# Analysis and Implementation of Compression Techniques for Deep Neural Networks

**Nishtha Shah**
nsshah15@asu.edu
*Arizona State University*

**Aesha Shah**
ashah116@asu.edu
*Arizona State University*

**Priyal Padheriya**
ppadheri@asu.edu
*Arizona State University*

***Abstract*—Deep Neural Networks (DNNs) have achieved impressive results in a variety of applications, including image recognition, speech recognition, and natural language processing. However, these models are often large and computationally expensive, which can be problematic for deployment on resource-constrained devices such as mobile phones or embedded systems. While larger models may be necessary for optimal performance during training, compression techniques are needed to reduce the size and computational requirements of DNNs for deployment on low-power devices. In this context, model compression techniques for DNNs have gained increased attention in recent years, as there is a growing demand for deploying efficient and high-performance models on resource-constrained devices. In this project, we aim to present an analysis of compression techniques for convolutional DNNs to compare and contrast their performance against the baseline architectures. We quantify the performance of compression techniques based on the time taken to run inference, the memory footprint of the models and the prediction accuracy. We also present the results of our analysis on perturbed training data to measure the robustness of these techniques.**

***Index Terms*—Compression techniques, quantization, binarization, network pruning, GPUs, inference time**

## I. INTRODUCTION

Deep Neural Networks (DNNs) have shown remarkable performance in various tasks such as image recognition, speech recognition, and natural language processing. However, these models are often very large in size and require significant computational resources for both training and inference. During training, the primary goal is to extract as much structure from the given data as possible, and thus models can be large and computationally expensive. However, in real-world applications, such as mobile devices or embedded systems, model size and computational resources become of concern if the model is to be deployed for inference. Large models can take up a lot of memory and processing power, making them unsuitable for deployment in such scenarios. There is often a trade-off between model size, computational resources, and accuracy. This makes it challenging to deploy these models on low-power devices with limited computational resources and storage capacity.

Therefore, to make deep neural networks more practical for deployment on resource-constrained devices, model compression techniques are needed to reduce their size and computational requirements without compromising their accuracy. Model compression techniques in deep neural networks have become increasingly important due to the growing demand for deploying efficient and high-performance models on resource-constrained devices such as mobile phones and embedded systems.

Model compression techniques aim to address the aforementioned challenges by reducing the size of the model and the computational cost required for inference without compromising its performance. This involves various techniques such as pruning, quantization, and knowledge distillation. These techniques are typically applied in a sequential manner, with each technique building on the previous one. By using these techniques, it is possible to achieve significant compression in DNNs without sacrificing accuracy. This makes it possible to deploy DNNs on resource-constrained devices such as mobile phones, embedded systems, and IoT devices.

## II. RELATED WORK

### A. Network Pruning

Along with the advancement in high-capability DNN models, a significant amount of research has been conducted on model compression techniques since they began to require larger sizes and computational resources. One such technique is Network Pruning, which involves reducing the size of a deep learning model by eliminating components such as neurons, connections, channels, filters, etc. that are deemed less impactful, resulting in a lightweight model [1].

Early works in the 1990s performed pruning using a second-order Taylor approximation of the increase in the loss function of the network when weight is set to zero [2]. In Optimal Brain Damage, the saliency for each parameter was computed using a diagonal Hessian approximation, the low-saliency parameters were pruned from the network, and the network was retrained [3]. Later in the 2000s, magnitude-based weight pruning methods have become popular techniques for network pruning [4], [5], [6]. Magnitude-based weight pruning techniques are computationally efficient, scaling to large networks and datasets. An automated gradual pruning algorithm proposed in [7] prunes the smallest magnitude weights to achieve a preset level of network sparsity. In contrast with the works listed above, it focuses on comparing the model accuracy and size tradeoff of large-sparse versus small-dense models.

Recently, several papers have been published demonstrating better pruning criteria and techniques. For example, [8], proposed a new method for estimating the contribution of a neuron using the Taylor expansion applied on a squared change in loss induced by removing a chosen neuron and demonstrated that

even the first-order approximation shows significant agreement with true importance, and outperforms prior work on a range of deep networks. While, [9] proposed a new notion of sparsity for vectors named PQ Index (PQI), which follows the principles a sparsity measure should obey developing a new perspective on the compressibility of neural networks by measuring the sparsity of pruned models and postulated a hypothesis on the relationship between the sparsity and compressibility of neural networks. On the other hand, [10] shows the Influence of Pruning on the Explainability of CNNs. This expands the scope of pruning beyond compression.

