Advanced ML PA3

Muhammad Huraira Anwar [1] **Adeen Ali Khan** [2]

1. Abstract

As deep learning models grow in complexity, their storage and computational costs become significant challenges, particularly for real-time and resource-constrained applications. In this work, we explore three fundamental model compression techniques: pruning, quantization, and knowledge distillation. We apply both unstructured and structured pruning to a VGG-11 model trained on the CIFAR-100 dataset, reducing model size while maintaining accuracy through selective removal of redundant weights and channels. Additionally, we investigate post-training quantization (PTQ) and quantization-aware training (QAT) across various bitwidths, from fp16 to int4, to evaluate the trade-off between precision and performance. We also study the effect of knowledge distillation, comparing logit matching, feature matching, and contrastive representation distillation to examine the transfer of robustness and generalization from a larger teacher model to a smaller student model. Our results demonstrate the effectiveness of each technique under different constraints, providing insights into their impact on model efficiency and real-world applications.

2. Introduction

Deep learning models have achieved significant success in various applications, but their growing size and computational demands pose challenges for deployment in resource-constrained environments. As models like VGG-11, ResNet, and others become more prevalent, efficient techniques to reduce their memory footprint and computational complexity are crucial, especially for real-time systems and edge devices. Model compression techniques such as pruning, quantization, and knowledge distillation offer viable solutions by reducing model size while retaining performance.

In this assignment, we explore these three core techniques in the context of the VGG-11 architecture on the CIFAR-100 dataset. Pruning involves removing redundant or less important parameters to shrink the model without compromising accuracy. Quantization reduces the precision of model weights and activations, allowing for faster inference and lower memory usage. Knowledge distillation compresses models by transferring knowledge from a larger teacher model to a smaller student model, maintaining performance through distillation of logits or internal feature representations. This work investigates how each of these techniques can be applied to deep models, providing insights into their trade-offs and benefits. We also compare their performance, efficiency, and impact on accuracy under different compression levels, paving the way for deploying more compact and efficient models in real-world applications.

3. Methodology

Task 1: Pruning

In this task, we applied pruning techniques to compress a VGG-11 model trained on the CIFAR-100 dataset. The objective was to reduce the model's size and complexity by removing less important weights while maintaining accuracy. Two pruning methods were explored: unstructured pruning and structured pruning.

Pruning reduces the number of parameters in a neural network by removing connections that contribute the least to performance. This reduces model size and computational demands, which is especially beneficial for deployment on resource-constrained devices.

1.1. Unstructed Pruning

Unstructured pruning focuses on removing individual weights based on their magnitudes. We used the L2-norm to identify and prune the smallest weights across each layer. Sensitivity analysis was conducted to determine how pruning affected each layer's accuracy. This allowed us to adjust sparsity ratios per layer, aiming for an overall 25% sparsity while maintaining acceptable performance.

1.2. Structured Pruning

Structured pruning is a coarse-grained approach that removes entire channels (filters) from convolutional layers, making the network more hardware-efficient. We pruned channels based on their L2-norm and again performed a sensitivity analysis to determine optimal pruning ratios. This led to a model with 75% size reduction while considering both accuracy and computational efficiency.

1.3. Comparing the two approaches

We utilized Grad-CAM visualizations to analyze feature localization in the original, unstructured-pruned, and structured-pruned models. Grad-CAM highlights the regions of the input image that contribute most to the model's predictions, allowing us to observe how pruning affects these focus areas. By applying Grad-CAM to each model, we were able to visually compare the changes in attention across different layers and pruning techniques.

For each pruning method, we conducted a qualitative comparison. In unstructured pruning, individual weights were removed, resulting in irregular matrices, which could affect hardware efficiency. In structured pruning, entire channels were pruned, leading to more hardware-friendly models, but requiring careful tuning to ensure the removal of entire filters did not overly degrade performance.

In addition, we measured inference time to compare the computational efficiency of the original and pruned models. This was done to evaluate the practical benefits of pruning in terms of reduced computation. Finally, we conducted an accuracy vs. target sparsity analysis to understand how varying levels of sparsity affected the overall performance of the models across different pruning strategies.

Task 2: Quantization

2.1. Comparing Post Training Quantization and Quantization Aware Training

We explored multiple PTQ techniques to evaluate the effects of reduced bit-width on model performance. Initially, we created deep copies of a baseline model (pretrained on CIFAR-100) to apply distinct quantization formats: FP16, BF16, INT8, and INT4. FP16 and BF16 quantization were implemented by converting the model's datatype directly to *torch.float16* and *torch.bfloat16*, respectively. For INT8 quantization, we used the *torchao* library's built-in dynamic quantization methods (*int8 weight only*), applying quantization across both weights and activations. Additionally, we implemented INT4 quantization by defining a custom function that scales and clamps each weight tensor to fit within the range [-8, 7], mimicking INT4 bit-width. For fair comparison, we only quantized weights of int8 and int4.

We also explored techniques in different precision formats for QAT, which was applied to each quantized variant by establishing training routines that incorporated lower-precision arithmetic, ensuring model resilience to the precision-induced discrepancies. Additionally, an custom INT8 and INT4 configs were defined to align with the *fbgemm* backend for optimized computations. For the FP16 and BF16 QAT models, both model parameters and input data were converted to the respective types, ensuring compatible computations during training. The INT8 and INT4 models required extra preparation, like custom configs. The models underwent QAT for **5** epochs with regular validation steps to track accuracy in lower-bit environments. Post-training, the INT8 and INT4 models were further converted to a quantized form for efficient deployment, and each model was evaluated on a test set to benchmark accuracy and assess the trade-offs between quantization level and performance.

To assess performance across quantized models, we evaluated, for each quantized variant (FP16, BF16, INT8, and INT4) on a standard test set. Each function iterates over test samples, applying quantized inference to determine the accuracy of each model. A baseline accuracy was also established for comparison , after finetuning the VGG11 model for **5** epochs.

2.2. Scaling Law Analysis

In this part, we use the results from our PTQ and QAT experiments,to perform a scaling law analysis in order to examine how the model performance changes with decreasing bit precision. In order to analyse their impacts at varying bit-widths. By analyzing the scaling behavior of quantization, we can better understand how aggressively we can reduce the precision without severely impacting model performance.

