SE-Pruned ResNet-18: Balancing Accuracy and Efficiency for Object Classification on Resource-Constrained Devices

Zeyad Farisi
Applied College, Taibah University, Medina City, Saudi Arabia

***Abstract*—Deep learning-based image object classification methods often achieve high accuracy, but with the growing demand for real-time performance on resource-constrained edge devices, existing approaches face challenges in balancing accuracy, computational complexity, and model size. To alleviate this awkward situation, we propose a novel ResNet-18 architecture that integrates the Squeeze-and-Excitation (SE) module and model pruning. The SE module adaptively emphasizes informative feature channels to enhance classification accuracy, while pruning technology reduces computational costs by removing unimportant connections or parameters without significant accuracy loss. Extensive experiments on benchmark datasets demonstrate that the optimized model outperforms the original ResNet-18 in both accuracy and inference speed. The classification accuracy increases from 93.2% to 94.1%, the number of parameters is reduced by 30%, the Floating-Point Operations decreases from 1.81 giga to 1.32 giga, and the inference time is decreased from 15.2 milliseconds to 12.8 milliseconds per batch. What's more, the proposed model outperforms MobileNetV2, ShuffleNetV2, and EfficientNet-B0 in accuracy while maintaining competitive inference speed and parameter count. The experimental results highlight the model's potential for deployment on resource-constrained devices, expanding the practical application scenarios of object classification methods in edge computing and real-time detection tasks.**

***Keywords*—ResNet-18; squeeze-and-excitation model; model pruning; object classification; resource-constrained devices**

I. INTRODUCTION

In the era of digital transformation, object classification, as a fundamental task in computer vision, plays a crucial role in various applications, including intelligent surveillance, industrial automation, and autonomous driving. Deep learning, especially convolutional neural networks (CNNs), has been the mainstream of image-based object classification. Among numerous CNN architectures, ResNet-18, proposed by He et al. [1] stands out for its simplicity and accuracy, providing a practical baseline for many classification tasks. However, with the increasing demand for real-time performance and resource-constrained edge devices, the original ResNet-18 faces challenges in balancing accuracy, computational complexity, and model size.

The first challenge lies in feature representation. Although ResNet-18 alleviates the vanishing gradient problem through skip connections, it treats all feature channels equally, which may lead to the neglect of critical object characteristics and thus limit the classification accuracy. To address this issue, the SE module, introduced by Hu et al. [2], was proposed. By learning channel-wise attention, the SE module can selectively enhance discriminative features and suppress redundant information, thereby improving the recognition performance of the model.

The second challenge is computational efficiency. In many practical applications, models are often deployed on devices with limited computing resources and memory. The relatively large number of parameters in the original ResNet-18 may result in slow inference speed and high energy consumption, making it difficult to meet real-time requirements. Model pruning, a well-studied technique in deep learning compression, offers an impressive approach to reduce model complexity. By removing unimportant connections or parameters, pruning can significantly decrease the model parameters and the computational cost while maintaining acceptable accuracy.

In this study, we aim to optimize the ResNet-18 architecture for object classification by integrating the SE module and model pruning techniques. Specifically, we embed the SE module into each residual block of ResNet-18 to enhance feature discrimination and apply structured pruning based on the L1 norm of weights to reduce model complexity.

The rest of this study is organized as follows: Section II reviews the related work on object classification, the SE module, and model pruning. Section III details the proposed methodology. Section IV presents the experimental setup, results, and analysis. Finally, Section V concludes the study and discusses future research directions.

II. RELATED WORK

A. *Object Classification with Deep Learning*

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has been a significant milestone in the development of image-based object classification using deep learning. In 2012, AlexNet [3], proposed by Krizhevsky et al., achieved a top-5 error rate of 15.3% on the ImageNet dataset. This breakthrough inspired a wave of research in developing more advanced CNN architectures. Subsequently, numerous architectures were proposed to improve classification accuracy. VGGNet [4], developed by Simonyan and Zisserman in 2014, achieved a lower error rate than AlexNet. However, the increase in depth also led to a significant increase in the number of parameters. ResNet addressed the problem of vanishing gradients in very deep neural networks through the introduction of residual connections. This allowed for the training of extremely deep networks, and ResNet-18, a lightweight variant, became widely used due to its good balance between performance and computational cost. ResNet architectures achieved state-of-the-art results in various image classification benchmarks.

