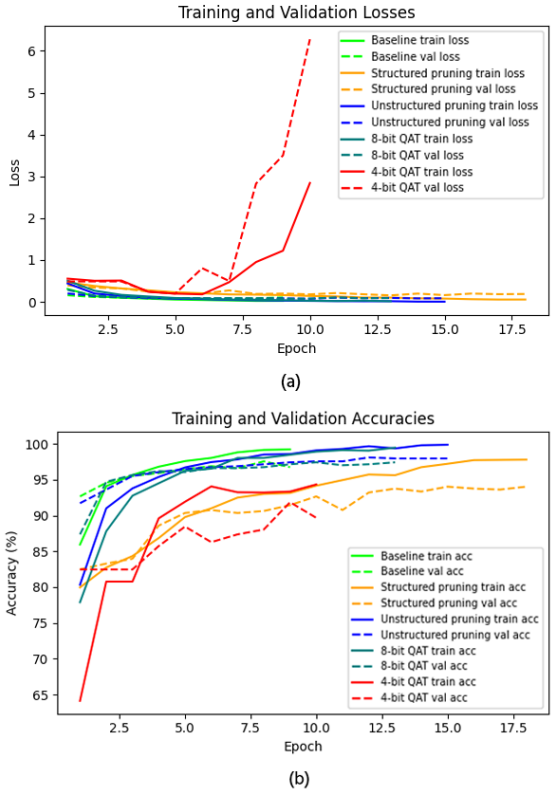

Fig. 11. The Oxford-IIIT pet training and validation.

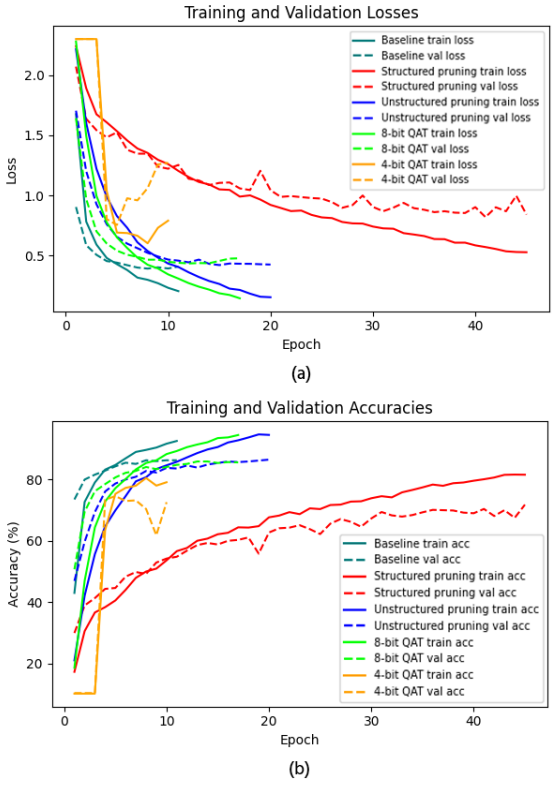

Fig. 12. CIFAR-10 training and validation.

effectively reduces both norms while preserving parameter density because network reduction outperforms weight removal in performance improvement.

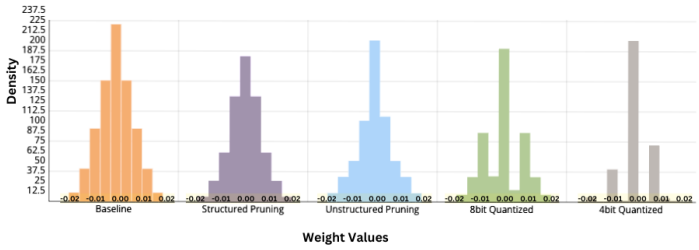

Fig. 13. Average weight distribution of VGG19 variants.

Fig. 13 shows the typical weight distribution statistics across all variants, which validates this observation. The precise and uniform weight distribution that occurs when pruning methods are structured leads to better hardware efficiency and more foreseeable inference behavior. The weight bins of the 4-bit quantized model became discrete from heavy quantization because of this pattern, which allowed it to achieve sparsity levels exceeding 48%, according to performance data.

The numerical data found in Table III proves the findings presented previously. The deployment times for 8-bit quantization kept its approximate duration at ≈472 milliseconds while simultaneously demonstrating zero memory consumption and minimal deviations in L1 and L2 norms when compared to the baseline model. 4-bit quantization outpaced 8-bit QAT

regarding inferential speed, even though it achieved peak sparsity levels together with minimum memory usage. This makes 8-bit QAT the most effective option for achieving performance-quality equilibrium.