2.3. Comparing fp16 and bf16

In this part, we analyze the difference between the 2 floating-point formats: fp16 and bf16. Both formats use **16** bits for storage, however bf16 retains more exponent bits, leading to greater numerical stability during training, especially when gradients are small. Comparing the model performance when quantized to fp16 and bf16, it is noticeable that while both formats allow for more efficient training compared to full precision, bf16 generally results in more stable training due to its ability to represent a wider dynamic range of values. This stability is particularly important in deep networks where gradients can vary widely in scale. We discuss the reasons why bf16 offers more stable training than fp16, especially in the context of deep learning training, highlighting the difference in how exponent bits are handled in each format and why this matters for training stability. We also thoroughly discuss tradeoffs in terms of memory, compute, precision, accuracy as well as representation of very small or very large numbers.

Task 3: Knowledge Distillation

3.1. Derivation of Distillation Loss

$$L_{distillation} = -T^2 \sum_{i=1}^{c} p_i \log(q_i)$$

Where:

$$p_i = \frac{\exp(v_i/T)}{\sum_{j=1}^{c} \exp(v_j/T)}$$

$$q_i = \frac{\exp(z_i/T)}{\sum_{j=1}^{c} \exp(z_j/T)}$$

Key Assumptions:

1. Logits are zero-meaned: $\sum_{j=1}^{c} z_j = \sum_{j=1}^{c} v_j = 0$
2. N represents total number of logits which is equal to c which represents total number of classes
3. Temperature T is constant
4. Omitting T^2 constant from initial term without loss of generality

Derivation:

1) First, expand q_i in the loss function:

$$L_{dist} = -\sum_{i=1}^{c} p_i \log\left(\frac{\exp(z_i/T)}{\sum_{j=1}^{c} \exp(z_j/T)}\right)$$

2) Using log properties:

$$L_{dist} = -\sum_{i=1}^{c} p_i \left[\log(\exp(\frac{z_i}{T})) - \log(\sum_{j=1}^{c} \exp(\frac{z_j}{T}))\right]$$

$$= -\sum_{i=1}^{c} p_i \left[\frac{z_i}{T} - \log(\sum_{j=1}^{c} \exp(\frac{z_j}{T}))\right]$$

3) To find $\frac{\partial L_{dist}}{\partial z_i}$, we consider:

- Direct dependence on z_i in the i-th term
- Indirect dependence through the denominator sum in all terms

4) By dividing the summation into two terms and taking derivative with respect to z_i by applying chain rule:

$$\frac{\partial L_{dist}}{\partial z_i} = -\left[\frac{p_i}{T} - \sum_{k=1}^{c} p_k \frac{\exp(z_i/T)}{T\sum_{j=1}^{c} \exp(z_j/T)}\right]$$

5) Using definition of q_i and simplifying:

$$\frac{\partial L_{dist}}{\partial z_i} = -\frac{1}{T}[p_i - q_i]$$

$$= \frac{1}{T}[q_i - p_i]$$

6) Due to zero-meaned logits, using Taylor expansion of exponential around zero:

$$\exp(x) \approx 1 + x + \frac{x^2}{2}$$

7) Applying to probability definitions, if the temperature is high compared with the magnitude of the logits, we can approximate:

$$p_i \approx \frac{1 + v_i/T}{c + \sum_{j=1}^{c} v_j/T} = \frac{1 + v_i/T}{c} \text{ (using zero-mean)}$$

$$q_i \approx \frac{1 + z_i/T}{c}$$

8) Substituting back and simplifying:

$$\frac{\partial L_{dist}}{\partial z_i} \approx \frac{1}{T}\left[\frac{1 + z_i/T}{c} - \frac{1 + v_i/T}{c}\right]$$

$$\approx \frac{1}{cT^2}(z_i - v_i)$$

9) Since $N = c$ (number of logits),:

Therefore:

$$\frac{\partial L_{dist}}{\partial z_i} \approx \frac{z_i - v_i}{NT^2}$$

Key steps in this derivation:

1. Used logarithm properties to expand the loss
2. Applied chain rule for differentiation
3. Used zero-mean assumption to simplify sums
4. Applied Taylor expansion around zero for exponentials
5. Used the fact that N equals the number of classes c
6. Omitted the T^2 scaling from the initial term

The result shows that the gradient pushes each student logit z_i towards its corresponding teacher logit v_i, scaled by temperature and number of logits. So in the high temperature limit, distillation is equivalent to minimizing

$$\frac{(z_i - v_i)^2}{2}$$

provided the logits are zero-meaned separately for each transfer case. At lower temperatures, distillation pays much less attention to matching logits that are much more negative than the average.

3.2. Comparing Logit Matching Approaches

In this task, we compared 3 different approaches namely Logit Matching, Decoupling, and Label Smoothing. All of them were focused on matching the logits or the probabilities of the student and teacher. We finetuned the teacher (**Vgg16**) for **8** epochs on the CIFAR 100 trainset, and then, for fair comparison, we trained the distilled students and the independent student (**Vgg11** models) for **5** epochs on the trainset before evaluating. Learning rate was fixed at **1e-4**.

3.3. Comparing Performance of Logit Matching, Feature Matching, and Contrastive Representation Distillation

In this task, we compared 3 different approaches namely Logit Matching, Hints, and Contrastive Representation. Some of them were focused on matching the logits, while others were matching the intermediate layers(feature representations) of the student and teacher. We finetuned the teacher (**Vgg16**) for **8** epochs on the CIFAR 100 trainset, and then, for fair comparison, we trained the distilled students and the independent student (**Vgg11** models) for **5** epochs on the trainset before evaluating. Learning rate was fixed at **1e-4**.

3.4. Comparing Probability Distributions of Logit Matching, Feature Matching, and Contrastive Representation Distillation

In this task, we used the probability distributions of the whole testset as output from teacher, independent and distilled students after evaluation, to compute average discrepancy of the distributions.

3.5. Examining Localization Knowledge Transfer

In this task, we used GRAD-CAM to visualize which local parts of the image, the teacher focuses on, and is that knowledge being distilled to the students or not. To evaluate this, we ran over the CIFAR-100 testset and computed an average cosine similarity score between teacher and each of the students.

3.6. Checking for Color Invariance with CRD

In this task, we prepared a color jitter dataset , altering brightness, hue, saturation values, and applying that transform to train and test sets. The values remain the same for consistency and fair comparison. Then, we finetuned both the teachers (**Vgg16**) for **8** epochs (one on normal trainset, on on color jittered trainset),hence the other teacher is color invariant. For fair comparison, we trained both the distilled students (one from color invariant teacher, other from normal teacher) for **5** epochs on the trainset before evaluating all of them on the color jittered testset . Learning rate was fixed at **1e-4**.

3.7. Testing the Efficacy of a Larger Teacher Model

In this task, we finetuned both the teachers (**Vgg16** and **Vgg19**) for **8** epochs on the CIFAR 100 trainset, and then, for fair comparison, we trained both the distilled students (one from Vgg16, other from Vgg19) for **5** epochs on the trainset before evaluating. Learning rate was fixed at **1e-4**.