### *B. Quantization*

Quantization has started becoming an important area of research with the increase in demand for deploying machine learning models on devices with limited computational resources. In this section, we will be discussing ways in which we can improve the efficiency of neural networks by quantizing them and reducing their memory footprint. Quantization involves reducing the precision of the weights and activations in a network, typically from 32 bits to 8 bits or fewer, allowing for faster inference on devices with limited computational resources, such as mobile phones and embedded systems.

Iandola et al. [11] propose a method for quantizing deep convolutional networks using integer arithmetic, which can be more efficient than floating-point arithmetic on some hardware. Courbariaux et al. [12] introduce a binary quantization method that reduces weights to -1 or 1, greatly reducing the memory footprint and computational cost of inference. Nagel et al. [13] propose a technique for training quantized neural networks, which involves adding noise to the weights and activations to improve accuracy.

Suvorov et al. [14] investigate mixed-precision quantization, which involves using higher precision for some weights and activations and lower precision for others. They demonstrate that this approach can improve the accuracy of quantized networks. Alma et al. [16] provide a comprehensive survey of post-training quantization methods, which involve quantizing a pre-trained network rather than training it from scratch.

Apart from this, we also learn how different techniques can be used to reduce the size of neural networks. Some of the recent papers provide a comparison and analysis of these techniques and propose ways for enhanced hybrid techniques. For example, Danilevsky et al. [15] study the effects of weight pruning and quantization on the size and accuracy of the BERT language model, concluding that combining both techniques can further improve the efficiency of the network. On the other hand, Luo et al. [17] propose a binary convolutional neural network (BCNN) method for achieving accurate binary convolutional neural networks, where both weights and activations are binarized.

### *C. Knowledge Distillation*

In deep neural networks, the number of parameters can be very high, making the models computationally expensive and resource-intensive. Knowledge distillation allows for the transfer of knowledge from a larger, more complex model (teacher network) to a smaller, simpler model (student network). [18] This compression of the model size can lead to faster training times, reduced memory requirements, and improved computational efficiency.

One of the proposed techniques for knowledge distillation in deep neural networks called "FitNets" [9] argues that while the traditional knowledge distillation methods transfer knowledge from the final layer of the teacher to the student network, this approach does not make use of the intermediate representations learned by the teacher network. The FitNets approach instead focuses on transferring this intermediate knowledge to the student network. We can also distill knowledge from multiple "noisy" teacher models instead of a single teacher model by introducing a regularization term [20] that can lead to better compression and generalization performance. Or even introduce a "teacher assistant network" [21] to improve the process by providing additional guidance about the teacher's internal representations.

There are also methods that use only the pre-trained teacher model's parameters and do not require any labeled data for the student model using a method known as "data-free distillation" [22]. This method enables knowledge transfer to low-resource settings where labeled data is scarce or expensive to obtain. Another technique called "Variational Information Distillation" (VID) [23] introduces a variational distribution over the intermediate representations of the teacher network with the aim of better capturing this correlation. The VID method also provides a natural way to perform model compression by eliminating the need for the teacher network during inference, which can significantly reduce computational requirements.

There is more recent work providing an extensive review of knowledge distillation and student-teacher learning techniques for visual intelligence [24] and computer vision tasks, including image classification, object detection, and semantic segmentation, and discusses the challenges and limitations of existing methods and proposes new research directions for further advancements in this field. Overall, we gain valuable insights and a comprehensive understanding of knowledge distillation and student-teacher learning in the context of visual intelligence.

## III. EXPERIMENTAL SETUP

All the experiments were performed on the Google Colab GPU. The exact specifications of the GPU are provided in Table I.

| S.No. | Specification | Detail |
|---|---|---|
| 1. | GPU | Nvidia K80/T4 |
| 2. | Memory | 12 GB |
| 3. | Memory Clock | 0.82 GHz |
| 4. | Performance | 4.1 TFLOPs |
| 5. | CPU Cores | 2 |
| 6. | Disk Space | 358 GB |

TABLE I. Google Colab GPU Specs

## IV. BASELINE SETUP

As discussed in the above sections, we will implement, analyze and compare 3 different DNN model compression techniques name Network Pruning, Quantization, and Knowledge Distillation, and evaluate their performance in different scenarios.

