Deep Models Compression

Jawad Saeed Muhammad Saad Haroon Daanish Uddin Khan

Abstract

Model compression techniques have gained prominence as neural network sizes continue to grow, making storage and computational efficiency critical, particularly for deployment on resource-constrained devices. This report explores three primary model compression techniques: Pruning, Quantization, and Knowledge Distillation, applied to a VGG-11 model trained on the CIFAR-100 dataset. We evaluate the effectiveness of structured and unstructured pruning, comparing their impact on model accuracy, interpretability, and inference efficiency. Additionally, we perform quantization experiments using various bit-widths, examining the trade-offs between precision and computational performance. Knowledge Distillation is used to transfer information from a larger teacher model to a student model, with different distillation techniques explored to assess their effect on model performance and feature alignment. The results highlight the strengths and trade-offs of each technique, with structured pruning and quantization offering computational gains and Knowledge Distillation enabling smaller models to retain high accuracy. These findings inform practical considerations for deploying compressed deep learning models on limited-resource devices. The repository for this can be accessed using the following link: https://github.com/Jawamegamind/PA3-Deep-Models-Compresion

1. Introduction

Deep learning has revolutionized various fields by enabling large-scale neural networks to achieve remarkable performance across a range of tasks. However, deploying these models on devices with limited resources, such as mobile phones and embedded systems, presents a significant challenge due to their high memory and computational requirements. Model compression techniques, such as Pruning, Quantization, and Knowledge Distillation, have emerged as solutions to address these challenges by reducing model size and computational demands without substantial loss of accuracy.

In this assignment, we apply these compression techniques to a VGG-11 model on the CIFAR-100 dataset, aiming to understand the trade-offs between computational efficiency and model performance. Pruning methods are used to reduce model parameters by eliminating redundant connections. We explore both unstructured pruning, which removes individual weights, and structured pruning, which removes entire channels, to assess the impact on accuracy and inference speed. Quantization reduces the bit-width of weights and activations, allowing for more compact model representation and faster computation on supported hardware. Additionally, Knowledge Distillation transfers knowledge from a larger teacher model to a smaller student model, preserving performance while significantly reducing model complexity.

This report provides an in-depth analysis of each compression method, presenting the methodology, results, and insights gained from applying these techniques. By examining the effects of different sparsity levels, bit-widths, and distillation methods, we aim to establish a comprehensive understanding of how these techniques can be combined or applied individually to achieve efficient deep learning models suitable for deployment in resource-limited environments.

2. Methodology

2.1. Task 1: Pruning on VGG-11 for CIFAR-100

In Task 1, we explored pruning techniques on a VGG-11 model adapted for CIFAR-100 classification, focusing on two types of pruning: unstructured and structured. The goal was to reduce the model's parameter count while maintaining high performance by selectively removing weights (unstructured) or entire channels (structured). We analyzed model sensitivity to pruning, determined optimal sparsity levels, and evaluated the effect of pruning on both accuracy and weight distributions.

Model Preparation and Fine-Tuning

We initialized a VGG-11 model pretrained on ImageNet and adapted it for CIFAR-100 classification by modifying the final fully connected layer to output 100 classes. The

model's feature extraction layers were frozen, with only the final layer trainable for fine-tuning. Fine-tuning was conducted on CIFAR-100 for 3 epochs using the Adam optimizer and Cross-Entropy Loss.

- **Data Preparation:** CIFAR-100 images were resized to 224×224 to match VGG-11's input requirements and normalized using CIFAR-100-specific mean and standard deviation values.
- **Training Configuration:** The learning rate was set to 0.001 with L2 regularization (weight decay of 10^{-4}). Training was conducted over 3 epochs with a batch size of 64. Validation accuracy was computed at the end to establish a baseline.

Sensitivity Analysis and Sparsity Adjustment

In both Task 1.1 and Task 1.2, sensitivity analysis was conducted to understand each layer's tolerance to varying sparsity levels (0.4 to 0.9) without significant accuracy degradation. Each layer was pruned iteratively across these sparsity levels, with original weights restored after each pruning. This analysis informed the assignment of layer-specific sparsity ratios to achieve an overall model sparsity of 25%. Layers identified as more sensitive were assigned lower sparsity, while less sensitive layers were pruned more aggressively to meet the overall target sparsity.

Task 1.1: Unstructured Pruning

- **Pruning Method:** Using L1 norm-based unstructured pruning, individual weights within convolutional and fully connected layers were zeroed out, aiming for a high initial sparsity ratio of 50% per layer for testing. This approach selectively removes weights, creating scattered zeros throughout the layer.

Task 1.2: Structured Pruning

- **Pruning Method:** Structured pruning targeted entire channels (filters) in convolutional layers by ranking channels based on their L2 norms and zeroing out those with the smallest norms. In fully connected layers, pruning was applied by removing units with the lowest L2 norms. This approach retains a contiguous memory layout, improving computational efficiency.

Post-Pruning Analysis and Weight Distributions

For both unstructured and structured pruning tasks, weight distributions were visualized pre- and post-pruning to assess the impact on model parameters. Layer-wise and overall model sparsity were verified to confirm adherence to the target 25% sparsity. The resulting post-pruning accuracy was recorded to analyze the trade-off between sparsity and model performance.

Task 1.3: Evaluation and Analysis of Pruning Effects on VGG-11 for CIFAR-100

In Task 1.3, we aimed to evaluate the effects of varying pruning levels on model performance, focusing on accuracy and interpretability. The goal was to assess how different sparsity levels impact the model's ability to generalize on the CIFAR-100 dataset and to visualize the attention areas using Grad-CAM.

Model Preparation and Loading

We began by loading the baseline, unstructured-pruned, and structured-pruned versions of the VGG-11 model. Each model was trained and saved previously with an adapted classifier layer to support CIFAR-100's 100 classes. The models were loaded from saved checkpoints, moved to the GPU, and verified for correctness.

Grad-CAM Visualizations

To visualize attention areas affected by pruning, we applied Grad-CAM (Gradient-weighted Class Activation Mapping) on a sample image from CIFAR-100. Grad-CAM allowed us to observe how the models, under different pruning types, highlighted parts of the image that contributed most to the classification decision. For each model, the last convolutional layer's activations were used to generate the heatmaps, which were overlaid on the original image for comparison.

Fine-Tuning and Accuracy Comparison

To further evaluate the models' generalization abilities, each pruned model was fine-tuned on the CIFAR-100 dataset for three epochs, with only the classifier's final layer unfrozen. The fine-tuning followed a consistent setup across all models, using the Adam optimizer with a learning rate of 0.001 and L2 regularization. After fine-tuning, we calculated the models' accuracies on the CIFAR-100 test set to measure how well they retained classification performance post-pruning.

