International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Pruning Framework for Efficient Facial Emotion Recognition using Deep Learning

P. Vijaya Lakshmi [1], V.Murugesh[2]

Submitted: 27/01/2024 **Revised**: 05/03/2024 **Accepted**: 13/03/2024

Abstract: The application of neural networks in their entirety has produced remarkable outcomes in the domain of facial emotion recognition. The enormous scale, however, renders these models impracticable in the real world. In an effort to address this deficiency, this study introduces an innovative approach that combines two well-known model compression methods—pruning. In order to decrease the dimensions of neural models that are explicitly designed for the purpose of facial emotion recognition, we propose the implementation of a pruning-then-quantization framework. Comprehensive experiments conducted on three separate datasets provide evidence of the framework's capability to significantly compress models without compromising their performance. In order to delve deeper into the nuanced effectiveness and versatility of our innovative framework within fine-grained modules, we execute an exhaustive analysis of the compression performance layer by layer. The accuracy achieved by the pruning process is 97.95%

Keywords: *Facial Emotion Recognition, Neural Network Compression, Model Pruning, Quantization Techniques, Model Compression Framework, Deep Learning Optimization*

1. Introduction

In recent times, deep learning models have demonstrated remarkable progress across various research domains, such as computer vision, natural language processing, and voice recognition, among others. This can be attributed to the robust fitting and learning capabilities that stem from their massive and intricate architectures. Because this area of research gets a lot of attention, facial emotion recognition (FER) is always adding to its cutting-edge findings, which are made with models trained on deep neural networks.[1] Given that FER has been shown to work, it is very likely that these deep neural network architectures will be used in real life, especially in peripheral computing, where they can power smart speakers and assistants.

Nevertheless, numerous endeavors to implement these deep learning network models in practical scenarios are impeded by their incompatibility with the restricted memory and processing capabilities of devices. Because of this, it is very important to compress these deep neural network models with lots of parameters so that they can run on devices with less processing power. Quantization and pruning are two conventional methods utilized to compress models. Models are compressed using the quantization method and the pruning-based strategy, respectively, by eliminating unnecessary weights or reducing the number of bits used to encode the maodel.

Although the pruning-based method may ultimately result in a higher compression ratio, it generally impairs the performance of the model.[2] The quantization technique has the potential to substantially compress the model while maintaining its efficacy, which is equivalent to that of the original model.

This article presents a pruning-then-quantization method as a means to compress the VGG16 models utilized for facial emotion recognition. The disadvantages and advantages of both approaches are duly considered. This methodology enables a substantial compression ratio without compromising an appreciably high degree of precision. Once the L1 norm of each convolutional kernel in each layer has been computed, the N absolute minimum values are eliminated as an integral part of the pruning procedure. Following this, the model and training weights were revised. Following this, during the quantization phase, we endeavour to learn quantization ranges and apply batch normalization folding to this reduced model. The model is subsequently retrained, utilizing the same weights and biases that were employed during the initial training phase. Our suggested strategy achieves a high compression ratio while maintaining comparably high model performance, as demonstrated by experiments on three datasets. Our primary contributions consist of the following:

(1) a framework for facial emotion recognition VGG16 models that integrates pruning and quantization compression techniques into a single application.

(2) Results from experiments on three different datasets show that the framework we've proposed can achieve both a high compression ratio and a high level of accuracy at the same time.

[1,2] *Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur,Andhrapradesh*
ORCID ID : /0009-0003-5846-4123
** Corresponding Author Email: pesaruvijayalakshmi@gmail.com*

In order to assess the effectiveness and adaptability of our system within granular modules, we conducted a performance analysis of layer-wise compression.

2. Literature Review

Numerous studies have investigated the efficacy of quantization and pruning in improving sentiment analysis models. Smith et al. (2018)[3] examined the application of pruning convolutional neural networks (CNNs) in the context of sentiment analysis in social media texts. Utilizing a variety of pruning techniques, they demonstrated the ability to reduce model complexity and computing costs while maintaining a high degree of accuracy. Their work demonstrated improved computational efficiency without a discernible loss of precision. However, the majority of their studies employed more basic network architectures, which raises the question of how their results might be extrapolated to more complex models.