### A. Dataset

The baseline dataset selected for comparing the performance of these 3 techniques is MNIST [25]. MNIST dataset consists of 28x28 grayscale images, where the training data is comprised of 60,000 images and test data of 10,000 images. Fig. 1 shows an example of an image of MNIST dataset.

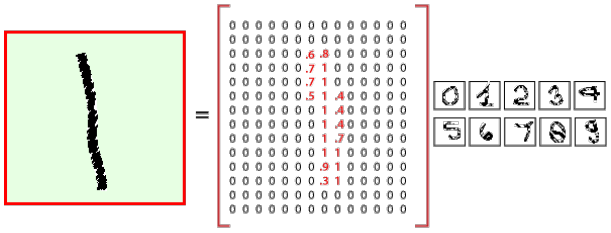

Fig. 1: MNIST dataset sample image

### B. Model

In order to compare the performance of 3 different techniques, we will select a baseline model and then compress it using the selected techniques for analyzing their performances. Usually, we select any of the standard models like VGG-16, ResNets, EffiecientNets, MobileNet, etc. as a baseline. For this project, we will use LeNet-5 [26]. Fig. 2 shows the layer-wise architecture of the selected baseline model LeNet-5.

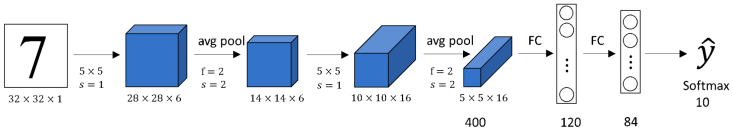


Fig. 2: LeNet-5 model architecture

Also, we fixed the hyper-parameters for a more fair comparison. The selected hyper-parameters are as follows:

1) Framework: TensorFlow 2.x
2) Optimizer: Adam
3) Loss: Sparse Categorical Cross-entropy
4) Learning Rate: default = 0.001
5) Epochs: 10
6) Batch-size: 256
7) Validation Split: 0.1(%)

### C. Evaluation Metrics

For evaluating the performance of the mentioned compression techniques, we will be using the prediction accuracy of the models on the MNIST dataset for comparison. The compression techniques will be compared on two metrics:

1) Model size
2) Inference time

This will give us a quantitative measurement of how efficient the compressed networks are compared to the non-compressed standard variants.

## V. TECHNICAL APPROACH

### A. Network Pruning

Network pruning is one of the most popular techniques for compressing DNNs. As the same suggests, It removes i.e. prunes redundant or unnecessary parameters to effectively reduce the computational cost and memory requirements of deep neural networks and most importantly, while maintaining their accuracy.

The basic concept of pruning is to identify the parameters in a neural network that contribute very little to the overall performance of the network and then remove those parameters. Pruning converts a large model to a storage-friend one ensuring minimum loss of accuracy. It works on DNNs' Large-sparse v/s Small-Dense trade-off. These parameters can be any part of the DNNs such as individual neurons, connections, layers, filters, channels, etc. They are explained as follows:

1) Weight Pruning - In this technique, as shown in Fig. 3 the weights of the model are pruned so as to decrease the size of the model and it is done in such a way that it does not affect the model performance drastically.
2) Filter Pruning - In filter pruning the filters in the deep learning model are decreased so as to decrease the computational and memory requirements of the model.
3) Layer Pruning - In layer pruning, some of the less important layers of the model are removed (or pruned) to decrease the model size and computation requirements.

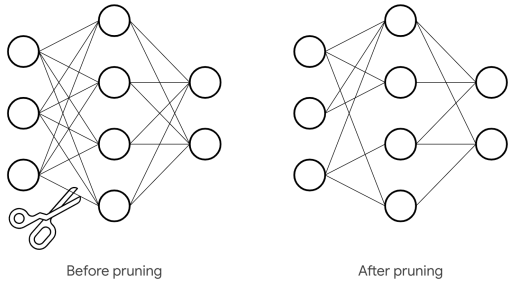


Fig. 3: Network Weight Pruning working example

For this project, we have selected Network Weight Pruning. Weight Pruning uses masking to prune unnecessary weights. The mask comprises set (1) and off-set bits (0) that decide whetheher to keep the weight or to remove it as it gets multiplied by a 0. Fig. 4 shows how the weights are pruned using masking.