Inference Time and Storage Comparison

To investigate the computational efficiency of the pruned models, we measured inference time and storage size. Inference time was computed by averaging results over 50 runs on a sample image. Storage was assessed by calculating the size occupied by non-zero and zero weights, as well as the on-device storage needed for each pruned model. This analysis allowed us to quantify both the storage savings and computational load reductions achieved by pruning.

SPARSITY VS. ACCURACY PLOT

Finally, we assessed the effect of increasing sparsity on the baseline model's performance by iteratively applying unstructured pruning at target sparsity levels of 0, 10%, 25%, 50%, and 75%. Each pruned version was fine-tuned for three epochs on CIFAR-100, and the resulting test set accuracy was recorded. The accuracy was then plotted against the target sparsity levels, providing a visualization of how model performance degrades with increased pruning severity.

2.2. Task 2

2.2.1. PREPARATION

Preparation for this task involved the loading of the **VGG11** and the **CIFAR-100** dataset. Since VGG11 is built for ImageNet classification the last layer was modified for 100 class classification. Additionally, when loading in the CIFAR-100 dataset the images were resized to **224x224** and were normalized for best results on CIFAR-100.

2.2.2. FINETUNING

Before implementing QAT and PTQ for the various bit widths, we fine-tuned the VGG11 model on the CIFAR-100 dataset for 5 epochs. The model's last layer in the linear classifier was modified for 100 class classification instead of 1000 class. Following fine-tuning we achieved a baseline test accuracy of 61.85% which was used for comparison with the PTQ and QAT.

2.2.3. QUANTIZATION SETUP

For this task since the goal is to quantize models at different bit widths, we made use of the **torchao** library for this. Specifically, **INT4** and **INT8** bit widhts required the use of the library for PTQ and QAT. On the other hand, PTQ and QAT for **FLOAT16** and **BFLOAT16** were carried out by simply casting the model to the specific data type using the .to function.

2.3. Task 3

In our knowledge distillation (KD) experiments, we examined multiple techniques: **logit matching**, **hint-based distillation**, **contrastive representation distillation (CRD)**, and tested specialized KD properties such as **localization knowledge transfer** and **color invariance**. We also evaluated the impact of teacher model size on distillation efficacy by comparing student models trained with VGG-16 and VGG-19 teachers. All experiments utilized the CIFAR-100 dataset, with pre-processing transformations including resizing to 224x224 and standard normalization. Below, we detail each distillation method, followed by our evaluation approach.

2.4. Logit Matching

Logit matching uses KL divergence loss between the soft logits of the teacher and student, scaled by a **temperature parameter** $T = 3.0$. This method encourages the student to align its output logits with the teacher, using a loss that approximates:

$$\frac{\partial L_{KL}}{\partial z_i} \approx \frac{1}{NT^2}(z_i - v_i)$$

where z_i and v_i are the logits for the student and teacher, respectively, and N is the total number of logits. The **student and teacher architectures** were VGG-11 and VGG-16, respectively. For training, we used a batch size of 128, **AdamW optimizer** with a learning rate of 0.0001, and **cross-entropy** as a secondary loss component to ensure accurate class predictions.

2.5. Hint-based Distillation

Hint-based distillation aligns intermediate **feature maps** between the student and teacher. We applied an **MSE loss** on feature maps from specific guided and hint layers (layer 17 of VGG-16 and layer 11 of VGG-11) using a regressor to map student features to the teacher's space. The same batch size and optimization settings as logit matching were used, with a **hint weight** of 0.5 applied to the distillation loss.

2.6. Contrastive Representation Distillation (CRD)

CRD introduces a memory buffer to contrast feature representations across batches, reinforcing similar representations in the student for positive pairs while pushing apart negative pairs. Using parameters $N = 4096$ and temperature $T = 0.07$, CRD was implemented on VGG-11 students trained with VGG-16 and VGG-19 teachers. The batch size, learning rate, and training settings were consistent with the other methods.

2.7. Localization and Color Invariance Testing

For **localization knowledge transfer**, we generated **Grad-CAM** visualizations to compare focus alignment between student and teacher models. We quantified similarity with **SSIM scores** to assess each KD technique's efficacy in transferring localized attention from the teacher.

To evaluate **color invariance**, we finetuned a teacher model on color-jittered data and assessed student models' robustness to color variation using this augmented data. We applied each KD method to train the students, then evaluated them on color-jittered validation sets.

2.8. Teacher Size Comparison

To explore the impact of **teacher model size**, we trained students using VGG-16 and VGG-19 as teachers. This allowed us to observe if a larger teacher model would lead to significant improvements in student performance, particularly for methods like CRD, which might benefit from a richer representation space.

3. Results

3.1. Task 1

First, we present the weight distributions of the baseline model and after applying both unstructured and structured pruning. This provides a visualization of how pruning alters the model's weight distribution.

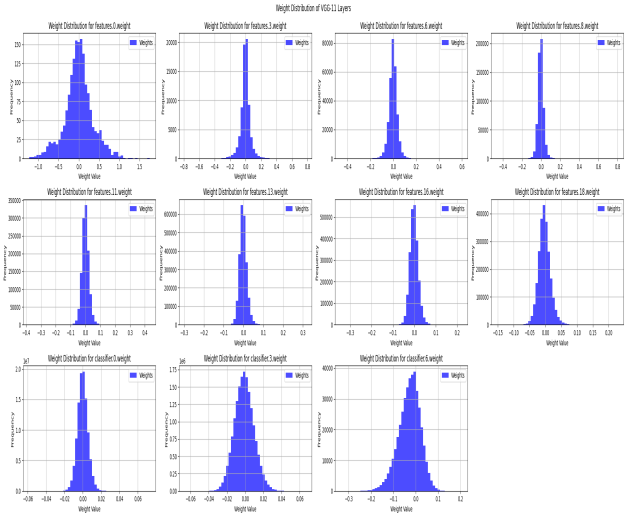

Figure 1. Visualization of the Weight Distribution - (Pre-Pruning)

Figure 1 shows the initial weight distribution of the model before any pruning has been applied. This distribution serves as the baseline reference for comparison with post-pruning distributions.