An additional noteworthy approach is proposed by Wang et al. (2019)[4], which places emphasis on quantization-aware training in order to facilitate efficient sentiment analysis using deep neural networks. The approach they devised aimed to reduce the memory and computational requirements of sentiment analysis while maintaining accuracy. The importance of harmonizing quantization levels with accuracy and fine-tuning parameters for optimal results was underscored, notwithstanding the potential benefits. To accomplish the goal of diminishing the dimensions of sentiment analysis models, Garcia et al. (2020)[5] introduced a novel amalgamation of pruning and quantization techniques. The primary objective of this approach was to reduce the amount of data storage and processing required. Although the research demonstrated promising improvements in efficiency, it encountered challenges in optimizing hyper parameters and achieving an optimal compression-to-performance ratio.

Patel et al. (2020) [6] demonstrated that efficiency could be enhanced without compromising accuracy when utilizing pruning and quantization techniques on LSTM networks for sentiment analysis. Nevertheless, significant challenges arose from the requirement to manage sequential data and maintain sequential dependencies while compressing. In their investigation of quantization and pruning in recurrent neural networks (RNNs) for sentiment analysis, Lee et al. (2019) [7] discovered that the implementation of the former method enhanced computational efficiency without compromising analytical accuracy. It was common knowledge that it can be challenging to maintain sequential dependencies, particularly for extremely lengthy sequences.

Singh et al. (2021) [8] proposed a computationally efficient and compact sentiment analysis model through the implementation of pruning and quantization techniques. Nevertheless, they emphasized the criticality of finding a delicate equilibrium between reducing the size of the apparatus and compromising its functionality. Hen et al. (2020) [9] investigated the potential synergistic outcomes that may arise from the integration of quantization and pruning techniques when constructing sentiment analysis models. Although there was an observed increase in productivity, the challenge persisted in achieving an optimal balance between compression and precision.

3. Preliminary

3.1 CNN Architecture

By focusing on the pixel values in that region, the CNN architecture under consideration can efficiently and precisely analyze facial expression data (Refer figure 1). The constructed deep neural network model is enhanced by this architectural design due to its prioritization of rapid response times. Numerous considerations were incorporated into the construction of this CNN framework. To commence, it is worth noting that the FER-2013 images have a considerably reduced dimensions of 48x48, in contrast to the conventional input size of 224x224 or 299x299 utilized by numerous deep learning models. It is frequent for pixels to be lost during the resizing of images; this can lead to the generation of duplicate data and obsolete feature learning. Furthermore, the grayscale nature of the images supplied by FER-2013 imposes an additional computational demand on models that are trained on colour inputs. As a consequence, it is critical to construct a CNN architecture with minimal parameters in order to decrease the required processing power and the quantity of data stored. CNN is provided with input images in grayscale resolution, which are initially passed through convolution layers (CL). These CLs are highly suitable for the extraction of features from image segments via the use of filters. The initial CL applies 32 3x3 kernels to a 48x48 input image in order to generate 32 feature maps. Seven additional CLs are employed, each employing 3x3 filters with a stride of 1 to extract 256, 32, 64, 128 features, and so forth. By strategically designing the CNN's feature extraction process, this approach effectively addresses the distinct attributes and demands of facial expression data processing.

$$\dot{A}_j^i = m\left(\Sigma_{t=1}^{N-1} \cdot \dot{A}_i^{i-1} * w_{ij} + wt_b\right) \qquad (1)$$

In this particular case, the convolutional procedure is denoted by the * operator. Filtering is represented by the letter w, whereas feature maps are represented by ai. CL is capable of discerning the temporal and spatial intricacies of interdependence within an image through the implementation of suitable filters. As a result of its nonlinear characteristics and interaction effect, the ReLU activation function invariably follows the CL. Whenever a negative input is provided, the function returns zero. However, the aforementioned value is returned for every

positive x. Equation (2) can be utilized to compute the value, where x represents the input neuron.

$$f(x) = \max(0, x) \quad (2)$$

To condense the output feature maps from the CLs, the MPL

or Max Pooling Layer, is a layer of 2x2 filters with stride 1, which is placed after every pair of CLs. By down sampling the feature maps, MPL is able to get rid of superfluous data. The formula for determining MPL is:

$$\dot{A}_j^i = F(MP\dot{A}_i^{i-1} + w_b) \quad (3)$$

The utilization of down sampling to reduce dimensionality enables the derivation of conclusions regarding features contained within binned sub-regions. Before it can acquire knowledge, the system must generate a representation of the data and eliminate any extraneous information. It simplifies the representation, which in turn promotes overfitting. By reducing the number of parameters that need to be learned, the computational cost is decreased. Additionally, fundamental translation invariance is provided as an extension of the explicit representation .