### B. Quantization

Quantization is another widely used model compression technique. It is useful for reducing the precision of weights

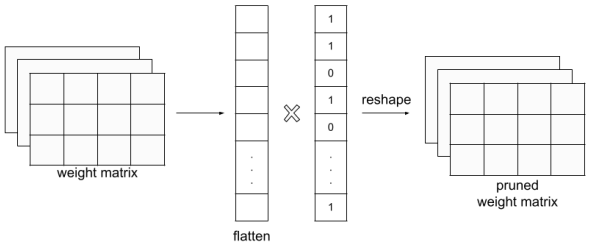


Fig. 4: Weight Matrix getting pruned using Masking

and activations of neural networks. This is used in order to reduce the memory requirements and computational complexity, making it easier to deploy them on resource-constrained devices like mobile phones. The main idea behind the concept is that if we convert floating point numbers of the weights and inputs into integers, we are going to consume significantly less memory and the speed of network calculations will increase drastically. The below Fig. 5 explains how the quantization process works.

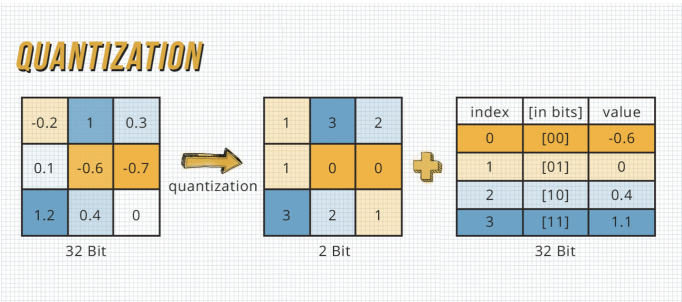


Fig. 5: How a model is quantized

Quantization can be challenging as it can introduce errors, making it necessary to fine-tune the network weights or adjust the quantization parameters. Quantization can reduce accuracy, but fine-tuning the network can minimize this loss. However, different layers may require different precision levels, making it challenging to find an optimal quantization scheme that balances accuracy and complexity. Despite these challenges, quantization has proven effective in reducing the memory and computational requirements of neural networks without significantly sacrificing accuracy. Below Fig. 6 shows the difference between a compressed and a quantized model.

Quantization can be performed during training or after training. By lowering the values of weights and activations to a lower bit width. It can be achieved by using various methods such as uniform quantization, logarithmic quantization, and hybrid quantization. Depending on the degree of compression needed, quantization from any of the three types below can be used:

1) Quantization Aware Training - This is used in preparation of models by on low-precision hardware during the training process. It involves simulating the effects of quantization. It helps to improve the efficiency and speed of inference while maintaining a high level of accuracy. The below Fig. 7 explains what happens in the back when a model is compressed using quantized aware training.