4. Results

Task 1: Pruning

Metric	**Original Model**	**Unstructured Pruning**	**Structured Pruning**
Inference Time (s)	24.10	23.87	15.02
Storage Size (MB)	492.77	492.77	361.63
Feature Localization	Detailed	Detailed	Slightly Blurred

Table 1. Comparison of Original, Unstructured Pruned, and Structured Pruned Models

In this analysis, we compare the original VGG-11 model, an unstructured pruned model, and a structured pruned model in terms of accuracy, inference time, storage efficiency, and visual interpretability via Grad-CAM. Table 1 summarizes the key metrics across these models.

The inference times indicate that the structured pruned model significantly reduces computation time (15.02 seconds) compared to both the original model (24.1 seconds) and the unstructured pruned model (23.87 seconds). This improvement is expected since structured pruning removes entire channels, simplifying operations and making the model more efficient for hardware acceleration. Unstructured pruning, by contrast, removes individual weights, maintaining model complexity and having less impact on computational time.

Storage efficiency follows a similar trend. The structured pruned model achieves a smaller footprint of 361.63 MB, while both the original and unstructured pruned models have a size of 492.77 MB. Structured pruning's focus on channel-level sparsity allows more effective compression, whereas unstructured pruning's selective removal of weights has limited impact on overall storage size.

The Grad-CAM visualizations in Figure 1 reveal feature localization differences. Both the original model and unstructured pruned model retain more detailed activation patterns around key regions, indicating minimal disruption to feature extraction. The structured pruned model, however, shows slightly broader or blurred activations, suggesting some loss

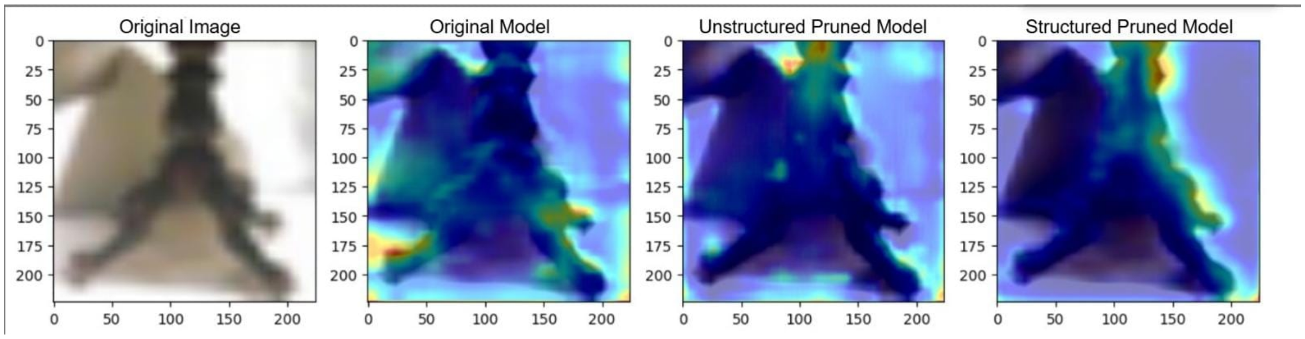

Figure 1. Grad-CAM comparisons for original model, structurally pruned, and unstructurally pruned models

in spatial detail, likely due to the removal of entire channels that capture specific features.

Overall, as summarized in Table 1, structured pruning demonstrates greater efficiency in computation and storage with a minor trade-off in spatial detail. This makes it suitable for hardware-accelerated deployment, while unstructured pruning may be preferred when retaining detailed feature extraction is essential.

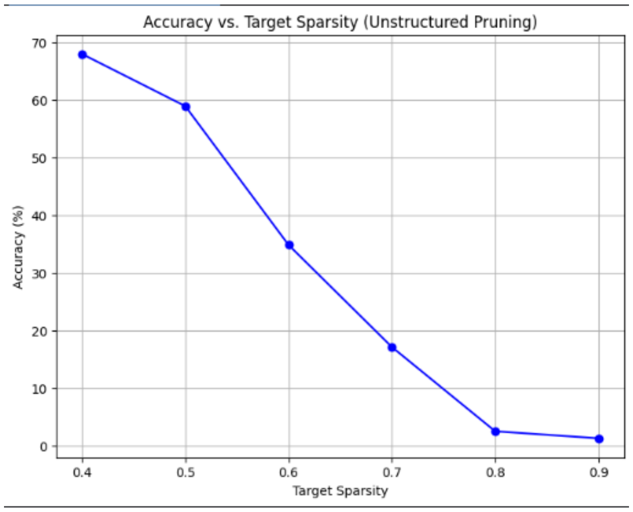

Figure 2. Sparsity vs Accuracy

We also tried to analyze the accuracy vs. sparsity relationship in unstructured pruning for the VGG-11 model. The results in Figure 2 illustrate that as sparsity increases, the model experiences a notable decline in accuracy, particularly beyond a certain threshold. Initially, at moderate sparsity levels, the model retains much of its predictive power, but as sparsity continues to rise, the accuracy sharply decreases. Beyond around 0.6 sparsity, this drop becomes more pronounced, as seen in the graph, reflecting the removal of critical weights and connections essential for maintaining performance. The corresponding increase in training loss at higher sparsity levels indicates that the model struggles to minimize error with fewer parameters, making fine-tuning less effective at these extremes. Overall, while unstructured pruning can be useful at lower sparsity levels for model compression with minimal accuracy loss, pushing sparsity too high leads to significant performance degradation, suggesting that this method is best applied conservatively.

Task2: Quantization

Comparing Post Training Quantization and Quantization Aware Training

Bit Width	PTQ %	QAT %
FP16	57.95	62.72
BF16	57.97	62.26
INT8	57.93	36.64
INT4	34.89	1.34

Table 2. Accuracies of the finetuned VGG11 model across varying bit-widths in Post Training Quantization and Quantization Aware Training

The results indicate that the FP16 and BF16 quantization methods yield similar accuracies, which are close to the baseline VGG11 accuracy of **58.53**%, and sometimes even outperform it. The INT8 quantization shows a significant drop in performance in QAT, even though it is at par with FP16 and BF16 in PTQ. On the other hand, INT4 in particular, is displaying a large drop in accuracy, from the baseline and other bit-widths in both PTQ and QAT.

Scaling Law Analysis

To analyze the scaling behavior of model performance with varying bit-widths in both Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT), we will examine the accuracy trends of the VGG11 model on CIFAR-100 under different quantization methods. The baseline accuracy of the VGG11 model at **5** epochs is **58.53**%, which will serve as a reference point for the analysis.