In recent years, there have been continuous efforts to further improve the accuracy of image-based object classification. Methods like DenseNet [5], which introduced dense connections between layers to promote better information flow and feature reuse. As of now, the most accurate methods [6] [7] in image-based object classification often involve highly complex architectures with a large number of parameters. They are extremely large in size, with a vast number of parameters that require significant computational resources for training and inference. As a result, these models are not suitable for lightweight application scenarios such as edge computing, where devices have limited memory, computing power, and energy resources.

B. SE Module

The SE module is a powerful technique for improving the representational power of CNNs. Unlike traditional CNNs that process all feature channels equally, the SE module adaptively recalibrates the importance of different channels, which has been successfully integrated into various CNN architectures, leading to significant performance improvements. Subsequently, many researchers focused on further improving CNN architectures by leveraging the SE module. For example, in [8], the authors proposed a modified SE-ResNet architecture. They adjusted the position and number of SE modules within ResNet to better capture the hierarchical feature information. Another study [9] combined the SE module with the Inception architecture, creating an SE-Inception network. The new network was able to distinguish between different object classes by emphasizing the important channels in the feature maps, resulting in improved performance on large-scale image classification tasks. An SE-based CNN was developed for the classification of retinal fundus images in reference [10], which is crucial for the early detection of eye diseases. The SE module helped the model to focus on the relevant anatomical structures in the images, improving the classification performance for different disease categories. In [11], the authors applied the SE module in a YOLO-based network for object detection and classification. Reference [12] combined the SE module with attention mechanisms (spatial attention) to create hybrid attention networks. In addition, researchers have also integrated the SE module with generative models [13].

C. Model Pruning Techniques

Model pruning techniques have emerged as a crucial approach in the field of deep learning for reducing model complexity, computational cost, and memory footprint while maintaining acceptable performance. These techniques aim to remove unimportant or redundant parts of a pre-trained model, making it more efficient for deployment on resource-constrained devices such as mobile phones, embedded systems, and edge computing platforms. LeCun et al. [14] in 1989 proposed a simple form of pruning for neural networks. They removed small-magnitude weights from a neural network, demonstrating that a significant portion of the weights could be eliminated without sacrificing much accuracy. Han et al. [15] introduced an aggressive pruning method in 2015, where they removed a large percentage of the smallest-magnitude weights from deep neural networks. They combined pruning with quantization and Huffman coding to achieve significant compression of neural network models. Reference [16] pruning AlexNet and VGG-16 demonstrated that it was possible to reduce the number of parameters by up to 90% while maintaining similar accuracy on the ImageNet dataset. Molchanov et al. [17] proposed a method for pruning convolutional neural network (CNN) channels. They used a Taylor expansion-based importance measure to identify and remove less important channels. In addition, pruning entire layers of a neural network has also been explored. Guo et al. [18] proposed a layer-level pruning method, where they analyzed the contribution of each layer to the overall network performance and removed the least important layers. Zhou et al. [19] proposed a data-dependent sparse structure selection method. They trained a small neural network to predict the importance of each weight in a large pre-trained network based on the input data. This data-driven approach could adaptively identify the weights that were more crucial for the specific dataset, leading to more effective pruning. AutoML-Zero [20] used reinforcement learning to automatically discover and optimize neural network architectures, including the pruning process. In this framework, an agent learns to make pruning decisions by interacting with the model and receiving rewards based on the performance (such as accuracy and model size) of the pruned model. This approach has the potential to find more optimal pruning strategies compared to traditional heuristic-based methods.

III. Methodology

A. Integration of SE Module into ResNet-18

1) Architecture of SE-ResNet-18: The core idea of integrating the SE module into ResNet-18 is to enable the network to adaptively recalibrate the importance of different feature channels, thereby enhancing its discriminative power for object classification. ResNet-18 consists of four residual blocks, each containing two convolutional layers with skip connections. In our proposed architecture, we embed an SE module into each residual block, forming the SE-ResNet-18. The model structure can be seen in Fig. 1.