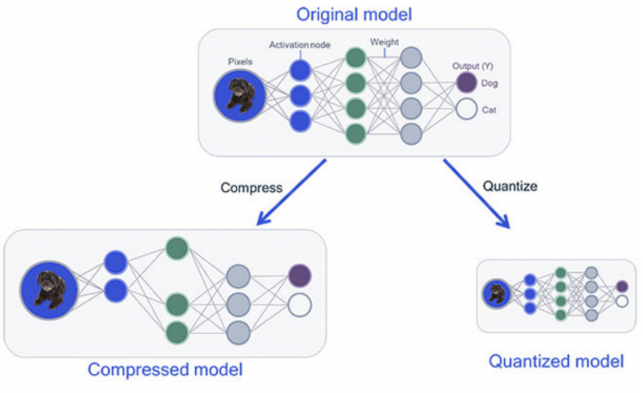


Fig. 6: Compressed model v/s Quantized model

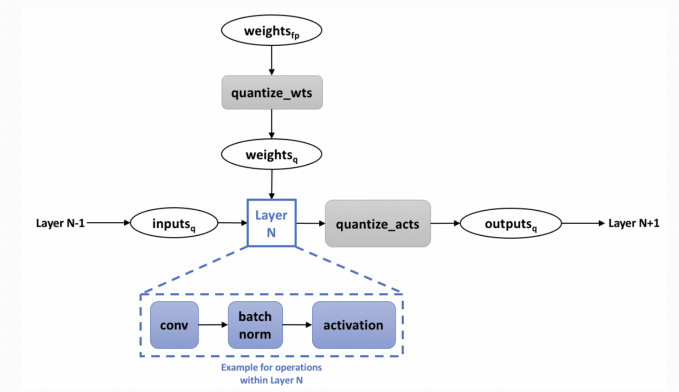


Fig. 7: Compressed model v/s Quantized model

2) Post Training quantization - This type of quantization is applied to the model after completion of the training phase. This is a quicker approach but accuracy might get affected.
   a) Static Quantization - This is used for deployment deep neural networks on devices with limited resources. The quantization range is based on the representative data set that is to be selected before deployment. With the help of this, the minimum and maximum values for each layer in the neural network is calculated.
   b) Dynamic Quantization - This is also a model compression technique similar to the above one but this technique allows each batch of the model's input to use a different quantization range based on distribution of the data.

For the purpose of our analysis, we tried compressing the LeNet-5 baseline model by use of both quantized aware training and post training quantization. We do quantize aware training by using quantize_model() function. Then we again fine tune the model by training it but this time by lesser

number of epochs and again convert it using TFLlite model. It is observed that even though quantization aware trained model produces slightly better results than statically quantized model, it takes even more time to train this model since it takes into account the injection of quantized blocks.

### C. Knowledge Distillation

The need for neural network compression arises from the fact that trained models are often larger and slower than what would be ideal for inference. To address this problem, knowledge distillation was proposed. Knowledge distillation is a compression technique for deep neural networks (DNNs) that involves training a smaller, simpler "student" model to mimic the behavior of a larger, more complex "teacher" model. Furthermore, deploying deep learning models on edge devices such as mobile phones and embedded systems necessitates the use of lighter models that have lower memory and computational requirements. Smaller models, however, are prone to underfit large datasets, while bigger models can be too resource-intensive and have higher inference times, making them unsuitable for deployment on edge devices with hardware restrictions.

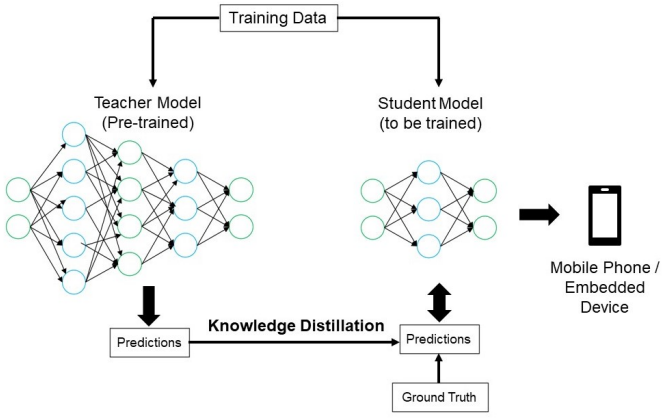


Fig. 8: Knowledge Distillation

The main idea behind knowledge distillation is that the teacher model has learned to capture important features and relationships in the training data, and the student model can benefit from this knowledge by learning to mimic the teacher's behavior. The student model learns to capture the same features and relationships as the teacher model by using the soft targets i.e. probability distributions produced by the teacher model for each input example in addition to the usual hard targets (i.e. true labels). Knowledge distillation can significantly reduce the number of parameters required to train a DNN, leading to faster inference and lower memory requirements.

Temperature is an important hyper-parameter we tune in knowledge distillation. It controls the smoothness of the distribution of probabilities generated by the teacher model. A higher temperature leads to a smoother distribution and can result in a more generalized model, while a lower temperature puts more emphasis on ground truth labels. So, we make observations for multiple values of temperature and its impact on the performance and evaluation metrics. Thus, the choice of temperature is a crucial factor in achieving optimal results with knowledge distillation.