For PTQ, it is apparent that as the bit-width decreases from FP16 to INT4, the accuracy drops significantly, particularly at INT4, which shows a considerable performance degradation compared to the baseline. On the other hand, for QAT, it offers higher accuracy at FP16 and BF16 when compared to the baseline, indicating that QAT has a notable positive effect at these bit-widths. However, for lower bit-widths like INT8 and INT4, QAT performance drops sharply, with INT4 showing a drastic decline in accuracy. This highlights that while QAT can effectively maintain or even enhance accuracy at higher bit-widths, it struggles to preserve performance at very low bit-widths like INT4, similar to PTQ.

Task 3: Knowledge Distillation

Results of Distilled Students and Independent Student

As observable from the table below, all the distilled students perform about **2**% to **3**% better than the independent student and almost approach the accuracy of the teacher as well, given more epochs, they would converge even more closer to teacher accuracy. Except for the case of Hints model, it performs less than the independent student, and given more epochs, this gap increases. This trend suggests that most knowledge distillation techniques are effective in aligning student models with teacher performance, likely due to the enhanced learning from shared feature representations or softened probabilities. However, the Hint-based model's lower performance implies that feature matching may be less effective in transferring essential knowledge for this task, and extended training may further amplify this discrepancy rather than close it.

Contrastive student has least value of KL divergence of **1.55**, which highlights that it has the closest probability distributions as compared to the Teacher, even though it was matching the intermediate feature representations, and not the logits. While Hints, Independent student, and Logit Matching have about the same value of KL Divergence, around **1.8**. This suggests that the Contrastive method's focus on feature similarity effectively aligns the student's output distribution with the teacher's, even without direct logit matching.

Model	**Discrepancy**	**Accuracy %**
VGG 16 (Teacher)	-	63.36
VGG11 (Independent)	1.8038	58.40
Logit Matching	1.7842	61.30
Label Smoothing	-	60.28
Decoupling	-	60.11
Hint Based	1.8005	57.44
Contrastive	1.5504	61.85

Table 3. Average probability distribution KL divergences with teacher, and accuracies on CIFAR 100 testset, for teacher and all the students

Localization Knowledge Transfer Comparison

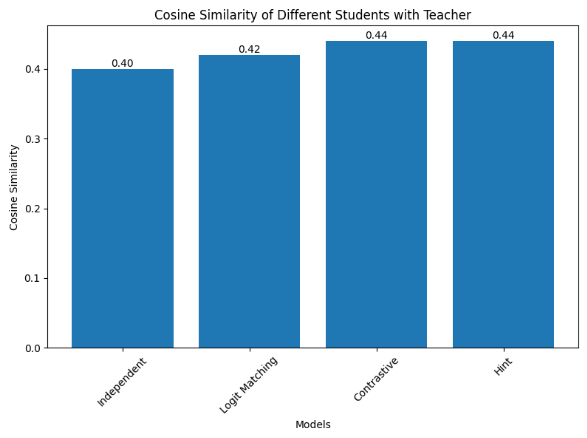

Figure 3. How similar the different students are with respect to teacher, in terms of the local knowledge distilled to them

As observable from the graph above, the distilled students have a greater cosine similarity score (**0.43** on average) with the teacher as compared to the independent student (**0.40**). The contrastive and hint models perform the best, since they both work on feature representations (intermediate layers), the transfer of localized knowledge is greater for those models.

Comparison with Color Invariant Teacher

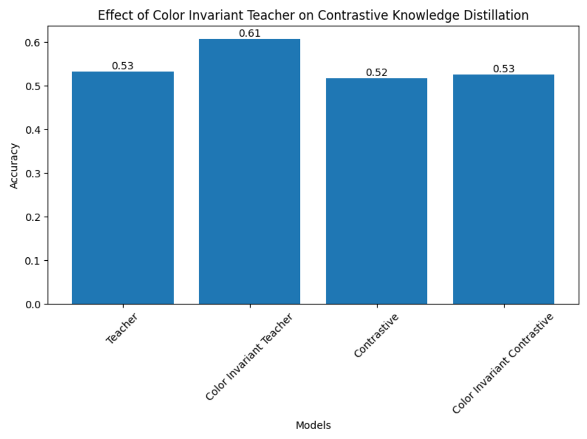

Figure 4. How the model accuracies of distilled students change across the different teachers (as they become color invariant)

As observable from the graph above, the normal contrastive student almost performs about **1**% worse than the color invariant contrastive student, who had the knowledge distilled by the color invariant teacher. And obviously, the color invariant teacher had a very stark difference (about **8**% more) than the normal teacher, when evaluated on the transformed dataset.

Comparison with Larger Teacher Model

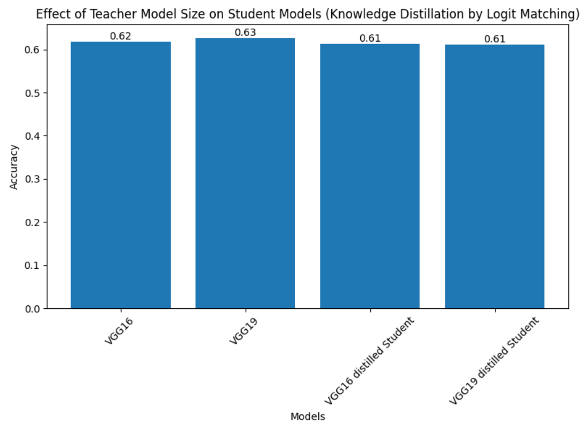

Figure 5. How the model accuracies change across the distilled students as the teacher size varies

As observable from the graph above, both the distilled students (by **Vgg16** and by **Vgg19**) perform about the same, in fact there is a slight dip (about **0.5**%) from VGG16 distilled student to VGG19 distilled student.

5. Discussion

Task 1: Pruning

Unstructured pruning and structured pruning offer complementary approaches to optimizing a VGG-11 model, each with unique strengths and trade-offs. In unstructured pruning, individual low-importance weights are selectively removed based on their L2 norms, achieving a target of 25% sparsity while preserving the model's accuracy. This method involves conducting a sensitivity analysis to determine each layer's tolerance to pruning, allowing for targeted sparsity ratios that maintain accuracy. Although unstructured pruning retains detailed feature localization and minimal performance degradation, its impact on storage and inference time is limited. This is because individual weight removal does not alter the overall structure or flow of computations within the network, so hardware might not fully leverage the sparsity unless specifically optimized for it.