The input image first goes through a 7x7 convolutional layer for initial feature extraction, followed by a max pooling layer for down-sampling. Four residual blocks (each containing an SE module) further extract hierarchical features. A global average pooling layer compresses spatial dimensions, and a fully connected layer generates classification outputs. The structure of residual blocks in Fig. 1 can be seen in Fig. 2.

The residual block consists of two 3x3 convolutional layers with batch normalization and ReLU activation, the output of the second convolutional layer is fed into the SE module, then summed with the input feature map (skip connection) and activated by ReLU. The structure of SE block in Fig. 2 can be seen in Fig. 3.

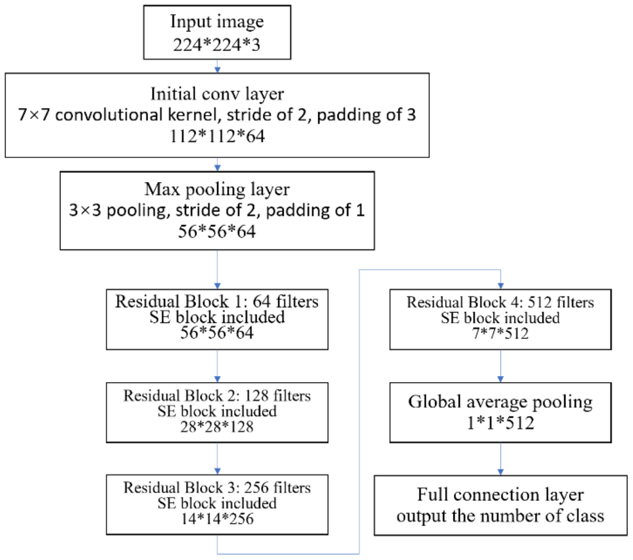
Input image
224*224*3
Initial conv layer
7×7 convolutional kernel, stride of 2, padding of 3
112*112*64
Max pooling layer
3×3 pooling, stride of 2, padding of 1
56*56*64
Residual Block 1: 64 filters
SE block included
56*56*64
Residual Block 2: 128 filters
SE block included
28*28*128
Residual Block 3: 256 filters
SE block included
14*14*256
Residual Block 4: 512 filters
SE block included
7*7*512
Global average pooling
1*1*512
Full connection layer
output the number of class

Fig. 1. The structure of SE-ResNet-18.

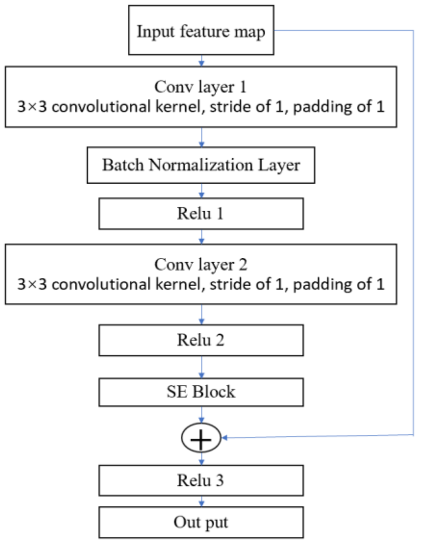
Input feature map
Conv layer 1
3×3 convolutional kernel, stride of 1, padding of 1
Batch Normalization Layer
Relu 1
Conv layer 2
3×3 convolutional kernel, stride of 1, padding of 1
Relu 2
SE Block
Relu 3
Out put

Fig. 2. The structure of residual block.