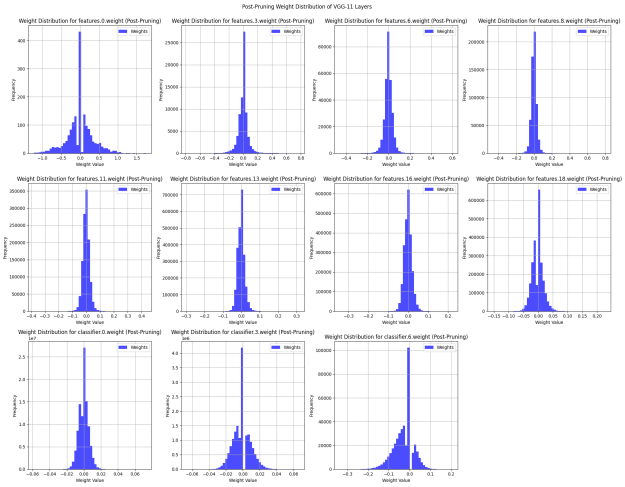

Figure 2. Weight Distribution After Unstructured Pruning

Figure 2 illustrates the weight distribution after applying 50% unstructured pruning. This approach prunes individual weights based on their magnitudes, resulting in sparsity within layers while maintaining the overall layer structure.

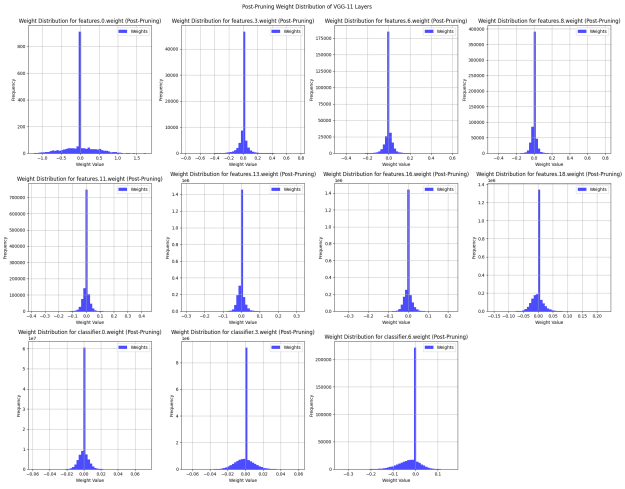

Figure 3. Weight Distribution After Structured Pruning

In Figure 3, we observe the weight distribution after 50% structured pruning, which removes entire channels or filters rather than individual weights. This method changes the network structure in a more coarse-grained manner compared to unstructured pruning.

Next, we conducted a sensitivity analysis for both unstructured and structured pruning approaches. This analysis shows how different layers in the model react to varying sparsity levels, helping us understand which layers are more

critical for preserving accuracy.

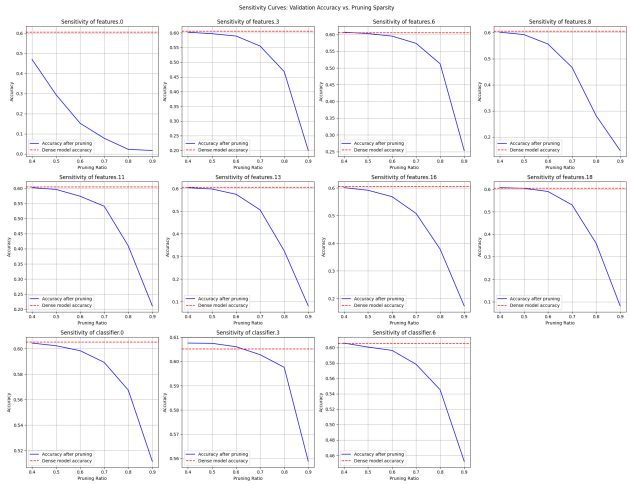

Figure 4. Task 1.1 Unstructured Pruning - Sensitivity Analysis

Figure 4 shows the sensitivity analysis for Task 1.1, where we applied unstructured pruning. This plot helps identify layers that are more resilient to pruning and those that require careful tuning to avoid significant accuracy loss.

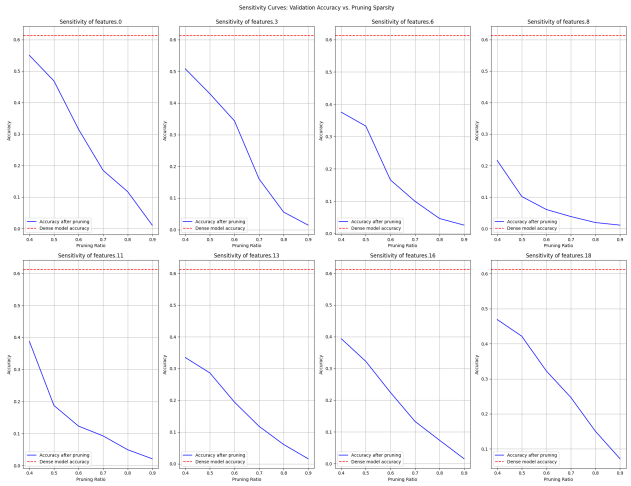

Figure 5. Task 1.2 Structured Pruning - Sensitivity Analysis

Figure 5 presents the sensitivity analysis for Task 1.2, where structured pruning was applied. This plot offers insights into how channel-wise pruning affects different layers, helping us determine which layers are critical for maintaining model performance.

3.2. Task 1.3: Comparison and Analysis

In Task 1.3, we compared the baseline model, unstructured pruned model, and structured pruned model using various metrics. These include Grad-CAM visualizations to observe differences in feature focusing, accuracy comparison, inference time, and a qualitative analysis of pruning effects.

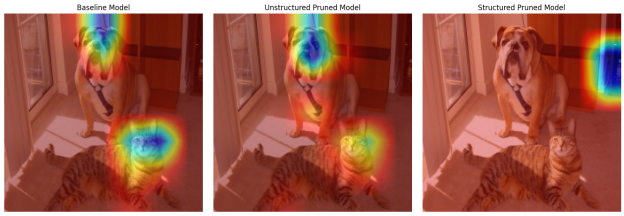

Figure 6. Task 1.3 - Grad-CAM Visualizations for Baseline, Unstructured Pruned, and Structured Pruned Models

Figure 6 shows Grad-CAM visualizations for all three models, highlighting the differences in feature focus across the baseline, unstructured pruned, and structured pruned models. These visualizations help assess how pruning impacts the model's attention to specific image features.

Next, we present the accuracy comparison across the models, showing how unstructured and structured pruning impact performance relative to the baseline.

Accuracy Comparison of Baseline, Unstructured Pruned, and Structured Pruned Models on CIFAR-100 Test Set

	Model	Accuracy (%)
0	Baseline Model	61.000000
1	Unstructured Pruned Model	42.920000
2	Structured Pruned Model	26.380000

Figure 7. Task 1.3 - Accuracy Comparison of Baseline, Unstructured Pruned, and Structured Pruned Models

Figure 7 displays the accuracy comparison table for all three models on the CIFAR-100 dataset. It highlights the trade-offs between model sparsity and accuracy for unstructured and structured pruning.

The following figure shows inference time and storage requirements for each model, highlighting the computational benefits and trade-offs of each pruning approach.