Structured pruning, on the other hand, targets entire channels or filters, focusing on enhancing hardware efficiency and simplifying computations. By pruning channels rather than individual weights, structured pruning significantly reduces both inference time (from 24.1 to 15.02 seconds) and storage requirements (from 492.77 MB to 361.63 MB). This approach not only reduces the model size but also aligns well with hardware that can take advantage of channel-wise sparsity for faster computations. A sensitivity analysis is also used here to guide the channel selection process, identifying channels that can be pruned with minimal impact on model performance. While structured pruning leads to a compact, hardware-friendly model, it does result in slightly blurred feature localization in Grad-CAM visualizations. This is due to the loss of specific channels that capture particular details, impacting the model's ability to focus as precisely on certain features.

Together, unstructured and structured pruning provide flexible options for model optimization. Unstructured pruning offers fine-grained control, preserving accuracy and feature localization while making minimal changes to the network's structure. Structured pruning, however, provides significant gains in computational and storage efficiency, making it well-suited for deployment on hardware-accelerated platforms. Combining these strategies, or choosing between them based on deployment needs, allows for an adaptable approach to creating efficient yet high-performing models.

Task2: Quantization

Comparing Post Training Quantization and Quantization Aware Training

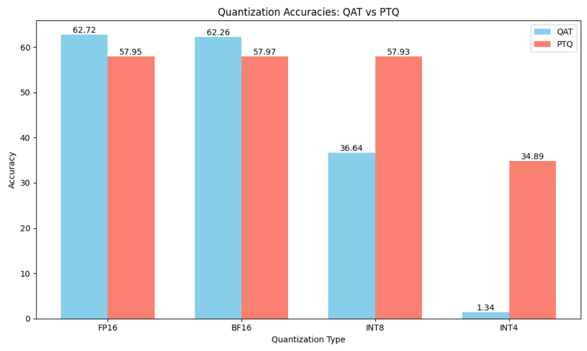

Figure 6. How the model performs across varying bit-widths in QAT and PTQ

The similar performance of FP16 and BF16 quantization methods, which are close to the baseline VGG11 accuracy, can be attributed to the fact that both of these formats retain a high level of precision compared to INT8 and INT4. FP16 and BF16 both use reduced precision in terms of bit width but still offer enough accuracy to preserve the model's performance, making them more suitable for training and inference tasks in many scenarios. These formats allow for faster computations and lower memory usage, without significantly impacting the accuracy, especially when trained with a moderate number of epochs, as observed in the results.

On the other hand, the INT8 quantization method shows a significant drop in performance during Quantization-Aware Training (QAT), despite being similar to FP16 and BF16 in Post-Training Quantization (PTQ). This discrepancy could be due to the aggressive quantization process in QAT, where model weights and activations are quantized to 8 bits, resulting in more information loss during training compared to FP16 and BF16, which use higher bit widths. As for INT4, its substantial drop in both PTQ and QAT is likely due to the extreme reduction in precision, leading to even greater information loss during training and inference. The use of just 4 bits severely limits the model's ability to capture fine-grained details in the data, significantly affecting its performance across all stages of quantization.

Scaling Law Analysis

When we analyze the results from Figure 6, several trends become clear:

Diminishing Returns at Lower Bit-widths: As the bit-width decreases from FP16 to INT4, we observe diminishing returns in terms of accuracy, especially at the INT4 level, which exhibits a steep drop in performance.

Impact of QAT vs PTQ: QAT offers significant improvements at higher bit-widths like FP16 and BF16. In contrast, at lower bit-widths like INT8 and INT4, QAT does not provide substantial benefits over PTQ. In fact, INT8 and INT4 accuracies for QAT are lower than those for PTQ, suggesting that QAT's effectiveness diminishes as precision decreases.

From this scaling law analysis, we can conclude that while QAT can offer substantial performance improvements at higher bit-widths, its advantages become less pronounced at lower bit-widths. The results suggest that PTQ might be a more effective method at lower bit-widths, while QAT is more beneficial when higher bit precision is maintained. This analysis helps in understanding the trade-offs between precision reduction and model performance, offering guidance for selecting the appropriate quantization strategy based on the desired balance between performance and computational efficiency.

Comparing fp16 and bf16

FP16 uses 16 bits to represent a floating-point number, with 1 sign bit, 5 exponent bits, and 10 mantissa bits. This allows for high precision (11 bits of precision) at the cost of a smaller dynamic range, also it requires more memory to store the data.

BF16, also, uses 16 bits to represent a binary floating-point number, with 1 sign bit, 8 exponent bits, and 7 mantissa bits. This allows for a smaller dynamic range and lower precision (8 bits of precision) compared to FP16. However, it requires less memory to store the data.

BF16 has a similar dynamic range to FP32 (since it is a truncated version of FP32) so underflows/overflows won't happen as often as they happen in FP16.

The largest number that can be represented by FP16 is **65535**, whereas a BF16 number can be as large as **3.39e+38**. Both methods can represent a number like **0.0001**.

However, let's consider a value like **1e-08 (0.00000001)**

FP16: 0.00000000000000
(Binary: 0 — 00000 — 0000000000, Hex: 0000)
BF16: 0.00000001001172
(Binary: 0 — 01100100 — 0101100, Hex: 322C)

Or a value like **100000.00001**

FP16: inf
(Binary: 0 — 11111 — 0000000000, Hex: 7C00)
BF16: 99840.00000000000000
(Binary: 0 — 10001111 — 1000011, Hex: 47C3)

In both the cases, FP16 fails and either makes the result 0 or infinity , as it is not able to represent it, however BF16 is able to represent it, even though, it is a little unprecise.

Even though BF16 and FP16 both require 2 bytes per parameter, the memory requirements of BF16 training are actually lower than in FP16 training. Because of the low precision of BF16, the gradients in BF16 training are directly stored in FP32 and consequently do not require an extra copy.

BF16's wider range allows training without scaling at all. Hence, it requires no loss scaling as opposed to FP16 and may lead to a better accuracy (due to loss of information in scaling).

Since BF16 is basically the FP32 number's first 2 bytes, converting between BF16 and FP32 is as easy as discarding the last 2 bytes of the FP32 number and copying the BF16 number into the first 2 bytes of the FP32 number's storage location. Converting between FP16 and FP32 instead requires more logic because of the differently sized mantissa.

Furthermore, in FP16 mixed precision training, gradients are constructed in FP16, then converted to FP32 for the parameter update. This step isn't needed in BF16 training because gradients are directly stored in FP32.

Lastly, the number of operations per iteration is reduced a bit because gradient/loss scaling isn't needed anymore with BF16.

FP16 is commonly used in deep learning training and inference, especially for tasks that require high precision in representing small fractional values within a limited range.