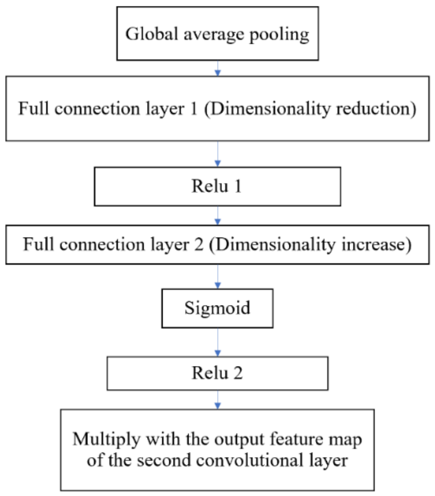
Global average pooling
Full connection layer 1 (Dimensionality reduction)
Relu 1
Full connection layer 2 (Dimensionality increase)
Sigmoid
Relu 2
Multiply with the output feature map
of the second convolutional layer

Fig. 3. The structure of SE block.

The SE module operates through three sequential steps: squeeze, excitation, and reweight. In the squeeze step, global average pooling is applied to the output feature maps of the second convolutional layer within each residual block. This operation compresses the spatial dimensions $(H \times W)$ of the feature maps into a single value for each channel, aggregating global context information across the entire feature map. Mathematically, for a feature map $X \in R^{C\times H\times W}$, the squeeze operation F_{sq} generates a channel-wise descriptor $z \in R^{C}$:

$$z_c = F_{sq}(X_c) = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W} X_c(i,j) \tag{1}$$

where, X_c represents the c-th channel of the feature map X.

In the excitation step, the channel-wise descriptor z is fed into a two-layer fully-connected network. The first fully-connected layer reduces the dimensionality of z from C to C/r (where r is the reduction ratio), followed by a ReLU activation function. The second fully-connected layer then restores the dimensionality back to C, and a Sigmoid activation function is applied to generate a set of channel-wise weights $s \in R^{C}$, where each element s_c represents the importance score of the c-th channel:

$$s = F_{ex}(z,W) = \sigma(W_2\delta(W_1 z)) \tag{2}$$

where, $W_1 \in R^{\frac{C}{r}\times C}$ and $W_2 \in R^{C\times\frac{C}{r}}$ are the weights of the fully-connected layers, δ is the ReLU function, and σ is the Sigmoid function. Finally, in the reweight step, the channel-wise weights s are used to scale the original feature maps X. The output of the SE module $Y \in R^{C\times H\times W}$ is calculated as:

$$Y_C = F_{scale}(X_c, s_c) = s_c \cdot X_c \tag{3}$$

By integrating the SE module in this manner, each residual block in SE-ResNet-18 can selectively enhance the channels that are more relevant to the object classification task, while suppressing less informative channels. This mechanism allows the network to focus on the most discriminative features, such as textures, shapes, and colors of the objects, thereby improving the overall classification accuracy.

2) Parameter and computational analysis: The integration of the SE module into ResNet-18 introduces additional parameters and computational costs. The extra parameters mainly come from the two fully - connected layers in the excitation step. For each SE module, the number of additional parameters is $C\times\frac{C}{r}+\frac{C}{r}\times C$. In ResNet-18, with a total of 8 residual blocks, the total number of additional parameters introduced by the SE modules is $8\times\left(C\times\frac{C}{r}+\frac{C}{r}\times C\right)$. However, compared to the overall number of parameters in ResNet-18 (about 11.7 million), the increase in parameters due to the SE module is relatively small when r is set appropriately. In terms of computational cost, the main operations added by the SE module are global average pooling, two fully - connected layers, and element - wise multiplication. Although these operations increase the computational load, the improvement in feature representation often outweighs the cost, especially in complex object classification tasks where fine-grained feature discrimination is required.

B. Model Pruning Strategy

1) Unstructured pruning based on L1 norm: To further optimize the SE-ResNet-18 for better computational efficiency and model size reduction, we adopt an unstructured pruning strategy based on the L1 norm of weight parameters. Unstructured pruning has the advantage of flexibility, as it can remove individual weights without imposing a specific structural constraint on the network.

The L1 norm of a weight parameter w is defined as $\| w \|_1 = | w |$. Weights with smaller L1 values contribute less to the output of the network and are thus considered less important for the classification task.