Inference Time and Storage Comparison with Zero and Non-Zero Weights and On-Device Storage

	Model	Avg Inference Time (ms)	Non-Zero Storage Size (MB)	Zero Storage Size (MB)	Total Storage Size (MB)
0	Baseline Model	4.178910	492.725342	0.000000	492.775402
1	Unstructured Pruned Model	4.134698	370.150112	122.575230	492.775797
2	Structured Pruned Model	3.990059	483.934528	8.790813	492.775747

Figure 8. Task 1.3 - Inference Time and Storage Comparison

In Figure 8, we see a comparison of inference time and storage requirements for each model. This analysis reveals the efficiency improvements achievable through pruning while examining the trade-offs in storage and computational speed.

In the qualitative analysis table, we compare structured and unstructured pruning with respect to aspects like hardware compatibility, memory access patterns, and storage requirements.

Comparison of Structured vs. Unstructured Pruning

	Aspect	Structured Pruning	Unstructured Pruning
0	Pruning Granularity	Entire channels, filters, or blocks are removed.	Individual weights are pruned within layers.
1	Hardware Compatibility	Efficient on general-purpose hardware (GPUs/TPUs) due to dense operations.	May need specialized hardware for efficient sparse matrix operations.
2	Memory Access Patterns	Contiguous, optimized for general memory access.	Irregular, leading to complex memory access and indexing.
3	Parallelism	Maintains high parallelism due to dense structure.	Reduced parallelism; sparse operations may hinder effective usage.
4	Implementation Complexity	Easier to implement with standard libraries and retraining.	Requires complex handling and often sparse matrix libraries.
5	Impact on Model Accuracy	Coarser granularity can cause higher accuracy loss without careful retraining.	Achieves higher sparsity with selective pruning, often less accuracy degradation.
6	Storage Requirements	Compact, fits standard storage formats without extra encoding.	Often requires specialized sparse storage formats for efficiency.

Figure 9. Task 1.3 - Qualitative Analysis of Structured vs. Unstructured Pruning

Figure 9 provides a qualitative comparison between structured and unstructured pruning. It highlights differences in hardware requirements, memory access efficiency, and storage benefits for each approach.

Finally, we present the Accuracy vs. Sparsity graph for unstructured pruning, which shows how pruning impacts model accuracy as the sparsity level increases.

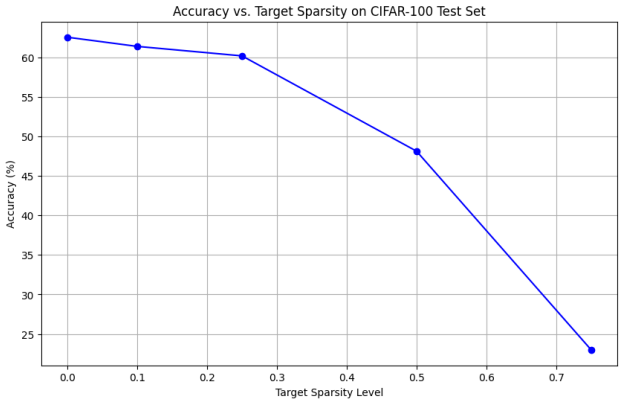

Figure 10. Task 1.3 - Accuracy vs. Sparsity Graph (Unstructured Pruning)

Figure 10 illustrates the relationship between accuracy and sparsity for the unstructured pruning approach. It provides insights into the tolerance of model accuracy at various sparsity levels, useful for determining optimal sparsity levels.

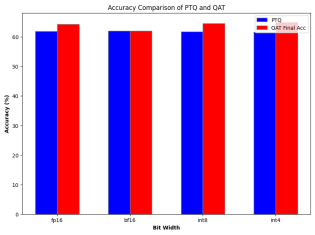

Figure 11. PTQ vs QAT Accuracies at Different Bit Widths

3.3. Task 2

3.3.1. Scaling Law Analysis

Using the results from the experiments, we observed the following trends in accuracy for different bit-widths across Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT):

- **FP16 vs. BF16:** Both FP16 and BF16 achieved similar PTQ accuracies (61.9%), with FP16 showing a slight improvement after QAT (final accuracy 64.31%). This minor difference highlights the greater numerical stability of BF16 due to its extra exponent bits. However, BF16's stability may limit further recovery with QAT, as fewer numerical errors arise. These results are supported by the findings in Figure 11.

- **INT8:** INT8 PTQ accuracy is close to FP16 and BF16, but QAT significantly improves accuracy from 61.83% to 64.59%. This suggests that QAT effectively recovers accuracy at INT8, likely because QAT helps mitigate the quantization errors introduced at this bit-width.

- **INT4:** INT4 QAT accuracy is 64.90%, indicating an improvement over the PTQ performance, which comes out to be to be 61.64%. The INT4 results suggest diminishing returns due to low precision, where additional quantization error reduces the potential accuracy gains.

3.3.2. Comparing fp16 and bf16

Based on the results both the data types are quite close in terms of accuracy. Bf16 achieves a slightly higher accuracy of 61.98% than fp16's 61.88% following PTQ. However, following QAT, fp16 had a higher final accuracy of 64.31% than bf16 with 61.98%, which solidifies the claim that bf16 is more stable than fp16.

3.4. Task 3

Our experiments evaluate the performance of various knowledge distillation (KD) techniques across several metrics, including accuracy, KL divergence, SSIM score for localization alignment, and color invariance. Tables and figures are referenced to highlight specific results.

Accuracy Comparison: The accuracy of each KD method with VGG-11 students trained on VGG-16 and VGG-19 teachers is shown in Figure 12 and Table **??**. The teacher models achieved 67.78% (VGG-16) and 67.62% (VGG-19) accuracy, while the highest student accuracy was observed with hint-based KD (47.97% for VGG-16 teacher). Contrastive Representation Distillation (CRD) saw a significant improvement with the larger teacher (47.62% for VGG-19), suggesting that CRD benefits from a more complex teacher model.

KL Divergence and SSIM Scores: To measure distributional alignment between teacher and student, we computed KL divergence and SSIM scores. Figure **??** illustrates that hint-based distillation achieved the lowest KL divergence (0.4829) and highest SSIM score (0.2946), indicating the closest alignment with the teacher model. Logit matching followed closely, while CRD and independent students showed higher KL divergence, reflecting less alignment (Table 13).

Color Invariance Performance: We evaluated color invariance by testing student models on a color-jittered validation set (Figure **??**). The teacher model fine-tuned on color-jittered data achieved 62.76% accuracy, while hint-based distillation retained the most color robustness among the students (42.29%). CRD exhibited the lowest color invariance (29.95%), suggesting limited transfer of robustness from the teacher.

GradCAM Visualizations: Figure 14 shows GradCAM heatmaps for the models on a sample image. Hint-based distillation achieved the highest focus alignment with the teacher model, as indicated by the SSIM scores, while logit matching showed close alignment as well. CRD and independent models displayed less localized focus.