We performed knowledge distillation by training the teacher model having the same architecture as the baseline model i.e. LeNet-5 model. Next, we trained the student model including the predictions obtained from the teacher model. The student has the architecture as follows:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_7 (Conv2D) | (None, 26, 26, 4) | 40 |
| max_pooling2d_7 (MaxPooling2D) | (None, 13, 13, 4) | 0 |
| flatten_7 (Flatten) | (None, 676) | 0 |
| dense_13 (Dense) | (None, 16) | 10832 |
| dense_14 (Dense) | (None, 10) | 170 |

Total params: 11,042
Trainable params: 11,042
Non-trainable params: 0

Fig. 9: Student Model Architecture

However, the student model may not be able to capture all the necessary information and that might lead to a decreased accuracy. Therefore, knowledge distillation is often used in conjunction with other compression techniques, such as pruning and quantization, to achieve optimal compression while maintaining accuracy.

## VI. RESULTS AND DISCUSSION

### A. Network Weight Pruning

Network weights can be pruned in 2 fashions, as mentioned below:

1) Constant
2) Scheduled

Here, Constant Pruning means out of the total number of weights in each layer a fixed percentage of weights are pruned. While in Scheduled Pruning, the percentage of weights being pruned is slowly increased if the accuracy is not being affected.

On pruning the trained baseline LeNet-5 model, it was found that this Network Weight Pruning technique has high model compressibility. In addition, the improvement over the inference latency was very significant. These betterments also ensured the minimum loss of accuracy. All these can be concluded from the results shown in Table II.

From Table II, it can be seen that when the baseline model is pruned using Scheduled Pruning starting from 50% to 80%, the loss in accuracy is minimum (only 0.015%). But on the other hand, the space occupied by the model is reduced by approximately 3 times and the latency for inference time is reduced almost by 2 times.

| Model Sparsity | Test Accuracy (%) | Model Size (Mb) | Inference Time (sec) |
|---|---|---|---|
| **Baseline** | 98.85 | 164.11 | 4.034 |
| **Constant 50%** | 98.80 | 101.97 | 2.800 |
| **Constant 80%** | 98.10 | 55.28 | 2.045 |
| **Scheduled 50-80%, Final 80%** | 98.60 | 55.28 | 2.643 |

TABLE II. Network Weight Pruning Results

### *B. Quantization*

As mentioned above we performed quantization on the model in two different ways - performing training while simulating the effects of quantization and by compressing the model with the help of quantization after the training phase is completed.

The Table III provides a comparison of 3 different model. The compresses models differs in test accuracy, model size, and inference time. The baseline model achieved a high test accuracy of 98.85% but had a relatively large model size of 164.11 MB and a slow inference time of 4.034 seconds. The quantized model, which underwent post-training quantization, had a lower test accuracy of 97.25%, but with a significantly reduced model size of 44.23 MB (approximately 4%) and a faster inference time of 2.03 seconds (approximately 2.5 times faster). The third model, which underwent quantized aware training, achieved a test accuracy of 98.32%, with a smaller model size of 41.17 MB and an even faster inference time of 1.61 seconds. Both quantized models were able to significantly reduce the model size and improve the inference time, while still maintaining a relatively high level of test accuracy compared to the baseline model. It is worth noting that combining quantization to a pruned model would produce better results.

### *C. Knowledge Distillation*

We employ knowledge distillation to train the student model by utilizing the teacher model's predictions and the ground truth. We vary the temperature hyper-parameter to compare the performances of the student model, considering the changes in accuracy, model size, and inference time for each model as seen in Table IV.

Our findings indicate that increasing the temperature produces a more generalized student model, whereas reducing the temperature enhances its ability to learn precise decision boundaries. We determine an optimal temperature value of around 6 or 7 through experimentation. Additionally, we note a significant reduction in model size (about 3.5 times lower than the original baseline model) and inference time (about half the baseline model's time) for the distilled model with a slightly lower but still comparable accuracy to the baseline. Combining knowledge distillation with other model compression techniques can improve performance while minimizing the trade-off between evaluation metrics.

### *D. Comparison and Discussion*

Network weight pruning, quantization, and knowledge distillation are three model compression strategies that show potential for lowering model size and speeding up inference while preserving high test accuracy.