The pruning process consists of the following steps:

- Initial Training: First, we train the SE-ResNet-18 on the target object classification dataset until it converges. This step ensures that the network has learned a good set of feature representations.
- Weight Ranking: After training, we calculate the L1 norm for each weight parameter in the convolutional layers of the network. Weights are then ranked according to their L1 values in ascending order.
- Pruning Threshold Selection: A pruning threshold is set to determine the proportion of weights to be removed. For example, if we remove 30% of the weights with the smallest L1 values. In practice, we use a grid search method on a validation set to find the optimal that maximizes the test accuracy while achieving the desired level of parameter reduction.
- Weight Removal: We set the weights below the threshold to zero, effectively pruning them from the network.
- Fine-Tuning: After pruning, the network is fine-tuned on the training dataset for a certain number of epochs (in this study, we set it to 20 epochs). Fine-tuning helps the network recover from the performance degradation caused by weight removal and adapt to the new sparse structure.

2) Pruning-aware training: To further improve the effectiveness of pruning and mitigate the negative impact on model performance, we also incorporate pruning-aware training techniques. During the initial training phase of SE-ResNet-18, we add a regularization term to the loss function that encourages weight sparsity. Specifically, we use L1 regularization:

$$L_{regularized} = L_{task} + \lambda \sum_{w \in \Theta} | w | \tag{4}$$

where, L_{task} is the original loss function for the object classification task, λ is the regularization strength, and Θ is the set of all weight parameters in the network. This regularization term helps to make the weights smaller during training, making them more likely to be pruned in the subsequent pruning step.

C. Overall Optimization Process

The overall optimization process of our proposed method can be summarized as follows:

1) Initialization: Start with the original ResNet-18 architecture and initialize the network weights.

2) SE Module Integration: Embed the SE module into each residual block of ResNet-18 to form SE-ResNet-18.

3) Initial Training: Train the SE-ResNet-18 on the object classification dataset with data augmentation techniques (such as random cropping, horizontal flipping) and L1 regularization to encourage weight sparsity.

4) Model Pruning: After the initial training converges, apply the unstructured pruning based on the L1 norm to remove redundant weights.

5) Fine-Tuning: Fine-tune the pruned SE-ResNet-18 on the training dataset to recover and improve the classification performance.

6) Evaluation: Evaluate the optimized model on the test dataset to measure its accuracy, computational efficiency, and model size.

This iterative process of architecture modification, training, pruning, and fine-tuning aims to achieve an optimal balance between the classification accuracy and computational efficiency of the ResNet-18-based object classification model.

IV. Experiments

The CIFAR-10 dataset is employed to evaluate the proposed method. The dataset is partitioned into 50,000 training images and 10,000 test images. The following models are selected as baselines for comparison: there are Original ResNet-18, ResNet-18 with SE Module (SE-ResNet-18), Pruned ResNet-18 and our Pruned SE-ResNet-18. Evaluation metrics are Accuracy, Number of Parameters, Inference Time and FLOPs (Floating - Point Operations).

Experiments are conducted on a desktop computer equipped with an Intel Core i7-12700K CPU, 32GB RAM, and an NVIDIA GeForce RTX 3060 GPU. The deep learning framework used is PyTorch 1.13, and CUDA 11.6 is employed for GPU acceleration. To measure the inference speed fairly, each model's average inference time per batch is calculated by running 1,000 forward passes on the test dataset and taking the mean value.

The proposed SE-Pruned ResNet-18 achieves a test accuracy of 94.1%, outperforming the Pruned ResNet-18 by 1.3% and approaching the accuracy of SE-ResNet-18 (94.5%). This indicates that the combination of SE module and pruning can maintain high accuracy even with a reduced number of parameters.

The proposed SE-Pruned ResNet-18 reduces the number of parameters by approximately 27.4% compared to the original ResNet-18, from 11.7 million to 8.5 million, while the FLOPs are decreased by 27.1%. Notably, its inference time of 12.8 ms/batch is 15.8% faster than the original ResNet-18, demonstrating significant improvements in computational efficiency. Although the Pruned ResNet-18 has a slightly lower inference time (12.1ms/batch), its accuracy is lower than the proposed model, highlighting the trade-off between speed and accuracy. The experimental results are shown in Table I.