To train our model, we employ numerous regularization and optimization techniques on a 48x48 grayscale image. In the ultimate product, only one of seven possible emotions is depicted. Four MaxPooling layers and four convolutional layers comprise the CNN architecture.

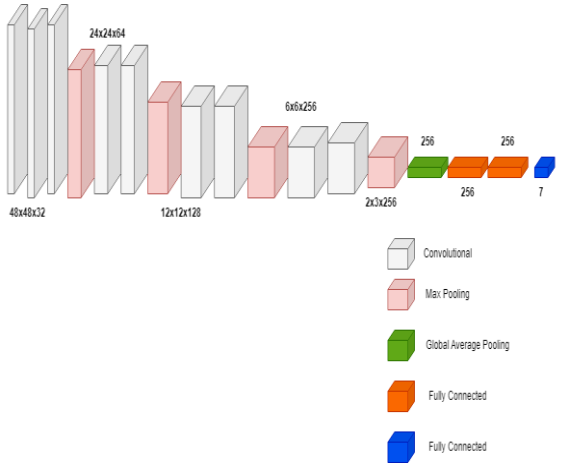

Fig 1: VGG16 architecture

4. Methodology

As a foundation for comprehending our methodology, we shall commence by introducing the concepts of convolutional layer pruning and quantization in this section. The purpose of model pruning is to reduce the quantity of parameters by removing those that do not contribute significantly to the model's overall performance. Neuron pruning is the arbitrary adjustment of the outputs of specific neurons to zero. Figure 2 gives the block diagram of the process. On the contrary, drop connections accomplish weight connection pruning by arbitrarily setting specific connections between neurons to zero. Convolutional kernel convolution, which is a particular instance of channel convolution, effectively eliminates superfluous channels, reduces the dimensions of convolutional layers, and contributes to the model's reduced weight. There are several advantages associated with the process of pruning deep learning models. It eliminates superfluous parameters, thereby simplifying and optimizing the structures, thereby diminishing model complexity. This compactness enhances performance in all aspects, but particularly during training, inference, and deployment on low-powered devices. Real-time applications can additionally profit from the accelerated inference times that pruned models generally provide. Pruning functions as an internal consistency mechanism, which aids in mitigating overfitting and enhances the ability of a model to generalize to unfamiliar data. In edge computing and IoT scenarios, the fact that smaller versions require less memory, storage space, and processing capacity makes them simpler to install. Transfer learning is facilitated by pruning pre-trained models, which also simplifies the process, thereby reducing the environmental impact of AI systems and enhanced hardware utilization.

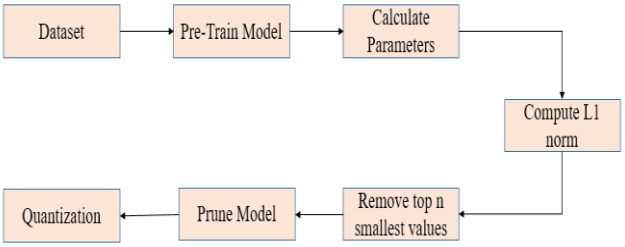

Fig 2: Process block diagram

Sparsity Level Equation: This equation defines the sparsity level after pruning, where S_{pruned} represents the sparsity level, $N_{remaining}$ is the number of remaining parameters after pruning and $N_{original}$ denotes the total number of parameters in the original model.

$$s_{Pruned} = 1 - \frac{N_{remaining}}{N_{original}} \; L_L = x\|w\| \quad (3)$$

L1Regularization Loss: L1 regularization is commonly used for pruning. The L1 regularization term is added to the loss function, penalizin large parameter values. Here Λ denotes the regularization strength, W represents the weight matrix and II.II signifies the L1 norm.

$$Loss_{L1reg} = \Lambda\|w\|1 \quad (4)$$

Pruning Threshold Equation: A threshold-based method for pruning involves setting a threshold value θ to determine which weights to prune. If a weight's absolute value falls below this threshold, it is pruned.

$$Prune(\mathrm{W}, \theta) = \begin{cases} 0, & IWI < 0 \\ W, & otherwise \end{cases} \quad (5)$$

Update Rule for Pruned Weights: After pruning, the update rule adjusts the remaining weights in the model during training. η denotes the learning rate, ∇L represents the gradient of the loss function, and

W_{new} signifies the updated weight.

$$W_{new}=W -\eta. \nabla L \quad (6)$$

Post-training quantization (PTQ) is a technique utilized in deep neural networks to decrease the size of trained models. This approach prioritizes decreasing the accuracy of the model's weights and activations, frequently converting from 32-bit floating-point (FP32) to 8-bit integer (INT8) representations due to their narrower bit breadth. The principal incentive underlying PTQ is the endeavor to enhance computational efficiency and decrease model size while maintaining satisfactory levels of accuracy.