Constant pruning and scheduled pruning were compared with respect to network weight pruning, and it was discovered that scheduled pruning with a gradually increasing percentage of weights being pruned resulted in the least accuracy loss while achieving appreciable reductions in model size and inference time. While reducing model size and inference time by up to 4 times and 2.5 times, respectively, respectively, quantization, both post-training and quantized aware training, also produced excellent results. In addition, a relatively high level of test accuracy in comparison to the baseline model was maintained. The ground truth and forecasts from a bigger instructor model were used to train a smaller student model using knowledge distillation. A significant decrease in model size and inference time was made while still retaining accuracy that was comparable to the baseline model by determining the optimal temperature value to be somewhere between 6 and 7.

Overall, every method has particular benefits and drawbacks. Combining these methods might improve performance even more while reducing the trade-off between evaluation metrics. The table V shows the comparison results for the three techniques.

## VII. CONCLUSION

In conclusion, the size and computational requirements of Deep Neural Networks (DNNs) have made it challenging to deploy them on low-power devices. However, model compression techniques such as pruning, quantization, and knowledge distillation have emerged as effective solutions to these challenges. By applying these techniques, it is possible to achieve significant compression in DNNs without compromising accuracy, making it feasible to deploy them on resource-constrained devices.

Our analysis of these compression techniques for CNNs underscores the importance of considering inference time, memory footprint, and prediction accuracy when assessing their performance. Constant and scheduled pruning approaches both have their advantages, with scheduled pruning showing better results when starting from 50% to 80% pruning. Quantization also yielded promising results, with quantized aware training performing better than post-training quantization. Knowledge distillation can further enhance model compression and generalization, with an optimal temperature value. Combining these techniques can result in improved performance while minimizing the trade-off between evaluation metrics. These techniques are crucial in developing efficient and faster models that require less storage, making them ideal for deployment in resource-constrained environments.

| Model Type | Test Accuracy (%) | Model Size (Mb) | Inference Time (sec) |
|---|---|---|---|
| **Baseline** | 98.85 | 164.11 | 4.034 |
| **Quantized Model (Post Training)** | 97.25 | 44.23 | 2.03 |
| **Quantized Model(Quantized Aware Training)** | 98.32 | 41.17 | 1.61 |

TABLE III. Quantization Results

| Model Type | Test Accuracy (%) | Model Size (Mb) | Inference Time (sec) |
|---|---|---|---|
| **Baseline** | 98.85 | 164.11 | 4.034 |
| **T = 1** | 97.69 | 43.535 | 1.328 |
| **T = 3** | 97.36 | 43.521 | 0.949 |
| **T = 8** | 95.83 | 43.594 | 1.346 |

TABLE IV. Knowledge Distillation Results

| Compression Technique | Test Accuracy (%) | Model Size (Mb) | Inference Time (sec) |
|---|---|---|---|
| **None (Baseline)** | 98.85 | 164.11 | 4.034 |
| **Pruning(Schld.)** | 98.60 | 55.28 | 2.643 |
| **Quantization** | 98.32 | 41.17 | 1.61 |
| **Knowledge Distillation(T = 1)** | 97.69 | 43.535 | 1.328 |

TABLE V. Comparison Results

## VIII. FUTURE WORK

https://www.overleaf.com/project/645301b2b29ebaf15943b27f

Model compression techniques are crucial for deploying efficient and high-performance models on resource-constrained devices such as mobile phones and embedded systems. As the demand for such devices continues to grow, there is a need for further research in this area. Here are some potential future works that could be explored:

1) Incorporate these techniques using other datasets such as CIFAR-10: The efficacy of model compression techniques may vary depending on the dataset used for training. Future studies could explore the effectiveness of these techniques on other popular datasets such as CIFAR-10.
2) Use a bigger model like ResNet-50 or BERT (model with more parameters) to test the results' scale: The scalability of model compression techniques is another area that could be explored. While the current study used LeNet-5 as the teacher model, future works could use larger models such as ResNet-50 or BERT to test the effectiveness of these techniques on models with more parameters.
3) Combining more than one techniques to obtain more efficient results: Model compression techniques can be used in combination with other methods to achieve even more efficient results. For example, techniques such as pruning, quantization, and knowledge distillation can be combined to achieve better results in terms of model size and computational requirements.

Thus, the field of model compression techniques is an active area of research, and there is scope for further exploration to achieve optimal performance on resource-constrained devices.