BF16 is becoming popular in hardware architectures designed for machine learning tasks that benefit from a wider range of representable values, even at the cost of some precision in the fractional part. It's particularly useful when dealing with large gradients or when numerical stability across a wide range is more important than precision of small values.

Task3: Knowledge Distillation

Comparison of Logit Matching, Decoupling, Label Smoothing

The results show that the distilled models generally achieve higher accuracy than the independent student model, with logit matching performing closest to the teacher. This reflects the effectiveness of knowledge distillation techniques in transferring crucial patterns from the teacher model to smaller, less complex models like VGG-11. Decoupling and label smoothing both contribute to effective knowledge transfer, yielding accuracy improvements compared to the standalone student. Logit matching approach of matching the softened labels of student and teacher likely helps it to capture localized and detailed information from the teacher more effectively, leading to a notable performance boost.

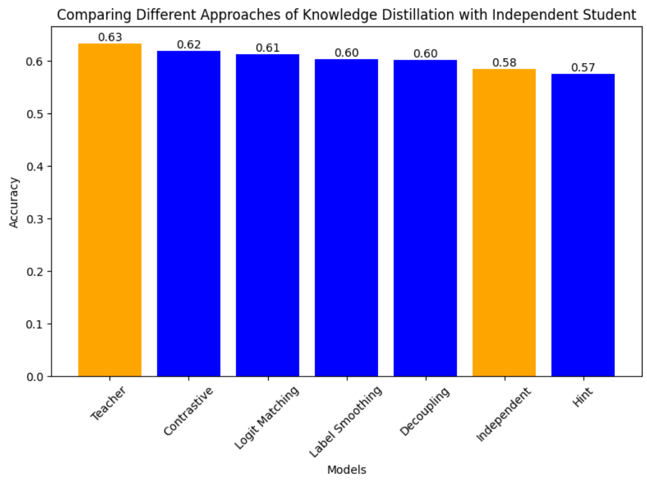

Figure 7. How the model accuracies change across the different knowledge distillation approaches

The relatively short training times (**8** epochs for the teacher and **5** for the students) may limit the overall accuracy gains possible. However, the knowledge distillation techniques still show a clear advantage, even with limited training, by leveraging the pretrained teacher's learned representations. The small gap between logit matching and the teacher highlights the potential of advanced distillation methods to create efficient yet accurate models suitable for resource-constrained environments.

Comparison of Logit Matching, Feature Matching, and Contrastive Representation Distillation

These results show that distillation methods significantly impact the student model's performance, allowing it to better approximate the teacher's accuracy. Contrastive Representation Distillation (CRD) achieves the highest accuracy among the student models, closely approaching the teacher's performance. This suggests that CRD's focus on aligning intermediate feature representations effectively transfers nuanced knowledge from the teacher to the student, helping the student capture critical patterns in the data. Logit Matching (LM) also yields strong results, indicating that transferring softened output probabilities is beneficial, as it provides the student with valuable information about inter-class relationships.

The independent student model, trained without any teacher guidance, has the second lowest accuracy among the distilled students, underscoring the advantage of knowledge distillation. Feature Matching (Hints) performs slightly worse than the independent student, possibly because aligning feature maps alone does not fully capture the decision-making structure of the teacher. The limited training (**5** epochs for students) may also restrict the potential gains from feature-based methods, which typically benefit from longer training to align features deeply. Overall, these findings highlight how knowledge distillation techniques, especially those targeting both output logits and representation alignment, effectively enhance student models in accuracy and robustness within resource constraints.

Comparison of Probability Distributions

KL divergence is a commonly used metric for comparing probability distributions, as it quantifies the divergence between the "true" distribution (teacher's output) and the student's learned distribution. In the context of knowledge distillation, it effectively measures how well the student model approximates the teacher's distribution, with lower values indicating closer alignment. By using KL divergence to compare the output distributions of the teacher and various student models, we can evaluate how effectively each knowledge distillation (KD) method transfers knowledge, as reflected by how similar the students' probability distributions are to that of the teacher. We also used a trick, to avoid *inf* KL divergence, which occurs due to 0 probability, by adding a very small *epsilon* value like **1e-10**, to all probabilites and normalizing them.

In analyzing the results, it is evident that the contrastive student achieves the lowest KL divergence at **1.55**, indicating the closest alignment with the teacher's distribution despite being trained to match intermediate feature representations rather than direct logits. This suggests that matching features can lead to robust, closely aligned output distributions, likely due to the comprehensive, fine-grained information

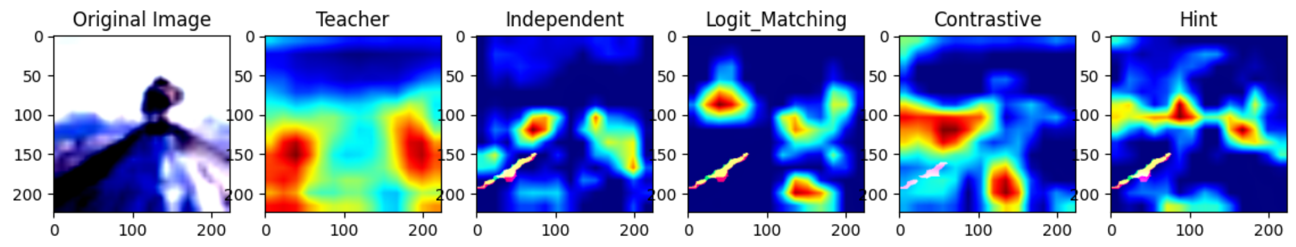

Figure 8. Visualising how the different models try to capture the same localized information as compared to the teacher

embedded in feature maps. The other KD methods, including Hints, Logit Matching, and the Independent student, yield similar KL divergence values around **1.8**, indicating that their distributions are less aligned with the teacher's compared to the contrastive approach. This outcome highlights that contrastive representation distillation may foster a closer mimicry of the teacher's predictions, potentially due to the richer, contrastive learning signal at the feature level.