TABLE I. EXPERIMENTAL RESULTS PRESENTATION

Model	Parameters (M)	Flops (G)	Inference time (ms/batch)	Accuracy
Resnet	11.7	1.81	15.2	93.2
SE-Resnet	12.3	1.98	17.8	94.5
Pruned-Resnet	8.2	1.25	12.1	92.8
ours	8.5	1.32	12.8	94.1

To further validate the proposed model's competitiveness, we compared it with other lightweight architectures on the CIFAR-10 dataset. The results are shown in Table II:

TABLE II. COMPARISON WITH OTHER LIGHTWEIGHT ARCHITECTURES

Model	Parameters (M)	Flops (G)	Inference time (ms/batch)	Accuracy
ResNet	11.7	1.81	15.2	93.2
MoblieNet V2 [21]	2.2	1.32	7.5	92.1
ShuffleNet V2 [22]	2.2	0.30	6.8	91.5
EfficientNet B0 [23]	5.3	0.14	9.2	93.7
Ours	8.5	1.32	12.8	94.1

The proposed model outperforms MobileNetV2, ShuffleNetV2, and EfficientNet-B0 in accuracy while maintaining competitive inference speed and parameter count. Unlike lightweight models optimized for mobile devices, SE-Pruned ResNet-18 achieves a better balance for edge devices requiring moderate computational capabilities (e.g., IoT gateways, embedded systems).

The experimental results validate the effectiveness of the proposed approach. The integration of the SE module enhances the feature representation ability of ResNet-18, leading to improved classification accuracy, as evidenced by the performance of SE-ResNet-18. Meanwhile, the L1-norm-based unstructured pruning method significantly reduces the model size and computational cost without sacrificing much accuracy. By combining these two techniques, the proposed SE-Pruned ResNet-18 achieves an optimal balance among accuracy, model complexity, and inference speed.

Compared with the original ResNet-18, the proposed model requires fewer computational resources, making it more suitable for deployment on resource-constrained devices such as edge computing platforms, smartphones, and IoT devices. The reduced inference time also enables real-time object classification applications, expanding the practical application scenarios of the model.

V. CONCLUSION

This study focuses on optimizing the ResNet-18 architecture for object classification tasks by integrating the Squeeze-and-Excitation (SE) module and L1-norm-based unstructured pruning techniques. Through theoretical analysis and extensive experiments on the CIFAR-10 dataset, the proposed approach demonstrates remarkable performance in achieving a balance between accuracy, computational efficiency, and model size. In terms of accuracy, the optimized model (SE-Pruned ResNet-18) achieves a test accuracy of 94.1%, outperforming the pruned-only ResNet-18 (92.8%) and approaching the accuracy of the SE-integrated ResNet-18 (94.5%). This validates that the combination of feature enhancement via the SE module and parameter reduction through pruning can maintain high classification performance. Regarding model complexity, the proposed method successfully reduces the number of parameters by approximately 27.4% compared to the original ResNet-18, from 11.7 million to 8.5 million. Meanwhile, the Floating-Point Operations (FLOPs) are decreased by 27.1%, and the inference time is reduced by 15.8%, resulting in a 12.8 ms/batch inference speed. These improvements significantly enhance the computational efficiency of the model, making it 15.8% faster than the original architecture. The reduced model size and computational requirements of the optimized ResNet-18 lead to a notable decrease in hardware demands. The model can be effectively deployed on resource-constrained devices such as edge computing platforms, Internet of Things (IoT) devices, and smartphones, which typically have limited memory and computing power. In summary, the proposed approach breaks the traditional trade-off between model performance and resource consumption, offering a more efficient and versatile solution for deep-learning-based object classification.

However, there is still room for improvement. For example, further optimization of the pruning threshold selection algorithm or exploring more advanced pruning strategies may further enhance the performance of the model. Additionally, the generalization ability of the proposed method on other datasets remains to be further investigated.

REFERENCES

[1] He K , Zhang X , Ren S ,et al，"Deep Residual Learning for Image Recognition," IEEE conference on computer vision and pattern recognition, PP: 770 – 778，2016.