4. Discussion

4.1. Task 1

Discussion for Task 1.1: Unstructured Pruning Results

In Task 1.1, the results reveal insights into how unstructured pruning affects the model's weight distribution, sensitivity to sparsity, and overall performance.

Weight Distribution Analysis

The pre-pruning weight distribution shows a centered spread of weights around zero, indicating a balanced initialization and training across layers. Post-pruning, the distribution shifts significantly as many weights are set to zero. This result is expected with unstructured pruning, as it reduces the active parameters without specific structural constraints. The distributions post-pruning show a dense concentration near zero with fewer high-magnitude weights, reflecting the selective retention of important weights. This shift demonstrates the pruning's effectiveness in reducing model complexity, as unimportant weights are removed while the essential parameters remain largely unaffected.

Sensitivity Analysis Results

The sensitivity analysis results reveal the varying tolerance levels of different layers to pruning:

- **Convolutional Layers**: These layers exhibited resilience to sparsity, as evidenced by minimal accuracy degradation even at higher pruning ratios. This suggests that convolutional filters can still capture important features despite a substantial reduction in active weights, possibly because spatial patterns in images remain identifiable with fewer parameters.
- **Fully Connected Layers**: In contrast, fully connected layers showed significant sensitivity to pruning, particularly the final classification layer. Accuracy dropped rapidly as the pruning ratio increased, indicating that these layers play a crucial role in fine-grained class distinctions. The results suggest that excessive pruning in fully connected layers can lead to notable performance degradation.

These findings from the sensitivity analysis underscore the importance of preserving more weights in fully connected layers compared to convolutional ones, guiding the need for a differentiated approach to sparsity allocation.

Layer-Specific Sparsity Allocation and Rationale

Based on the sensitivity results, we assigned customized sparsity ratios to achieve the target 25% overall sparsity while minimizing accuracy loss. Convolutional layers were pruned more aggressively, with a sparsity ratio of 53%, given their resilience, while fully connected layers were pruned conservatively.

- **Convolutional Layers (53% Sparsity)**: These layers can tolerate a high level of sparsity without a substantial impact on accuracy. The 53% sparsity ratio allowed

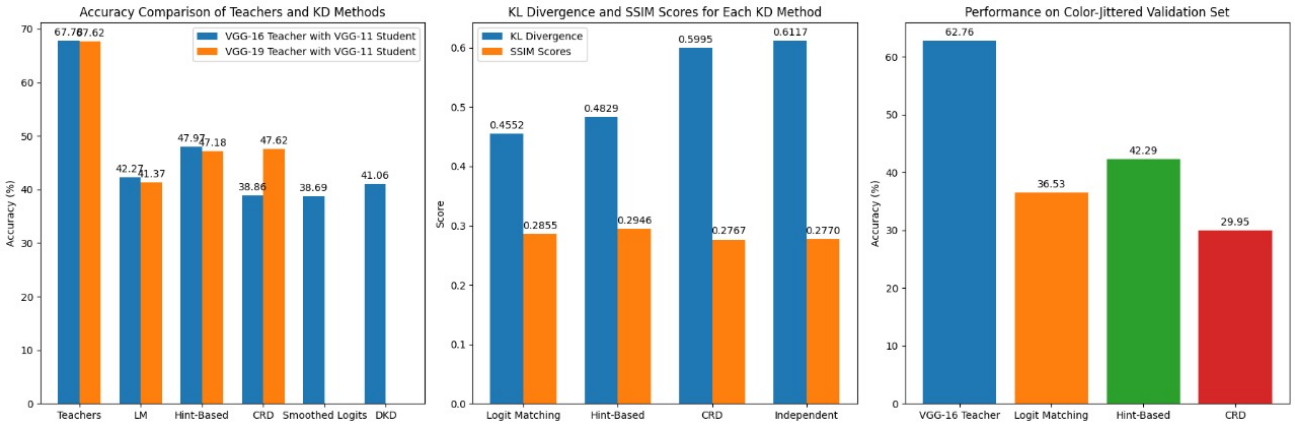

Figure 12. (Left) Accuracy Comparison of Teachers and KD Methods; (Middle) KL Divergence and SSIM Scores; (Right) Performance on Color-Jittered Validation Set.

Metric	VGG-16 Teacher	Logit Matching	Hint-Based	CRD	Smoothed Teacher Logits	DKD
Accuracy (%) - VGG-16 Teacher with VGG-11 Student	67.78	42.27	47.97	38.86	38.69	41.06
Accuracy (%) - VGG-19 Teacher with VGG-11 Student	67.62	41.37	47.18	47.62	-	-
KL Divergence (Average over Query Images)	-	0.4552	0.4829	0.5995	-	-
SSIM Score (Average)	-	0.2855	0.2946	0.2767	-	-
Color-Jittered Validation Set Accuracy (%) - VGG-16 Teacher	62.76	36.53	42.29	29.95	-	-

Figure 13. Summary Table of Main Results across KD Methods and Metrics.

us to achieve significant parameter reduction without compromising the model's ability to capture essential spatial features.

- **Fully Connected Layers (23% to 38% Sparsity)**: For fully connected layers, particularly the final classification layer, lower sparsity ratios (23% to 38%) were chosen to maintain critical connections necessary for accurate predictions. This selective allocation preserved essential parameters in the layers most sensitive to pruning, mitigating the accuracy loss.

The actual sparsity achieved in each layer aligns closely with the assigned sparsity, as shown in Table 1. This targeted allocation approach enabled us to meet the overall sparsity target of 25% while preserving model accuracy close to the baseline.

Table 1. Assigned and Actual Sparsity Ratios for Each Layer in Task 1.1

Layer	Assigned Sparsity	Actual Sparsity
features.0	0.53	0.530
features.3	0.53	0.530
features.6	0.53	0.530
features.8	0.53	0.530
features.11	0.53	0.530
features.13	0.53	0.530
features.16	0.53	0.530
features.18	0.53	0.530
classifier.0	0.23	0.230
classifier.3	0.23	0.230
classifier.6	0.38	0.380
Overall	**0.25**	**0.249974**

Conclusion of Task 1.1 Results

Overall, the results from Task 1.1 demonstrate that unstructured pruning can effectively reduce model complexity and parameter count while retaining accuracy, provided that layer-specific sensitivities are accounted for. The approach of assigning higher sparsity to convolutional layers and