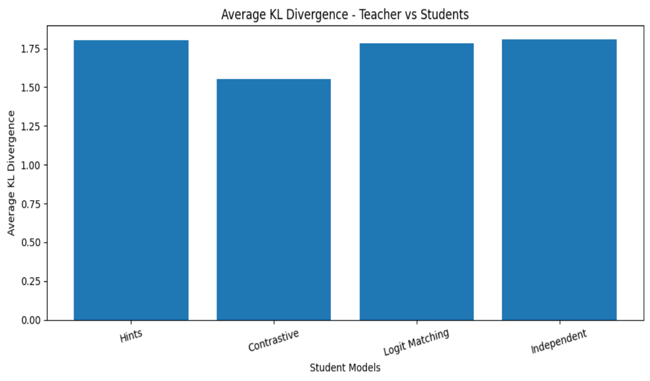

Figure 9. How similar the probability distributions of the students are with respect to the teacher

Localization Knowledge Transfer Comparison

The average cosine similarity scores reveal subtle yet important variations in how the student models align with the teacher's focus regions. Independent students achieve a **0.40** similarity with the teacher, indicating some degree of alignment but with noticeable divergence. Logit Matching improves slightly to **0.42**, showing that matching output distributions provides a slight increase in spatial attention alignment. The Hint-based (feature matching) and contrastive representation distillation methods, both at about **0.44** demonstrate a higher similarity, suggesting that these methods help the student models better capture the spatial information learned by the teacher.

The Grad-CAM visualizations in Figure 8 provide further insight into these results, illustrating how each model prioritizes different regions of the image. Independent students may focus on disparate or less relevant areas compared to the teacher, while models trained with Hint and contrastive distillation show more overlap in regions of interest, reflecting their higher cosine similarities. This alignment suggests that focusing on internal feature representations, as in contrastive and hint-based methods, effectively guides the student toward capturing finer spatial and semantic nuances of the teacher's attention, leading to stronger local alignment in focus areas. However, it is also notable, that there is a very low variance in the similarities of the students.

Comparison with Color Invariant Teacher

These results illustrate the effect of color invariance on model performance, particularly in challenging conditions introduced by the color-jittered dataset. The normal teacher model, which was trained on unaltered data, achieves an accuracy of **0.533** on the jittered set. In comparison, the color-invariant teacher—trained on color-jittered images—performs significantly better, reaching an accuracy of **0.613**. This gap highlights the benefits of color augmentation during training, as it allows the color-invariant teacher to generalize better to images with varied brightness, contrast, saturation, and hue. The images and their corresponding color-jittered transformations underscore this shift in the data distribution that the color-invariant teacher effectively learns to handle.

For the students distilled through contrastive representation distillation (CRD), we see that both the normal CRD student and the color-invariant CRD student fall behind the color-invariant teacher. The normal CRD student achieves an accuracy of **0.5175**, while the color-invariant CRD student reaches a slightly higher **0.525**. This marginal improvement suggests that although the color-invariant CRD student indirectly inherits some robustness to color changes (since its teacher is color-invariant), its reliance on standard, non-jittered training data limits its full potential in handling color variation. The small accuracy improvement shows that while some color-invariant features are transferred indirectly, the student would benefit more directly if the training data itself was also color-augmented. For student models, achieving robustness similar to the teacher might require a combination of both data-level augmentations and feature-level distillation from a color-invariant teacher.

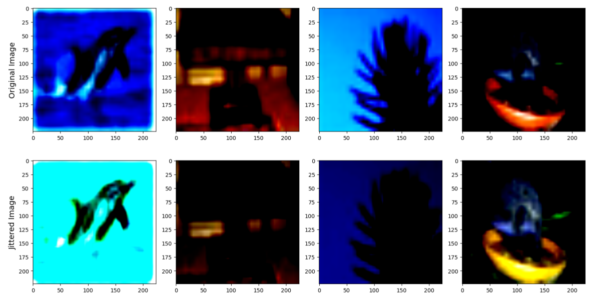

Figure 10. Visualising how the dataset changes when a color jitter transformation is applied to it

Comparison with Larger Teacher Model

As observed in Figure 5, there is a slight performance dip of approximately **0.5%** when comparing the VGG16 distilled student to the VGG19 distilled student. This suggests that a more complex or stronger teacher model, such as VGG19, does not always guarantee an improvement in the performance of the student model during knowledge distillation. This could be due to the potential mismatch in the complexity of the teacher model relative to the student's capacity, where the student may struggle to effectively learn from a more complex teacher. The results highlight the importance of balancing teacher-student model compatibility in knowledge distillation, as well as the potential diminishing returns from using more powerful teachers if the student model cannot adequately capture or benefit from the additional knowledge.

Task 4: Analysis

4.1. Convenience and Practicality

Pruning is often regarded as one of the more intuitive methods. The initial implementation can be straightforward, as many deep learning frameworks provide built-in support for pruning techniques. However, effective pruning requires careful consideration of which weights to prune and how to retrain the model afterward to maintain its performance. This step can add time to the implementation process, especially when determining the optimal pruning strategy.

Quantization, while also accessible, introduces its own set of challenges. The quantization technique, like **PTQ** and **QAT** can significantly affect the model's performance. The choice between these methods requires experimentation and may necessitate a deeper understanding of the model's behavior and architecture. Therefore, while quantization can be convenient with the right tools, it often demands more in-depth tuning and validation to achieve desired results.

Knowledge Distillation is the most involved technique. It can necessitate additional training cycles and potentially complex adjustments to the training process. The effectiveness of knowledge distillation is influenced by several factors, such as the choice of distillation loss functions and the architecture of the student model. Consequently, it requires conducting extensive experiments to optimize these aspects, leading to a higher implementation complexity compared to pruning and quantization.

4.2. Potential Drawbacks and Advantages

Pruning offers significant advantages, particularly in reducing model size and inference time without heavily compromising performance. By eliminating less important weights, pruned models can be made more efficient and faster, which is especially beneficial for deployment on resource-constrained devices. However, a major drawback of pruning is the risk of performance degradation if not executed carefully. If too many important weights are removed, or if the model is not adequately retrained post-pruning, the overall accuracy can suffer. Additionally, pruning can introduce sparsity in the model, which may not always translate into improved inference speed, particularly on hardware that does not exploit sparse operations efficiently. Edge cases can also arise, such as when pruning leads to overfitting on specific datasets, causing the model to generalize poorly to unseen data.

Quantization provides a substantial advantage in terms of reducing the model's memory footprint and accelerating inference, particularly on hardware optimized for low-precision arithmetic. However, one of the primary drawbacks is the potential loss of accuracy that can occur during the quantization process, especially at very low bit-widths (e.g., **int8** or **int4**), which can lead to significant information loss. In experiments, certain architectures demonstrated that quantization could degrade performance, especially for tasks requiring high precision. Furthermore, quantization can introduce numerical instability, particularly in layers that are sensitive to changes in input distribution, leading to unexpected outcomes in edge cases. For example, models that are quantized without proper calibration can behave unpredictably on certain inputs, undermining their robustness.