Pruning methods that do not follow structured guidelines prove to be the most suitable techniques for preserving exact accuracy on complex datasets. The sparse retraining technique achieved superior results to baseline measurements across the Oxford-IIIT Pet data through its 97.96% accuracy, which surpassed the 96.74% baseline score, thus indicating that the method enhances generalization effectiveness. The model limitation was reduced through data augmentation and early stopping techniques, although no regularization methods such as dropout or L2 penalties were implemented. A minor degree of model bias may result from this implementation when applying it to complex datasets.

The use of four bits during quantization resulted in substantial accuracy deterioration when implementing it for CIFAR-10 and Oxford-IIIT datasets. Researchers need to develop complex quantization methods that choose the best quantization parameters when sensitivity levels change between different network parts.

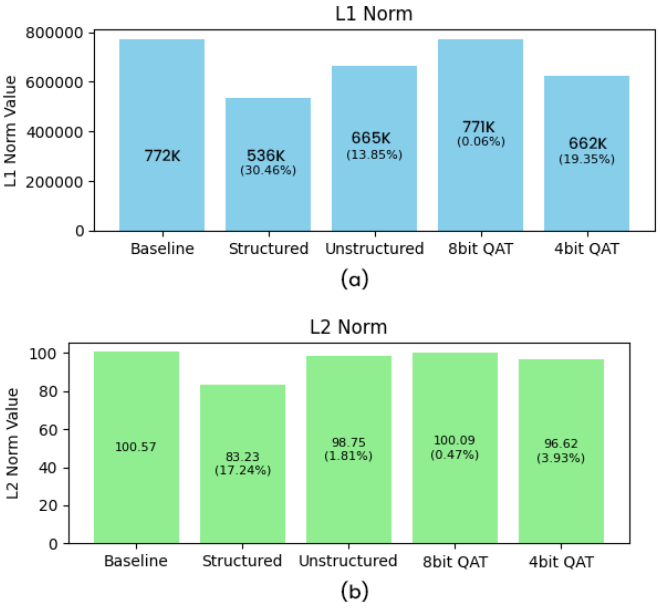

Fig. 14. L1, L2 norm of VGG19 variants.

The computational efficiency of quantized models is undeniable, according to Fig. 14 and Table III, thus making them optimal picks for mobile devices that require energy-efficient approaches, research shows.

A. Parameter Influence Analysis

According to our findings in the three datasets:

1) Pruning rate (30%): The one chosen when unsuccessful experimental tests indicated optimum accuracy-compression balance. Both the larger rates (40%) resulted in a significant loss in accuracy on the CIFAR-10, and the lower ones (20%) did not provide enough beneficial compression.

2) Quantization precision: Inference speed remained high on 8-bit with the same baseline accuracy as the more digitally prioritized architectures. The 4-bit had the best compression (75%), but it experienced brittle accuracy loss (10 to 15%) and overhead, as non-native compute code had to run on the T4 GPU.

3) Learning rate schedule: Cosine annealing was crucial to quantized models, and a fixed learning rate was not working well during QAT fine-tuning runs.

4) Batch size (32): Trade-off memory limitations and stability of training under all compression methods and data sets.

VI. Future Work

Future investigations should explore mixed-precision quantization methods, a scenario in which various layers of the network are quantized (i.e., encoded to reduce precision) at different bit depths in order to achieve the best out of precision and out of the computational budget. The search for hardware acceleration optimization methods, like Tensor Processing Units (TPUs) or Application-Specific Integrated Circuits (ASICs), might enable the adoption of quantum models, especially in resource-constrained environments. An additional and more promising region of research is the development of adaptive rephrase precision in real-time in accordance with the behavior of the input or application-defined constraints, thus providing robust and flexible models.

Moreover, combining structured pruning with QAT is an attractive research area that can achieve the highest computational efficiency with little loss of model accuracy. Benchmarking those optimization techniques against huge sets of various types of datasets, including realistic large-scale ones, would bring valuable knowledge to their generic acceptability. Finally, energy usage trend analysis among available quantized models in the deployment environment will be done to identify which models are energy efficient. This would be a major step in making greener artificial intelligence (AI) solutions without sacrificing AI performance and with less environmental impact of deep learning models.

VII. Conclusion

In this study, we optimized the model VGG19 by model compression techniques, structured pruning, unstructured pruning, and QAT 8-bit and 4-bit precision. We aimed at speeding up deep convolutional networks through limited resource platforms without compromising the accuracy of classification.

Experimental tests performed on the three benchmarks MNIST, CIFAR-10, and Oxford-IIIT Pet have shown that unstructured pruning and 8-bit QAT provided the best tradeoff between model sizes and predictive accuracy. The inference time and the model size were well-traded, and the same produced an acceptable accuracy by structured pruning. At the same time, the 4-bit QAT achieved the best compression rate (approximately 75%) and sparsity (approximately 48%), but degradation in accuracy occurred on more complex datasets, such as CIFAR-10 and Oxford-IIIT Pet.