## IX. INDIVIDUAL CONTRIBUTION

1) Nishtha Shah - Network Pruning
2) Aesha Shah - Knowledge Distillation
3) Priyal Padheriya - Quantization

## REFERENCES


[1] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" arXiv preprint arXiv:2003.03033, 2020.

[2] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In D. S. Touretzky, editor, Advances in Neural Information Processing Systems 2, pages 598–605. Morgan-Kaufmann, 1990.

[3] Hassibi, D. G. Stork, and G. J. Wolff. Optimal brain surgeon and general network pruning. In IEEE International Conference on Neural Networks, pages 293–299 vol.1, 1993. doi: 10.1109/ICNN.1993.298572.

[4] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. CoRR, abs/1510.00149, 2015a.

[5] Abigail See, Minh-Thang Luong, and Christopher D. Manning. Compression of neural machine translation models via pruning. In CoNLL, pages 291–301. ACL, 2016.

[6] Sharan Narang, Gregory F. Diamos, Shubho Sengupta, and Erich Elsen. Exploring sparsity in recurrent neural networks. CoRR, abs/1704.05119, 2017.

[7] Zhu, Michael, and Gupta, Suyog. "To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression." ArXiv, 2017.

[8] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, Jan Kautz; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 11264-11272.

[9] Diao, Enmao, Wang, Ganghua, Zhan, Jiawei, Yang, Yuhong, Ding, Jie, and Vahid Tarokh. "Pruning Deep Neural Networks from a Sparsity Perspective." ArXiv, (2023).

[10] Weber, David, Merkle, Florian, Schöttle, Pascal, and Stephan Schlögl. "Less is More: The Influence of Pruning on the Explainability of CNNs." ArXiv, (2023).

[11] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1609.07061

[12] Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Advances in neural information processing systems (pp. 4790-4798)

[13] Nagel, M., Weber, M., and Wohlmayr, M. (2019). Quantized neural networks: Training neural networks with low precision weights and activations. In 2019 IEEE International Conference on Image Processing (ICIP) (pp. 2050-2054). IEEE.

[14] Suvorov, D., Iashin, V., and Vetrov, D. (2020). Mixed-precision quantization for end-to-end training of neural networks. In Advances in Neural Information Processing Systems (pp. 15027-15036).

[15] Danilevsky, M., Karpov, A., Mashikhin, A., and Torgashev, D. (2021). Compressing BERT: Studying the effects of weight pruning and quantization on the model size and accuracy. arXiv preprint arXiv:2102.07973.

[16] Alam, M., Najeeb, S., and Zhang, M. (2021). Post-training quantization for deep neural networks: A survey. arXiv preprint arXiv:2107.04125.

[17] Luo, W., Wu, C., Qian, C., and He, X. (2021). Towards accurate binary convolutional neural network. In Proceedings of the 29th ACM International Conference on Multimedia (pp. 540-548).
[18] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. arXiv preprint arXiv:1503.02531.
[19] Romero, A., Ballas, N., Kahou, S. E., and Bengio, Y. (2015). Learning from teacher networks. arXiv preprint arXiv:1503.00075.
[20] Kim, Y., Kim, C., Kim, S., and Kim, J. (2016). Deep model compression: Distilling knowledge from noisy teachers. arXiv preprint arXiv:1610.09650.
[21] Zhang, H., Cui, Y., Neumann, G., and Chen, W. (2018). Improved knowledge distillation via teacher assistant. arXiv preprint arXiv:1802.00179.
[22] Li, J., Li, W., and Talwalkar, A. (2018). Data-free learning of student networks. arXiv preprint arXiv:1808.03856.
[23] Aghdam, S. R., Golkari, S., and Huang, Y. (2018). Variational information distillation for knowledge transfer. arXiv preprint arXiv:1806.05594.
[24] Wang, L., and Yoon, K. J. (2020). Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1-1.
[25] Chen, F., Chen, N., Mao, H., and Hu, H. (2018). Assessing four Neural Networks on Handwritten Digit Recognition Dataset (MNIST). ArXiv. /abs/1811.08278
[26] Berngardt, O. I. (2023). Improving Classification Neural Networks by using Absolute activation function (MNIST/LeNET-5 example). ArXiv. /abs/2304.11758