Knowledge Distillation presents a unique advantage in transferring knowledge from a large, complex model to a smaller, more efficient one. This can result in a smaller model that retains much of the accuracy of its larger counterpart, making it particularly suitable for deployment scenarios where computational resources are limited. However, one notable drawback is that the performance of the distilled model heavily relies on the quality of the teacher model. If the teacher model is poorly trained or lacks robustness, the student model will likely inherit these deficiencies, potentially leading to worse performance compared to traditional training methods. Also, considering that choosing a larger teacher does not always result in better distillation makes the choice of teacher even more crucial and time-consuming. Furthermore, the distillation process may require additional computational resources during training due to the need for dual training (teacher and student), which can be a significant drawback in time-sensitive applications. Edge cases may also arise where the distilled model fails to generalize as well as the original model, particularly in tasks with high variability or noise.

4.3. Effectiveness Given a Fixed Compute Budget

Pruning can be highly effective in scenarios where there are strict limitations on memory and processing power. By selectively removing less significant weights from the model, it reduces both the size and computational demands without necessarily requiring a complete retraining from scratch. This simplicity can lead to faster deployment times and lower memory usage during inference, making it particularly suitable for edge devices with constrained resources. However, the effectiveness of pruning is highly dependent on the method used to determine which weights to remove. If pruning is not performed judiciously, it can lead to a loss of accuracy that negates the benefits of reduced resource usage. Aggressive pruning without careful retraining may lead to deteriorating model performance.

Quantization excels in scenarios where inference speed is critical and the model needs to operate in real-time on limited hardware. It significantly reduces memory bandwidth and computation time. However, the challenge lies in the potential accuracy loss, particularly when using very low bit-widths. It often requires additional calibration steps to mitigate accuracy loss, adding complexity to the process. Therefore, while quantization can improve inference efficiency significantly, it may not be the best option when the accuracy of the model is paramount.

Knowledge Distillation can provide a favorable trade-off between simplicity and performance, particularly when resources are fixed. The distilled model, which is trained to replicate the behavior of a larger, more complex teacher model, often maintains high accuracy while being more computationally efficient. However, the distillation process can be resource-intensive during training, as it requires maintaining both the teacher and student models simultaneously, which may exceed a limited compute budget. Despite this, once the student model is trained, it can deliver efficient inference without the overhead of the teacher model.

4.4. Can These Approaches Be Used Together

Pruning and Quantization: Combining these two techniques can yield significant benefits. Pruning reduces the model's size and computational complexity by removing unnecessary weights, making it an ideal candidate for subsequent quantization. By first pruning the model, the remaining parameters can be quantized more effectively and efficiently in Post Training Quantization, potentially minimizing the adverse effects of quantization-induced accuracy loss. This combination also reduces the memory footprint further, making it suitable for deployment in resource-constrained environments, such as mobile devices or IoT applications.

Knowledge Distillation and Quantization: The integration of Knowledge Distillation with Quantization can also enhance model efficiency. By first training a smaller student model to mimic a larger teacher model, we can achieve high accuracy in a compact form. This student model can then be quantized to improve inference speed and reduce memory usage. In practice, models distilled before quantization inherit the robustness of the teacher model while retaining the advantages of lower precision, as compared to directly quantizing the teacher model, which typically suffers from more significant performance loss due to its larger parameter space.

Pruning and Knowledge Distillation: While the combination of pruning and distillation may not be as straightforward as the previous pairs, it still holds promise. Pruning can simplify the teacher model before distilling it to the student model, thus making the distillation process more efficient. However, aggressive pruning of the teacher could diminish the valuable information it provides during distillation. Lightly pruning the teacher model while still preserving its essential features can lead to an effective student model with improved performance and reduced size.

Simultaneous Application: The concurrent use of all three techniques presents a compelling avenue for enhancing model efficiency. By first pruning a model, then distilling it to a smaller version, and finally quantizing that distilled model, one could create an extremely efficient architecture without substantial performance loss. However, this approach requires careful tuning and validation at each stage to ensure that the cumulative effects of these techniques do not lead to significant accuracy degradation, in addition to the compute required to combine these 3 compression techniques.

4.5. Summary

Based on the comprehensive evaluation of these three techniques, a combination of Structured Pruning and Post Training Quantization emerges as the most effective overall approach for model compression. This pairing leverages the strengths of both techniques: Pruning first reduces the model's complexity, making it a suitable candidate for Quantization, which further decreases memory usage and accelerates inference speed without drastically compromising accuracy utilising the PTQ method, and structural approach of pruning weights, with saves on space as well, without hurting its accuracy. The synergy between these techniques aligns well with practical deployment goals, especially in environments with stringent resource constraints.

While Knowledge Distillation is an excellent strategy for improving the accuracy of smaller models, its added complexity and training overhead may not always justify the benefits when simpler combinations of Pruning and Quantization can achieve similar or superior results. Therefore, for applications where reducing memory usage and speeding up inference are paramount, the integrated approach of Pruning followed by Quantization stands as the recommended strategy.This approach not only optimizes model performance but also aligns with practical deployment scenarios across various applications, ensuring that computational resources are utilized effectively while preserving accuracy.

6. Conclusion

In conclusion, model compression techniques such as pruning, quantization, and knowledge distillation offer effective strategies to reduce the size and computational demands of deep learning models without sacrificing too much accuracy. These methods are crucial for deploying neural networks on resource-limited devices, enabling real-time applications in areas like virtual and augmented reality, and smart devices. Through careful application of these approaches, we can create efficient, compact models capable of maintaining performance, which is essential for advancing AI in distributed and embedded systems.

7. Contributions

Task	**Done by**
task 1	Adeen
task 2	Adeen
task 3	Huraira
task 4	Huraira

8. References

[1] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[2] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015.

[3] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, October 2019.

[4] Akshay D. Mishra, Mostafa Rastegari, Rania Ali Amsal, Yelysei Bondarenko, Maart van Baalen, and Tijmen Blankevoort. Navigating the paper on neural network quantization, 2021.

[5] Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: left inference scaling laws, 2023.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[7] Adrian Radford, Nicolas Baels, Sameh Ebrahimi Kahou, Antoine Chasson, Carlo Gatta, and Yoshua Bengio. Prietas: Hints for thin deep nets, 2021.

[8] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation, 2022.

[9] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation, 2019.

[10] Utkarsh Ojha, Yuheng Li, Anirudh Sundara Rajan, Yingyu Liang, and Yong Jae Lee. What knowledge gets distilled in knowledge distillation?, 2023.

[11] Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A. Alemi, and Andrew Gordon Wilson. Does knowledge distillation really work?, 2021.

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[13] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation, 2022.

[14] Andrey Kuzmin, Markus Nagel, Mart van Baalen, Arash Behboodi, and Tijmen Blankevoort. Pruning vs quantization: Which is better?, 2024.