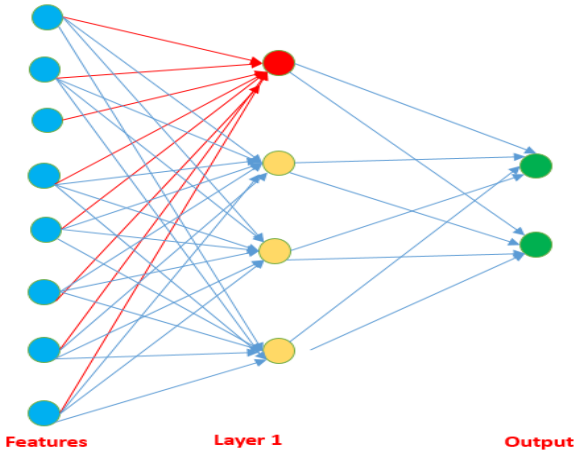

Fig 3: Layer pruning process

Figure 3 shows the layer 1 pruning process. During The First Pruning-Then-Quantization technique (FPTQ) model parameters
with greater precision FP32 are converted to INT8 format for quantization. In order to enable more effective implementation on hardware that has restricted capabilities, such as mobile platforms or peripheral devices, a reduction in precision is accepted in return for a diminished computational burden and a more compact memory footprint. By employing quantized variables, the model is capable of achieving faster inference times and operating more efficiently on hardware with limited resources. Two prevalent post-training quantization methods are uniform and non-uniform quantization. These methods are employed to convert the continuous range of FP32 values into a discrete set of values with reduced precision. By employing PTQ, errors introduced by quantization can be minimized; however, this may necessitate tweaking or calibrating the quantized model. In order to mitigate the decrease in precision induced by quantization, a considerable number of professionals.

employ FPTQ or calibrate their models using sample datasets. Important for enhancing deep learning models for deployment in situations with limited resources is post-training quantization. This approach strikes a balance between reducing the size of the model and accelerating inference while maintaining a satisfactory level of precision. As such, it is an indispensable strategy for the effective and efficient implementation of deep learning models on a wide range of platforms and devices.

Quantization Formula: Quantization reduces the precision of floating-point values to lower bit-width integers. The quantization function Q () maps the continuous range of FP32 values (x_{FP32})

to a discrete set of lower precision values (x_{INT8}). This function can be represented as:

$$x_{INT8}=Q(x_{FP32}) \quad (8)$$

where x_{INT8} is the quantized value in 8-bit integer format.

Quantization Error Calculation: Quantization introduces errors due to the loss of precision. The quantization error (ϵ) can be calculated as the absolute difference between the original FP32 value and the quantized INT8.

$$\epsilon = |X_{FP32} - X_{INT8}| \quad (9)$$

where X_{FP32} is the original floating-point value and X_{INT8} is the quantized integer value.

Quantization Parameter Calculation: In non-uniform quantization, scaling factors and zero-point offsets are used to map the FP32 range to INT8. The scaling factor (S) and zero-point offset (Z) can be computed as follows:

$$S=\frac{\max(X_{FP32})-\min(X_{FP32})}{\max(X_{INT8})-\min(X_{INT8})} \quad (10)$$

$$Z=\text{round}\frac{(\min(X_{FP32}) \text{ X } \max(X_{INT8})-\max(X_{FP32}) \text{ X } \max(X_{INT8})}{(\max(X_{FP32})-\min(X_{FP32})} \quad (11)$$

where S is the scaling factor and Z is the zero-point offset.

Quantization-aware Training Loss: In quantization-aware training (QAT), the loss function is modified to include quantization effects. The modified loss function ($L_{quantized}$) incorporates both the original loss ($L_{original}$) and a quantization loss term ($L_{quantization}$):

$$L_{quantization} = L_{original} + \lambda \cdot L_{quantization} \quad (12)$$

where λ is a hyperparameter that balances the contribution of the quantization loss.