These findings indicate that structured connectivity and lower-bit quantization-based pruning have the potential to greatly improve deploying VGG19 to edge devices and that

they can be further improved with the help of training strategies such as early stopping and cosine annealing. Nonetheless, the active quantization, such as 4-bit QAT, has to be tuned a little further to eliminate the inaccuracy, particularly with complex visual information.

Future work will look at the hybrid approaches to the compression of neural networks, especially at the intersection of mixed-precision quantization and structured pruning, towards pursuing an ideal balance between performance and intellectual accuracy of the inference. In addition, evaluating energy usage and optimization in comparison to edge-concentrated hardware acceleration will also be ideal in making these techniques practical in real-world low-power applications.

REFERENCES

[1] J. Ma, G. Wang, L. Zhang, and Q. Zhang, "Restoration and enhancement on low exposure raw images by joint demosaicing and denoising," *Neural Networks*, vol. 162, pp. 557–570, 2023.

[2] J. Sun, W. Yao, T. Jiang, D. Wang, and X. Chen, "Differential evolution based dual adversarial camouflage: Fooling human eyes and object detectors," *Neural Networks*, vol. 163, pp. 256–271, 2023.

[3] S. Mascarenhas and M. Agarwal, "A comparison between vgg16, vgg19, and resnet50 architecture frameworks for image classification," in *Proc. Int. Conf. Disrupt. Technol. Multidiscip. Res. Appl. (CENTCON)*, 2021, pp. 96–99.

[4] J. Chen, S.-h. Kao, H. He, W. Zhuo, S. Wen, C.-H. Lee, and S.-H. G. Chan, "Run, don't walk: Chasing higher flops for faster neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 12 021–12 031.

[5] J.-H. Luo and J. Wu, "Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference," *Pattern Recognition*, vol. 107, p. 107461, 2020.

[6] A. Kuzmin, M. Nagel, M. Van Baalen, A. Behboodi, and T. Blankevoort, "Pruning vs quantization: Which is better?" *Advances in neural information processing systems*, vol. 36, pp. 62 414–62 427, 2023.

[7] X. Qu, D. Aponte, C. Banbury, D. P. Robinson, T. Ding, K. Koishida, I. Zharkov, and T. Chen, "Automatic joint structured pruning and quantization for efficient neural network training and compression," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 15 234–15 244.

[8] Y. Ding and D.-R. Chen, "Optimization based layer-wise pruning threshold method for accelerating convolutional neural networks," *Mathematics*, vol. 11, no. 15, p. 3311, 2023.

[9] Y. He and L. Xiao, "Structured pruning for deep convolutional neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 12–34, 2023.

[10] T. Ganguli and E. K. P. Chong, "Activation-based pruning of neural networks," *Algorithms*, vol. 17, no. 1, p. 48, 2024.

[11] X. Liu, M. Ye, D. Zhou, and Q. Liu, "Post-training quantization with multiple points: Mixed precision without mixed precision," *arXiv preprint arXiv:2002.09049*, 2020. [Online]. Available: https://arxiv.org/abs/2002.09049

[12] X. Wu, E. Hanson, N. Wang, Q. Zheng, X. Yang, H. Yang, S. Li, F. Cheng, P. P. Pande, J. R. Doppa, K. Chakrabarty, and H. Li, "Block-wise mixed-precision quantization: Enabling high efficiency for practical reram-based dnn accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 1, pp. 1–14, 2024.

[13] H. Zhang, L. Wang, and T. Chen, "An improved vgg19 transfer learning strip steel surface defect recognition deep neural network based on few samples and imbalanced datasets," *Applied Sciences*, vol. 11, no. 6, pp. 1–15, 2021.

[14] J. Kim, P. Luo, and F. Wu, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 512, pp. 23–45, 2021.

[15] R. Das, A. Kumar, and S. Mehta, "A transfer learning with structured filter pruning approach for improved breast cancer classification on point-of-care devices," *Computers in Biology and Medicine*, vol. 125, pp. 55–67, 2021.

[16] Y. Huang and B. Singh, "A compact parallel pruning scheme for deep learning model and its mobile instrument deployment," *Mathematics*, vol. 10, no. 12, pp. 1–17, 2022.

[17] F. Wang, J. Li, and T. Xu, "Automatic group-based structured pruning for deep convolutional networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 5, pp. 2345–2356, 2022.

[18] J. Brown and M. White, "Do all mobilenets quantize poorly? gaining insights into the effect of quantization on depthwise separable convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2021, pp. 77–88.