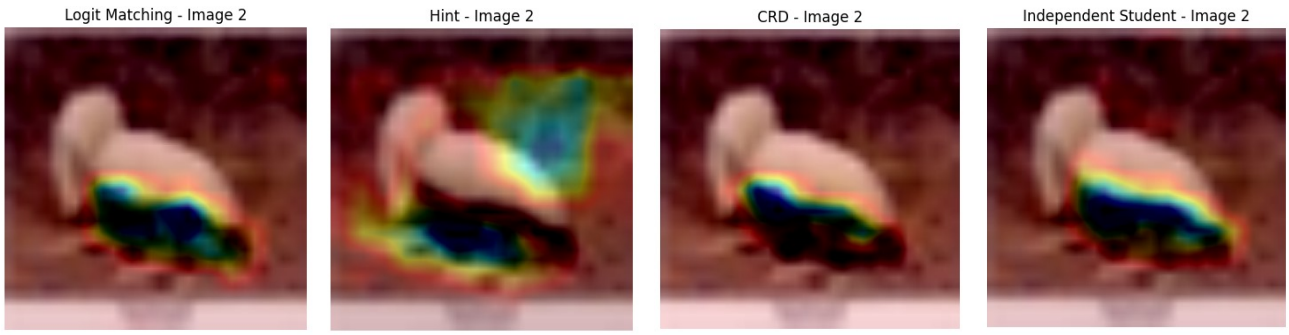

Figure 14. GradCAM Visualizations for Each KD Method (Sample Image).

lower sparsity to fully connected layers allowed us to reach the desired 25% sparsity target with minimal performance impact. This strategy balances model efficiency and accuracy, making it a practical choice for model compression in resource-constrained environments.

4.2. Task 1.2: Discussion of Results for Structured Pruning

4.2.1. Weight Distribution Analysis

In Task 1.2, we applied structured pruning to the model, targeting entire channels rather than individual weights within layers. This approach led to a more organized reduction in weight values, making the pruned layers exhibit more contiguous and regular weight distributions. Observing the post-pruning weight distributions, we note a significant concentration of weights around zero across layers, indicating the effect of structured pruning in reducing model complexity without introducing irregular sparsity patterns. This structured pruning results in a model with a reduced memory footprint and computational demand, which is essential for deployment on resource-constrained hardware.

4.2.2. Sensitivity Analysis Findings

We conducted a detailed sensitivity analysis for each layer to assess the impact of structured pruning on validation accuracy. Key observations from this analysis include:

- **Layer Sensitivity:** The analysis showed that initial layers, such as `features.0` and `features.3`, are highly sensitive to pruning, as accuracy decreases sharply when sparsity increases beyond 50%. This indicates that these layers play a crucial role in capturing fundamental low-level features necessary for the model's overall performance.
- **Resilience of Deeper Layers:** Deeper layers, such as `features.16` and `features.18`, displayed greater tolerance to higher sparsity ratios. For instance, `features.18` maintained an accuracy of approximately 24.7% at 70% sparsity, highlighting that deeper layers can handle greater pruning without significantly compromising accuracy. This aligns with the deeper layers' role in capturing more abstract, task-specific features rather than foundational patterns.
- **Accuracy Thresholds:** Layers such as `features.0` and `features.3` experienced rapid accuracy degradation beyond 50-60% sparsity, necessitating the application of lower sparsity ratios to preserve model performance. In contrast, layers like `features.18` could tolerate sparsity ratios as high as 80%, making them suitable for more aggressive pruning.

4.2.3. Final Sparsity Assignments and Strategy

Based on the sensitivity analysis results, we assigned specific sparsity ratios to each layer, balancing the model's sparsity and accuracy preservation. The final sparsity allocation aimed to maintain an overall model sparsity of 25% while distributing pruning across layers in a way that minimized accuracy loss:

- **Low Sparsity in Sensitive Layers:** Initial layers, such as `features.0` and `features.3`, received lower sparsity ratios (22-30%) to preserve their ability to capture essential low-level features, thereby preventing a sharp decline in model performance.
- **Higher Sparsity in Robust Layers:** The deeper layers, such as `features.16` and `features.18`, were assigned higher sparsity ratios (up to 30%). These layers retained reasonable accuracy despite higher sparsity, enabling us to reach the target 25% overall sparsity without a substantial impact on the model's predictive capabilities.

The table below summarizes the assigned and actual sparsity levels for each layer, illustrating how we adjusted each layer's sparsity based on its sensitivity to pruning.

Table 2. Per-Layer Sparsity Assignments for Structured Pruning

Layer	Assigned Sparsity	Actual Sparsity
features.0	0.27	0.270000
features.3	0.22	0.220000
features.6	0.31	0.310000
features.8	0.30	0.300000
features.11	0.25	0.250000
features.13	0.26	0.260000
features.16	0.24	0.240000
features.18	0.23	0.230000
Overall	0.25	0.250003

4.2.4. Rationale for Sparsity Distribution

The chosen sparsity distribution was aimed at achieving an optimal balance between model sparsity and performance:

- **Retention of Low-Level Features in Early Layers:** The initial layers were assigned lower sparsity ratios to preserve the model's capacity to capture fundamental features, which are crucial for overall accuracy. By maintaining sufficient weights in these layers, we prevent degradation in the model's foundational learning.
- **Leveraging Robustness of Deeper Layers:** Deeper layers were pruned more aggressively, taking advantage of their resilience to high sparsity levels. This approach allowed us to meet the 25% sparsity target while conserving accuracy, as these layers focus on more abstract, task-specific patterns that are less sensitive to sparsity changes.

In conclusion, Task 1.2's structured pruning approach effectively balanced sparsity and accuracy by distributing pruning according to layer sensitivity, resulting in a model with reduced complexity and minimal performance loss.

4.3. Task 1.3: Discussion of Results

4.3.1. Grad-CAM Visualizations

The Grad-CAM visualizations provide insight into the model's interpretability and focus regions for each pruned version compared to the baseline.

Baseline Model: The baseline model focuses on both the dog and the cat, highlighting essential areas around their faces. This suggests that the model effectively considers multiple objects in the scene and attends to critical features such as the eyes and nose to identify each animal.

Unstructured Pruned Model: The unstructured pruned model retains a similar focus to the baseline, with strong attention on the dog's head and the cat's face. While the intensity of focus is slightly reduced, the primary regions of interest remain the same. This indicates that unstructured pruning has preserved the model's interpretability and focus regions, which is consistent with the approach's design of selectively removing individual weights without losing entire channels or features.

Structured Pruned Model: In contrast, the structured pruned model shows a more noticeable shift in focus. Although the dog's face remains in focus, the attention on the cat's face has diminished, with heatmap regions shifting towards the background on the right side of the image. This behavior aligns with structured pruning's tendency to remove entire channels, potentially leading to a loss of feature-specific information.

Key Observations: 1. *Focus Retention:* Both the baseline and unstructured pruned models show similar focus regions, suggesting that unstructured pruning preserves the model's interpretability. 2. *Focus Shift:* The structured pruned model shows a shift in focus, which may indicate a reduction in the model's ability to capture all relevant objects in complex scenes.