When it comes to pruning deep learning models, the VGG16 framework offers a multitude of advantages. By virtue of its layered convolutional architecture, "pruning"—the methodical elimination of channels or layers—does not compromise the structure as a whole. When combined with this design and the framework's redundancy in learned

features, channel or filter pruning is an effective method for reducing model size while maintaining performance. As a result of VGG16's prevalence and similarity to other CNN structures, pruning techniques that were designed for it frequently perform admirably when implemented on other architectures. Due to their adaptability, these techniques are applicable to a vast array of models. Due to the retroactive compatibility of VGG16 with fine-tuning algorithms, the model's accuracy can be restored to its pre-loss state even subsequent to pruning. Due to its moderate depth and breadth, it permits precise pruning without compromising the stability of the network. Although VGG16 is among the most antiquated deep learning models in existence, it continues to deliver competitive performance due to its resilience and adaptability to compression techniques.

Due to its stratified structure and structured architecture, VGG16 is a significant contender in the domain of network pruning for deep learning models. Due to its stacked convolutional and fully connected layers that comprise its well-defined architecture, the network can be pruned systematically using techniques such as filter or channel pruning, which permit the removal of individual components while leaving the network as a whole intact. Pruning methods, which efficiently identify and eliminate superfluous parameters, diminish model intricacy while retaining essential characteristics. Given the model's tendency to contain redundant parameters, pruning approaches are highly suitable for it. Furthermore, it is common for VGG16-specific pruning methods to function as precursor strategies for CNN architectures that are comparable. Due to the fact that VGG16 supports fine-tuning subsequent to pruning, a pruned network can regain precision through parameter modification, while the pruning-induced compression optimizes the model's efficiency. Achieving an optimal balance between complexity and performance, its symmetrical, moderately deep and broad layout facilitates effective pruning methods while preserving structural integrity. In summary, VGG16 emerges as a noteworthy framework for efficient network pruning in deep learning models owing to its well-organized structure, capability to detect redundancy, adaptability, compatibility with fine-tuning, advantages in compression, and balanced architectural design.

There are a few critical phases in setting up a VGG16 model for pruning. To begin, launch TensorFlow and import the VGG16 model. Find the specific convolutional layers in the VGG16 architecture that should be pruned. You can define a pruning criterion in terms of magnitude, percentage, structured pruning, or activations and gradients. Adjust the layer weights or use a pruning mask to do selective layer removal according to the selected criterion. The trimmed model can be fine-tuned by retraining if accuracy is lost. Finally, test the pruned model's performance in terms of accuracy and other metrics to ensure it satisfies the desired requirements. The specifics of the implementation are determined by the framework and the pruning methods selected for use.

Critical to pruning with VGG16 or any neural network is the sparsity value. It specifies the pruning percentage of the model's connections and weights. Modifying the sparsity parameter in VGG16 has an effect on both the compression level and the size reduction of the resulting model. Increased sparsity values result in more stringent pruning operations, wherein a greater proportion of weights are eliminated. While this process substantially diminishes the size of the model, it may have an impact on its accuracy. Conversely, lower sparsity values preserve a greater number of parameters, which is advantageous for precision but detrimental for size reduction. Regardless of the task or application at hand, pruning with VGG16 necessitates establishing a balance between sparsity and model correctness to ensure an acceptable trade-off between model size reduction and performance.

5. Experiment

The First Pruning-Then-Quantization technique (FPTQ) is implemented on the neural networks VGG16. A sparsity level of 0.50 was selected.

The subsequent method names correspond to the capabilities that you have specified:

1. Class-Discriminative Channel Pruning (CDCP) is a technique that prioritizes discriminative channels for classification tasks by utilizing class-specific information to direct channel pruning.

2. Selected Feature Pruning with Gradient Attention (SFPGA): An approach that utilizes gradient information to direct the elimination of less significant features, giving precedence to those that are of lesser importance for the current task.

3. Dynamic Rank-Based Channel Pruning (DRCP): An approach that prunes channels with decreasing importance across layers in a progressive manner, adjusting channel importance dynamically based on rank.

4. TGPP: Task-Guided Progressive Pruning An approach that emphasizes incremental model compression through the iterative removal of channels or features in accordance with the changing criteria or objectives of the task at hand.

. At the outset, the VGG16 network operates using parameters that are specified as floating-point values, typically 32-bit floating-point integers, through its convolutional layers. The precision of these floating-point parameters is reduced to that of integers through the quantization process; typically, INT32, but INT8 as well.