[19] V. Kulkarni, A. Sharma, and P. Dubey, "Methodologies of compressing a stable performance convolutional neural network in image classification," *Neural Processing Letters*, vol. 51, no. 3, pp. 567–580, 2020.

[20] B. Shah and A. Mehra, "Pruning deep neural network models via minimax concave penalty regression," *Applied Sciences*, vol. 14, no. 9, pp. 1–18, 2024.

[21] H.-H. Chin, R.-S. Tsay, and H.-I. Wu, "A high-performance adaptive quantization approach for edge cnn applications," *arXiv preprint arXiv:2107.08382*, 2021.

[22] K. Ghosh, R. Patel, and D. Narang, "Training deep neural networks with joint quantization and pruning of features and weights," in *International Conference on Learning Representations (ICLR)*, 2022, pp. 1–12.

[23] I. A. Kandhro *et al.*, "Performance evaluation of e-vgg19 model: Enhancing real-time skin cancer detection and classification," *Heliyon*, vol. 10, p. e31488, 2024.

[24] I. Ahmad *et al.*, "Optimizing pretrained convolutional neural networks for tomato leaf disease detection," *Complexity*, vol. 2020, p. Article ID 8812019, 2020.

[25] B. K. Nyaupane, "Pneumonia of chest x-ray images detection using vgg architectures," *LEC Journal*, vol. 4, no. 1, pp. 43–48, 2022.

[26] R. Siddiqi and S. Javaid, "Deep learning for pneumonia detection in chest x-ray images: A comprehensive survey," *Journal of Imaging*, vol. 10, no. 8, p. 176, 2024.

[27] M. M. A. Monshi *et al.*, "Covidxraynet: Optimizing data augmentation and cnn hyperparameters for improved covid-19 detection from cxr," *Computers in Biology and Medicine*, vol. 133, p. 104375, 2021.

[28] H. Samma *et al.*, "Evolving pre-trained cnn using two-layers optimizer for road damage detection from drone images," *IEEE Access*, 2021, early access.

[29] S. D. Thepade *et al.*, "Face presentation attack identification optimization with adjusting convolution blocks in vgg networks," *Intelligent Systems with Applications*, vol. 16, p. 200107, 2022.

[30] M. Z. Iqbal, N. Garg, and S. B. Ahmed, "Table extraction with table data using vgg-19 deep learning model," *Sensors*, vol. 25, no. 1, p. 203, 2025.

[31] A. Stateczny, G. U. Kiran, G. Bindu, K. R. Chythanya, and K. A. Swamy, "Spiral search grasshopper features selection with vgg19-resnet50 for remote sensing object detection," *Remote Sensing*, vol. 14, no. 21, p. 5398, 2022.

[32] K. T. Ahmed *et al.*, "Maximum response deep learning using markov, retinal & primitive patch binding with googlenet & vgg-19 for large image retrieval," *IEEE Access*, 2021, early access.

[33] A. Faghihi, M. Fathollahi, and R. Rajabi, "Diagnosis of skin cancer using vgg16 and vgg19 based transfer learning models," arXiv preprint arXiv:2404.01160, 2024.

[34] Y. Kumar *et al.*, "Enhancing parasitic organism detection in microscopy images through deep learning and fine-tuned optimizer," *Scientific Reports*, vol. 14, no. 5753, 2024.

[35] M. Fu, H. Yu, J. Shao, J. Zhou, K. Zhu, and J. Wu, "Quantization without tears," arXiv preprint arXiv:2411.13918, 2024.

[36] F. E. Fernandes and G. G. Yen, "Pruning deep convolutional neural networks architectures with evolution strategy," *Information Sciences*, vol. 541, pp. 345–361, 2020.

[37] A. Bouguettaya and H. Zarzour, "Cnn-based hot-rolled steel strip surface defects classification," *The International Journal of Advanced Manufacturing Technology*, 2024.

[38] X. Li, Y. Chen, and J. Wang, "A mutual learning framework for pruned and quantized networks," *Journal of Computer Science & Technology*, vol. 38, no. 23, 2023.

[39] C. Gernigon, S.-I. Filip, O. Sentieys, C. Coggiola, and M. Bruno, "Adaqat: Adaptive bit-width quantization-aware training," in *2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS)*. IEEE, 2024, pp. 442–446.

[40] Q. Lu, "Tinyml computer vision using coarsely-quantized log-gradient input images," Stanford University, 2023. [Online]. Available: https://purl.stanford.edu/fb372vt6975

[41] P. P. F. Delgado, "Real-time implementation of 3d lidar point cloud semantic segmentation in an fpga," University of Minho, 2022.