4.3.2. Accuracy Comparison Table

The accuracy comparison table illustrates the performance differences between the baseline model, the unstructured pruned model, and the structured pruned model on the CIFAR-100 test set. The baseline model achieves an accuracy of 61%, while the unstructured pruned model and structured pruned model achieve 42.92% and 26.38%, respectively.

This result suggests that unstructured pruning retains a higher level of accuracy compared to structured pruning, as it selectively removes weights rather than entire channels, preserving more of the model's original representational capacity. Structured pruning, while potentially more efficient, appears to lead to a greater loss in accuracy, likely due to the removal of entire channels, which may significantly impact the model's expressiveness.

4.3.3. Qualitative Analysis Table

The qualitative analysis table provides a comparison between structured and unstructured pruning across various aspects, including hardware compatibility and storage requirements. A significant focus is on the hardware considerations for deploying these pruned models.

Hardware Considerations: Both structured and unstructured pruning methods are compatible with general-purpose PC hardware. The pruned models, which are approximately 492MB in size, fit comfortably within the capabilities of standard consumer-grade GPUs and PCs. Zero weights are handled as placeholders, allowing for efficient matrix multiplication without requiring specialized hardware.

However, specialized hardware such as NVIDIA's Tensor Cores provides an advantage, particularly for unstructured pruning. NVIDIA GPUs, optimized for sparse matrix operations, benefit from unstructured pruning by skipping over zero weights, enhancing computational efficiency. Structured pruning, on the other hand, maintains a dense and contiguous memory layout, making it inherently more compatible with general-purpose hardware as it avoids irregular memory access patterns and maintains consistent parallelism.

Inherent Limitations: Unstructured pruning achieves finer granularity with minimal accuracy loss but introduces complexities in memory access patterns, which may impact performance on hardware without sparse optimizations. Structured pruning, though easier to implement on general-purpose hardware, operates at a coarser granularity, which may lead to greater accuracy degradation due to the removal of entire channels.

4.3.4. Inference Time and Storage Table

The inference time and storage comparison table shows the differences in computational efficiency between the baseline model, unstructured pruned model, and structured pruned model. The unstructured pruned model has a slightly longer inference time than the baseline (4.13ms vs. 4.18ms), likely due to irregular memory access. However, it achieves a reduced non-zero storage size, saving around 122MB in storage. The structured pruned model has the shortest inference time (3.99ms) and a minimal increase in zero storage, highlighting its efficiency on general-purpose hardware.

4.3.5. Accuracy vs. Sparsity Plot

The accuracy vs. sparsity plot illustrates the impact of varying sparsity levels on model accuracy. As the sparsity level increases, model accuracy decreases, with a sharp drop observed beyond 50% sparsity. This indicates that while pruning provides computational efficiency, it also leads to a trade-off in performance, especially at higher sparsity levels where critical information may be lost.

4.4. Task 2

4.4.1. Scaling Law Analysis

Based on the results and the scaling laws discussed in the referenced paper, we can derive the following insights:

- **Diminishing Returns with Lower Bit-Widths:** As bit-width decreases from FP16 to INT4, accuracy gains diminish, particularly evident in INT4 performance. The referenced paper shows a similar trend, where accuracy improves as bit-width decreases but degrades below a critical threshold (e.g., INT4 or 3-bit). This is due to increased quantization error and loss of information at very low bit-widths.
- **Impact of QAT at Different Bit-Widths:** QAT is more effective in lower bit-widths, such as INT8 and INT4, where quantization error is higher. For higher bit-widths like FP16 and BF16, QAT's impact on accuracy is comparatively smaller, as the quantization error at these levels is initially low.

4.4.2. Comparing fp16 and bf16

- The main difference between fp16 and bf16 is based on the allocation of bits which leads to different precision. fp16 allocates 5 bits for the exponent and 10 for the mantissa which allows it to achieve greater precision at the cost of increased computational complexity. bf16 allocates 8 bits for the exponent and 7 bits for the mantissa which reduces its precision but results in reduced memory usage.
- The increased number of exponent bits in bf16 means that it can store a wider range of values as compared to fp16 but at a reduced precision. The wider range of values allows for much more stable training as it mitigates the risk of gradient explosions or underflow issues, which can lead to undefined loss values. These, in turn, cause deep networks to not learn anything.
- The results achieved following QAT are support these claims since during QAT for bf16 the training accuracy increased consistently while the testing accuracy remained the same. However this did lead to lower final accuracy as compared to fp16 since the greater precision allowed for more precise updates.

4.5. Task 3

Our experiments reveal distinct strengths and limitations across the different knowledge distillation (KD) techniques evaluated. The following summarizes our primary findings, with references to relevant figures and tables for clarity.

Overall Performance: As shown in Figure 12 and Table **??**, the VGG-16 and VGG-19 teacher models achieved comparable accuracy, around 67.7%. Among the student models, *hint-based distillation consistently outperformed other methods*, achieving the highest accuracy at 47.97% with the VGG-16 teacher. This suggests that aligning intermediate feature maps from the teacher provides the most effective guidance for student learning, especially when training on complex datasets like CIFAR-100.

Impact of Teacher Size: The CRD method showed a notable increase in performance with the larger VGG-19 teacher, reaching 47.62% accuracy (Figure 12). This improvement highlights the *importance of teacher capacity*

in CRD-based methods, where a larger teacher provides a richer set of representations that better support the contrastive learning approach. In contrast, logit matching and hint-based distillation displayed minimal sensitivity to teacher size, suggesting that their performance relies more on feature alignment than on the teacher's overall complexity.

Distributional and Localization Alignment: As indicated by the KL divergence and SSIM scores in Figure **??** and Table 13, hint-based distillation achieved the lowest KL divergence (0.4829) and highest SSIM score (0.2946). This alignment implies that hint-based KD effectively transfers not only output distributions but also localized attention from teacher to student. Logit matching showed similarly strong results, though slightly lower than hint-based KD. The CRD and independent student models displayed higher KL divergence, suggesting limited distributional alignment with the teacher, especially in terms of localized focus (Figure 14).

Color Invariance: Our evaluation on the color-jittered validation set (Figure **??**) revealed that hint-based KD led to the highest student robustness to color variations (42.29%), closely followed by logit matching (36.53%). CRD exhibited the lowest color invariance (29.95%), indicating that *feature-level alignment methods like hint-based distillation are more effective in transferring color robustness* from teacher to student.

Summary and Implications: Overall, our findings suggest that hint-based distillation offers the most balanced approach, excelling in accuracy, distributional alignment, and color invariance. The CRD method, while less robust in alignment and color invariance, benefits significantly from a larger teacher, making it a suitable choice for applications where a high-capacity teacher model is available. For tasks requiring precise attention alignment, such as object localization, hint-based KD is recommended based on its superior SSIM scores and GradCAM visualizations.