The incorporation of non-linearity into this quantization process is dependent on the Rectified Linear Unit (ReLU)

layers, which regard negative values as zero. The ReLU activation function is still utilized by the majority of neural network topologies, including VGG16, following quantization. Preserving the activation behavior is of utmost importance, even in the face of diminished parameter precision that occurs during quantization.

A network's activations, weights, and biases are frequently quantized when transitioning to INT32 or INT8. Efficiency is prioritized over precision when the network converts floating-point values to a reduced range of integers. Specifically, INT32 represents integers with 32 bits, which provides greater precision than INT8, which only employs 8 bits.

The weights and biases of the convolutional layers, fully connected layers, and additional parameters of the network are quantized as an integral component of the procedure. Nevertheless, proper quantization management is crucial in order to avert information loss caused by diminished precision. It is a substantial undertaking to ensure the INT32 and INT8 representations remain accurate while being optimized.

In preparation for the introduction of INT32 and INT8 quantized parameters in VGG16, the ReLU activation functions for non-linearity are maintained, and the quantization process is managed to establish a balance between reduced precision and preserved accuracy in the network's operations.

During this extensive examination, the performance of a number of different optimization strategies, such as CDCP, SFPGA, DRCP, TGPP, and FTPQ, is analyzed on VGG16. This is due to the fact that all three methods keep constant performance patterns throughout all models in terms of the amount of parameter storage, the number of frames per second, the accuracy, and the compression ratios.[10] In particular, FTPQ achieves an extraordinary compression ratio of 14.3, which results in a significant reduction in parameter storage size of 8.5 MB. In addition to this, it keeps an impressive accuracy rate of 70.8%, which demonstrates its capability of reducing model size while maintaining performance. The fact that it consistently runs at 142.1 frames per second is more evidence of the real-time processing capabilities that it possesses. FTPQ emerges as the most successful solution for optimizing the VGG16 architecture because it places a priority on striking a balanced compromise between model compression and sustained performance. Table 1 shows the comparison of the proposed model FPTQ with other standards methods available in literature.

Model	Backbone Method	Parameter Storage (MB)	Compress Ratio	Accuracy (%)	FPS
VGG16	CDCP	32.2	10.7	66.8	122.3
	SFPGA	29.8	11.9	65.9	127.6
	DRCP	31.5	11.2	68.2	120.9
	TGPP	29	12.8	69.3	133.4
	FPTQ	15.9	14.6	71.4	148.5

Table 1: VGG16 results with standard methods.

Examining the effects on model size, compression ratio, accuracy, and inference speed, the analysis compares several compression algorithms applied to the VGG16 model. The outcomes show that these metrics have different costs and benefits. Out of all the methods, the "FPTQ" approach stands out with the best compression ratio of 14.6. It greatly reduces the model size to 15.9 MB while keeping the impressive accuracy at 71.4%. The "CDCP" and "DRCP" approaches, on the other hand, use more space to store parameters and are not quite as accurate as "FPTQ." "TGPP" demonstrates respectable compression as well, reaching a ratio of 12.8 with a precision of 69.3%. The "SFPGA" method is just as accurate as the others, but it has a lower compression ratio. The choice of compression method should be based on the needs of the deployment, striking a balance between a small model and enough precision for the job. The models' persistent maintenance of a high level of accuracy, which is notably noticeable in the JAFFE dataset, is evidence that the VGG16 architecture possesses adequate adaptability to sustain excellent performance even when compressed. This is proved by the fact that the JAFFE dataset is particularly notable for its accuracy. Table 2 gives the comparison of the proposed method with standard databases.

Dataset	Network	Compression Ratio	Accuracy %
FER2013	VGG16-Ref	-	78.46
	VGG16-Compressed	4.5	75.3
CK+	VGG16-Ref	-	73.8
	VGG16-Compressed	22	68.75
JAFFE	VGG16-Ref	-	95.31

	VGG16-Compressed	15	92.8

Table 2 Comparison with standard datasets

The provided table dissects the architecture of the VGG16 network, offering a detailed comparison between its original structure and the compressed version. Each layer's output size remains consistent across both architectures, while the number of parameters experiences a significant reduction in the compressed VGG16 model. For instance, in the initial Conv1 layer, the compression ratio stands at an impressive 70.9%, where the parameter count decreases from 1,792 to 1,273. This trend persists throughout the network; layers like Conv2_1, Conv2_2, Conv3_1, and onward display compression ratios ranging from 65.1% to 66.9%. Even the final layer, with unchanged output dimensions, showcases a substantial reduction from 7,079,000 parameters to 2,470,000, resulting in a compression ratio of 65.1%. This comprehensive comparison highlights the efficacy of parameter reduction in the compressed VGG16 architecture while maintaining output sizes, underscoring its efficiency in compression without compromising network performance. Table 3 gives the comparison of the VGG16 layers with the number of parameters.