[2] Hu J , Shen L , Sun G ,et al. "Squeeze-and-Excitation Networks, " IEEE Transactions on Pattern Analysis and Machine Intelligence, PP:99-121. 2017.

[3] Krizhevsky, A., Sutskever, I., & Hinton, G. E, "ImageNet Classification with Deep Convolutional Neural Networks,". In Proceedings of the Neural Information Processing Systems (NIPS),2016.

[4] Simonyan, K., & Zisserman, "Very deep convolutional networks for large - scale image recognition." arXiv preprint arXiv:1409.1556, 2014.

[5] Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. "Densely Connected Convolutional Networks," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2261-2269. Honolulu, HI, July 21-26, 2017.

[6] Rao, Y., et al. "Dynamic vision transformer with adaptive token sparsity," the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 16507 – 16516, 2023.

[7] Ali, A., et al. "EdgeNeXt: Efficiently Amalgamated CNN - Transformer Architecture for Mobile Vision Applications." arXiv preprint arXiv:2206.10589, 2022.

[8] Zhang, X., et al. “An Improved Squeeze - and - Excitation ResNet for Image Classification.” The International Conference on Image Processing, Computer Vision, and Pattern Recognition (pp. 1-7), 2023.

[9] Wang, Y., et al. “SE - Inception: Enhancing Inception Networks with Squeeze and Excitation Modules for Image Classification.” Journal of Visual Communication and Image Representation, 71, 102796, 2020.

[10] Chen, M., et al. “Retinal Fundus Image Classification Using Squeeze - and - Excitation - based CNN.” Computers in Biology and Medicine, 147, 106028, 2022.

[11] Li, H., et al. “Improving Object Classification in Autonomous Driving with Squeeze - and - Excitation Enhanced YOLO Network.” Journal of Intelligent Transportation Systems, 27(1), pp.1-12, 2023.

[12] Zhao, X., et al. “Hybrid Attention Networks: Combining Squeeze and Excitation with Spatial Attention for Object Classification.” Pattern Recognition Letters, 181, pp. 235 – 242, 2024.

[13] Wu, Y., et al. “Generative Adversarial Networks with Squeeze - and - Excitation for Object Classification.” Neural Computing and Applications, 36(12), 14771-14784, 2024.

[14] LeCun, Y., Denker, J. S., & Solla, S. A. “Optimal Brain Damage.” In Advances in neural information processing systems, pp. 598 - 605, 1989.

[15] Han, S., Mao, H., & Dally, W. J. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.” The 43rd annual international symposium on computer architecture , pp. 473 - 484, 2015.

[16] Louizos, C., Welling, M., & Kingma, D. P. “Learning Sparse Neural Networks through L0 Regularization.” In Advances in neural information processing systems , pp. 2755 - 2765, 2017.

[17] Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. “Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning.” arXiv preprint arXiv:1611.06440, 2016.

[18] Guo, Y., Yao, A., Chen, M., & Liu, X. “Network Slimming: Learning Efficient Convolutional Networks through Network Slimming.” The IEEE conference on computer vision and pattern recognition, pp. 2755-2763, 2016.

[19] Zhou, H., et al. “Data - Dependent Sparse Structure Selection for Deep Neural Networks.” The IEEE/CVF International Conference on Computer Vision, pp. 8794 - 8803. 2023.

[20] Yang, J., et al. “Pruning for Mobile Devices: Balancing Model Size, Computation, and Power.” The 29th ACM international conference on multimedia, pp. 207-215, 2021.

[21] Sandler, M., A. Howard, M. Zhu, et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, June 18 - 23, 4510 – 4520, 2018.

[22] Ma, Ningning, Xiangyu Zhang, Hai - Tao Zheng, and Jian Sun. "ShuffleNet v2: Practical Guidelines for Efficient CNN Architecture Design." In Proceedings of the European Conference on Computer Vision (ECCV), 11218, 122-138, 2018.

[23] Tan, Mingxing, and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." In Proceedings of the 36th International Conference on Machine Learning, pp. 6105-6114. arXiv:1905.11946, 2019.