These insights underscore the importance of selecting a KD method based on specific objectives, such as whether feature alignment, output distribution alignment, or robustness to color variations is prioritized.

4.6. Task 4

This section compares three model compression techniques—**Pruning**, **Quantization**, and **Knowledge Distillation (KD)**—focusing on their convenience, practical implementation, advantages, and limitations. Each approach is evaluated under specific constraints, including limited computational resources, with additional discussion on their potential synergy when used together.

4.7. Convenience and Practicality

Pruning: Pruning can be straightforward, especially structured pruning, which removes entire layers or filters based on their contribution. Unstructured pruning, which removes individual weights, requires more fine-tuning and computational resources. Typically performed post-training, pruning simplifies the process but often requires fine-tuning to regain lost accuracy, adding complexity. Effective for deep neural networks, pruning may require experimentation to find optimal sparsity without compromising performance.

Quantization: Post-Training Quantization (PTQ) is intuitive, reducing parameters to lower-bit representations. Quantization-Aware Training (QAT) is more complex, requiring adjustments to account for quantized weights. PTQ is easy to implement, while QAT requires a more involved setup. Quantization benefits from hardware that supports lower-bit computations, like GPUs or TPUs. PTQ is simpler, but QAT retains accuracy better for highly quantized models.

Knowledge Distillation (KD): KD requires a separate teacher model to guide training of a smaller student model, adding complexity. The training process for both teacher and student can be time-intensive. KD is practical for large models where a teacher can guide a simpler student, but extensive experimentation is often needed to balance the student model's capacity and accuracy. KD achieves high accuracy in smaller models but depends heavily on the quality of the teacher model.

4.8. Potential Drawbacks and Advantages

Pruning: Reduces model size and improves inference time without additional hardware, ideal for large, overparameterized models. However, pruned models may be less robust, with reduced generalization and potential vulnerabilities such as adversarial susceptibility.

Quantization: Quantization reduces memory and computational demands, particularly beneficial on compatible hardware. Stable and robust if accuracy loss is tolerable, but highly quantized models (e.g., INT4) may lose accuracy. Models requiring precise numerical representation may suffer.

Knowledge Distillation (KD): KD retains high accuracy in small models by mimicking a larger teacher, ideal when performance must meet a specific threshold. However, it relies on a high-quality teacher model and can be resource-intensive. KD is effective when a smaller student closely follows a larger teacher but less effective when a high-quality teacher is unavailable.

4.9. Effectiveness Given a Fixed Compute Budget

Each technique's effectiveness varies under constrained resources:

- **Pruning**: Effective for moderately constrained budgets, as it only requires fine-tuning after pruning. Best suited for environments with limited compute.
- **Quantization**: Post-Training Quantization (PTQ) is low-cost and effective under budget constraints. Quantization-Aware Training (QAT) is more resource-intensive, best suited for scenarios with supported low-bit hardware.
- **Knowledge Distillation (KD)**: Requires both teacher and student model training, making it challenging under limited budgets. However, KD offers high accuracy when resources allow for both teacher and student models.

4.10. Combined Techniques

Pruning + Quantization: A highly effective combination that reduces model size (pruning) and compresses weights (quantization). By first pruning and then quantizing, we minimize accuracy loss from aggressive quantization.

Knowledge Distillation + Quantization: Distilling a high-performance teacher into a small, quantized student model can yield accurate, compact models suitable for edge deployments.

Pruning + Knowledge Distillation: Less common but effective when a smaller, pruned teacher model is needed. However, pruning the student after distillation may reduce its learning capacity.

4.11. Recommendations and Conclusion

Pruning is recommended for simplifying large models and improving inference time with minimal changes to the training process.

Quantization provides efficient compression with minor accuracy loss, ideal for low-compute environments or devices with bit-compatible hardware.

Knowledge Distillation excels in performance-critical applications, though it requires both teacher and student models, making it more resource-intensive.

Most Effective Combination: For an optimal balance of simplicity, accuracy, and efficiency, a combined **Pruning and Quantization** approach is recommended. This combination is highly effective for reducing memory usage and speeding up inference in resource-limited settings. For more demanding applications, **Knowledge Distillation with Quantization** yields a compact and accurate model ideal for edge or mobile deployment.

In conclusion, the choice of compression technique (or combination) should align with deployment needs, resource constraints, and performance goals. By leveraging the strengths of each technique, it is possible to achieve a balance between memory efficiency, inference speed, and accuracy retention.

5. Conclusion

Through extensive experimentation with Pruning, Quantization, and Knowledge Distillation on a VGG-11 model, we gained valuable insights into the trade-offs associated with each compression technique. Unstructured pruning, which selectively removes individual weights, preserves model interpretability and accuracy but introduces irregular memory access patterns that may hinder performance on certain hardware. Structured pruning, on the other hand, provides computational efficiency and compatibility with general-purpose hardware by removing entire channels, though it can lead to more significant accuracy degradation.

Quantization demonstrates that reducing the precision of weights and activations can yield substantial storage savings and inference speed improvements, especially on hardware optimized for lower bit-widths. However, aggressive quantization can negatively impact accuracy, particularly in sensitive layers. Knowledge Distillation effectively enables smaller models to retain performance by learning from larger teacher models, offering a balance between model size and predictive accuracy.

The findings indicate that the choice of compression technique should align with deployment requirements, balancing memory constraints, computational efficiency, and accuracy retention. While each technique offers unique advantages, combining Pruning with Quantization provides a robust solution for efficient model deployment on limited-resource devices. For high-accuracy applications, Knowledge Distillation with Quantization serves as an effective approach, particularly for edge or mobile environments. This study underscores the importance of compression techniques in advancing deep learning for real-world applications, enabling models to run efficiently in diverse and resource-constrained settings.

6. Contributions

- **Muhammad Saad Haroon:** Completed Task 1 + Report.
- **Jawad Saeed:** Completed Task 2 + Report.
- **Daanish Uddin Khan:** Completed Task 3 and Task 4

+ Report.

References

1. All assignment references provided in the course material.

2. NVIDIA Developer Blog: ”Accelerating Sparse Matrix Multiplication for Deep Learning.” This resource provides insights into the specialized sparse matrix optimizations for NVIDIA hardware, supporting the benefits of unstructured pruning on sparse-optimized hardware. Available online.

3. Gale, T., Elsen, E., and Hooker, S. ”The State of Sparsity in Deep Neural Networks.” 2019. This paper discusses the trade-offs in hardware requirements and efficiency for structured vs. unstructured pruning, highlighting the need for specialized hardware in certain cases. Available on arXiv.