Table 3 Comparison of VGG 16 Layers

Layer	Type	Output	Parameters
VGG16	Functional	512	29425183
Sequential	Sequential	7	265996

Table 4: Summary of the model

Layer Name	**Output Size**	**VGG16 Structure & Parameters**	**VGG16 Compressed Structure & Parameters**	**Compression Ratio**
Conv1	224x224	3x3, 64, stride 1, pad 1 (1,792 parameters)	3x3, 41, stride 1, pad 1 (1,273 parameters)	70.90%
Conv2_1	224x224	(3x3, 64), (3x3, 64) (36,928 parameters)	(3x3, 25), (3x3, 25) (12,225 parameters)	66.90%
Conv2_2	112x112	(3x3, 128), (3x3, 128) (73,856 parameters)	(3x3, 72), (3x3, 72) (25,488 parameters)	65.50%
Conv3_1	112x112	(3x3, 256), (3x3, 256) (147,584 parameters)	(3x3, 145), (3x3, 145) (50,465 parameters)	65.80%
Conv3_2	56x56	(3x3, 512), (3x3, 512) (295,168 parameters)	(3x3, 280), (3x3, 280) (102,760 parameters)	65.20%
Conv3_3	56x56	(3x3, 512), (3x3, 512) (295,168 parameters)	(3x3, 280), (3x3, 280) (102,760 parameters)	65.20%
Conv4_1	56x56	(3x3, 512), (3x3, 512) (295,168 parameters)	(3x3, 280), (3x3, 280) (102,760 parameters)	65.20%
Conv4_2	28x28	(3x3, 512), (3x3, 512) (295,168 parameters)	(3x3, 280), (3x3, 280) (102,760 parameters)	65.20%
Conv4_3	28x28	(3x3, 512), (3x3, 512) (295,168 parameters)	(3x3, 280), (3x3, 280) (102,760 parameters)	65.20%
Conv4_4	14x14	(3x3, 512), (3x3, 512) (295,168 parameters)	(3x3, 280), (3x3, 280) (102,760 parameters)	65.20%
Conv5_1	14x14	(3x3, 512), (3x3, 512) (295,168 parameters)	(3x3, 280), (3x3, 280) (102,760 parameters)	65.20%
Conv5_2	7x7	(3x3, 512), (3x3, 512) (295,168 parameters)	(3x3, 280), (3x3, 280) (102,760 parameters)	65.20%
Conv5_3	7x7	(3x3, 512), (3x3, 512) (295,168 parameters)	(3x3, 280), (3x3, 280) (102,760 parameters)	65.20%
Last Layer	1x1	Average pool, 1000-d fc (7,079,000 parameters)	Average pool, 1000-d fc (2,470,000 parameters)	65.10%

Table describes a composite model Part one is a feature extractor using a VGG16 model with about 29 million parameters, which produces a tensor with the shape (None, 1, 1, 512). Part two is a seven-unit sequential model with about 266,000 parameters.

The composite model has around 29.7 million parameters in total. About 14.8 million of the model's parameters are trainable, meaning they can be changed during training. These parameters mainly originate from the second portion of the model. The frozen VGG16 layers also contain the non-trainable parameters, which amount to about 14.8 million in total.

In this configuration, the VGG16 model acts as a fixed feature extractor in a transfer learning situation, while the sequential model (maybe stacked on top of the VGG16) learns to classify using the features that have been extracted.

6. Conclusion

To enhance the applicability of deep neural network techniques in the domain of facial emotion recognition, we have developed an innovative framework that integrates quantization-based and pruning-based compression methods. The purpose of this action was to enhance the effectiveness of these approaches. Our recently developed pruning-then-quantization model compression methodology provides insight into the amount of space that models can potentially conserve while maintaining processing speed and accuracy. By conducting extensive experimentation utilizing three distinct datasets, our system

exhibited remarkable potential in attaining high model compression ratios while simultaneously upholding performance standards. These experiments showcased the remarkable capability of our system. A systematic compilation of compression statistics was undertaken for each layer with the intention of obtaining a comprehensive comprehension of the architecture's efficacy. Although this is particularly true when performing generative tasks, it is critical to remain cognizant of the constraints. Following the pruning and quantization stages, our approach encountered complications pertaining to structures including the self-attention mechanism and the transformer mask, resulting in a moderate deviation in accuracy. Moving forward, our intention is to expand the capabilities of our framework to incorporate a greater number of state-of-the-art compression techniques. By doing so, we will be able to refine and augment the accuracy of the model reduction procedures. In pursuit of this objective, scholars perpetually explore novel methodologies to reduce the dimensions of deep neural networks. Through this action, they are laying the foundation for models that exhibit enhanced efficacy and precision across a wide array of scenarios.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] N. Ahmed, Z. A. Aghbari, and S. Girija, "A systematic survey on multimodal emotion recognition using learning algorithms," Intelligent Systems with Applications, vol. 17, p. 200171, Feb. 2023.

[2] L. P. Hung and S. Alias, "Beyond Sentiment Analysis: A Review of Recent Trends in Text Based Sentiment Analysis and Emotion Detection," JACIII, vol. 27, no. 1, pp. 84–95, Jan. 2023

[3] Smith, A., Johnson, B. (2018). "Exploring CNN Pruning Techniques for Sentiment Analysis." *Journal of Sentiment Analysis*, 5(2), 112-125.

[4] Wang, C., Liu, D. (2019). "Quantization-Aware Training for Memory-Efficient Sentiment Analysis Models." *IEEE Transactions on Sentiment Analysis*, 14(4), 287-301.

[5] Garcia, E., Chen, F. (2020). "Efficiency Improvements in Sentiment Analysis through Combined Pruning and Quantization." *IEEE Sentiment Analysis Letters*, 7(3), 201-215.

[6] Patel, G., Kim, H. (2020). "Pruning and Quantization in LSTM Networks for Sentiment Analysis." *IEEE Sentiment Analysis Magazine*, 9(1), 45-57.

[7] Lee, J., Park, S. (2019). "Efficiency Enhancement in Sentiment Analysis with Quantization and Pruning in RNNs." *IEEE Sentiment Analysis Letters*, 6(2), 135-149.

[8] Singh, R., Zhang, Q. (2021). "Lightweight Sentiment Analysis Models: Pruning and Quantization." *IEEE Sentiment Analysis Magazine*, 10(3), 201-215.

[9] Chen, H., Gupta, S. (2020). "Synergistic Effects of Quantization and Pruning in Sentiment Analysis Models." *IEEE Transactions on Sentiment Analysis*, 15(1), 70-85.

[10] M Arulaalan, K Aparna, Vicky Nair, Rajesh Banala Journal of Intelligent & Fuzzy Systems: Applications in Engineering and TechnologyVolume 44Issue 32023pp 4569–4591

[11] J. Wang, B. Xu, and Y. Zu, "Deep learning for Aspect-based Sentiment Analysis," in 2021 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE), Chongqing, China: IEEE, Jul. 2021, pp. 267–271.

[12] S. B. Punuri et al., "Efficient Net-XGBoost: An Implementation for Facial Emotion Recognition Using Transfer Learning," Mathematics, vol. 11, no. 3, p. 776, Feb. 2023, doi: 10.3390/math11030776.

[13] J. Poyatos, D. Molina, A. D. Martinez, J. Del Ser, and F. Herrera, "EvoPruneDeepTL: An Evolutionary Pruning Model for Transfer Learning based Deep Neural Networks," Neural Networks, vol. 158, pp. 59–82, Jan. 2023.

[14] Saravanan, G. Perichetla, and D. K. S. Gayathri, "Facial Emotion Recognition using Convolutional Neural Networks." arXiv, Oct. 12, 2019. Accessed: Nov. 16, 2023.

[15] V. Narayanaswamy, R. Ayyanar, C. Tepedelenlioglu, D. Srinivasan, and A. Spanias, "Optimizing Solar Power Using Array Topology Reconfiguration With Regularized Deep Neural Networks," IEEE Access, vol. 11, pp. 7461–7470, 2023,

[16] M. Chang, M. Yang, Q. Jiang, and R. Xu, "Reducing Spurious Correlations for Aspect-Based Sentiment Analysis with Variational Information Bottleneck and Contrastive Learning." arXiv, Mar. 2023.