Efficient Knowledge Distillation: Custom Teacher-Student Models for Resource-Constrained Environments

Joanna Lin [1] **Sikai Shen** [1]

Abstract

Recent advances in deep learning have led to models capable of unprecedented accuracy but often at the cost of increased computational complexity, making deployment challenging, especially on resource-constrained devices.
This project seeks to develop a lightweight model that is restricted to fewer than 12000 parameters, requires less than 2 hours of training, and achieves inference times of less than 0.05 seconds per batch, ensuring efficient training and deployment on resource-constrained edge devices. Utilizing a knowledge distillation framework, our methodology begins by training several models within designated time intervals. We then integrate various teacher-student combinations from our initial training phases as well as selected pretrained models. In our experiments, the teacher-student models outperform our baseline smaller student model by 30%. This project improves the understanding of knowledge distillation mechanisms and offers practical feasibility of deploying high performance models in resource-constrained environments.

1. Introduction

Recent advancements in the field of deep learning have created an era of highly accurate models spanning diverse domains. However, a critical investigation from papers such as "Scaling laws for neural language models" underscores a fundamental trade-off: the pursuit of increased accuracy often results in heightened computational complexity. Consequently, deploying these sophisticated models on resource-constrained devices poses a formidable challenge. Overcoming this necessitates the development of lightweight models capable of operating efficiently within stringent resource constraints while achieving high accuracy.

Knowledge distillation (KD) emerges as a well-established technique designed to alleviate the computational burden associated with inference tasks. At its core, KD involves the training of a compact "student" model by distilling knowledge from a larger and more complex "teacher" model. Unlike alternative approaches such as quantization and pruning, KD offers a unique advantage in its adaptability to specific application requirements and constraints.

In this paper, we explored various distillation methods tailored to specific resource constraints, aiming to elucidate the discrepancies in accuracy and loss between offline and online distillation approaches. Furthermore, we leverage the vision transformer pretrained model as a benchmark to assess performance, particularly in scenarios involving extremely complex and high-accuracy teacher models. Through our investigation, we endeavor to shed light on the efficacy of knowledge distillation in optimizing model performance under resource constraints.

2. Related Work

2.1. Knowledge Distillation

The concept of knowledge distillation (KD) as a method to compress the knowledge of a larger, more complex model into a smaller, more efficient model was first formalized by (Hinton et al., 2015). They introduced the basic framework where the "teacher" model's soft outputs (logits) are used to train a "student" model, which significantly improved the student's performance by transferring the representational knowledge of the teacher. This foundational work has inspired numerous advancements in the field of model compression and efficient computing.

In knowledge distillation, a small student model is generally supervised by a large teacher model (Ba & Caruana, 2014; Hinton et al., 2015). Subsequent research (Chen et al., 2017; Urban et al., 2017; Hu et al., 2022) has explored various dimensions of KD, focusing on different distillation techniques, architectures, and applications. For instance, the work by (Hu et al., 2022) provides a comprehensive survey of teacher-student architectures, emphasizing the diversity in approaches to harness a teacher's knowledge, including offline and online distillation methods. These methods differ primarily in whether the teacher model is static during the

[1]Cornell University, Ithaca.

student's training (offline) or if both models are trained simultaneously (online), which can influence the effectiveness and efficiency of knowledge transfer.

Another significant aspect of KD research focuses on the operational environments where these models are deployed. With the increasing deployment of deep learning models on resource-constrained devices, such as mobile phones and embedded systems, the need for lightweight models has never been more critical. (Zhang et al., 2019) address this by introducing self-distillation techniques where a model acts as both the teacher and student, iteratively refining its capabilities. This method simplifies the distillation process by eliminating the need for a separate teacher model, thereby reducing the computational overhead.

3. Methodology

3.1. Knowledge Distillation with Offline Distillation

In the standard approach to knowledge distillation (Hinton et al., 2015; Kim et al., 2018) , knowledge is transferred from a pre-trained teacher model to a student model. This process unfolds in two distinct stages: 1) the larger teacher model is initially trained on a dataset prior to distillation; and 2) knowledge is extracted from the teacher model in the form of logits or intermediate features, which then guide the training of the student model during the distillation phase. The primary benefit of offline methods lies in their simplicity and ease of implementation. For instance, the teacher model could comprise a collection of models that have been trained using various software packages, potentially across different machines. The knowledge from these models can be extracted and preserved in a cache.

Due to the limited training resource and time, we have designed our own neural network architecture for the teacher-student model in following structure.

3.1.1. Student

The Student model starts with an initialization that takes parameters for the number of input channels, output channels, the number of classification classes , and the dropout rate. The network architecture comprises two main sections: the convolutional layers and the classifier. The first section contains two convolutional layers, each consisting of a 2D convolution with a kernel size of 3x3 and padding of 1 to maintain spatial dimensions, followed by a LeakyReLU activation function for non-linear processing and helping prevent the dying neuron problem, and a Dropout for regularization. After processing through these layers, the data undergoes global average pooling. The classifier section is made up of fully connected layers: an initial linear transformation that maps pooled features into a 30-dimensional space, a LeakyReLU activation, a dropout based on the specified rate, and a final linear layer that classifies the features into the desired number of classes. This setup, particularly with its use of dropout at multiple stages, is designed to combat overfitting, making the model robust for training on diverse image datasets.

Table 1. Student time and memory costs for 1 batch (100 data) on CPU

Operation	Time total (ms)	Mem (Mb)
model inference	124.773	0
create tensor	306.000us	50.18
conv2d	57.069	25.00
dropout	45.907	50.02
leaky relu	3.705	25.01
dataloading	14.777	1.17

Based on the profile of the model, the majority of time was spent on convolution layers and the dropout layer. Significantly, the total convolution layers took around 45% of the inference CPU time while the total dropout took around 36%. On the side of memory usage, dropout layer dominated among the operations, reaching around twice of convolution layers and leaky relu layers.

3.1.2. Teacher

The Teacher model has a configurable number of convolutional layers, allowing an exploration of the impact of model complexity on the training of corresponding Student models. Initiated with parameters for input channels, output channels, number of convolutional layers, classification classes, and dropout rate, the model begins with a standard convolutional layer followed by a LeakyReLU activation and a dropout. The convolutional layers are dynamically created based on number of convolutional layers, each composed of a sequence that includes a convolution (without bias for simplification), LeakyReLU for non-linear processing, batch normalization, and a dropout. The output of these layers is then processed through global average pooling. The classifier is identical to that of the Student model, consisting of two linear transformations—the first mapping to 30 dimensions followed by a dropout and LeakyReLU, and the final mapping to the number of classes. This configuration not only aids in reducing overfitting but also allows for detailed examination of how the size and depth of the teacher model influence the learning and performance of corresponding student models.

Based on the profile of the teacher 4 model, we can see a significant increase in the inference time as this model contains more complex structures. Percentage-wise, the teacher 4 model stayed consistent with the student model with convolution layers taking up around 48% CPU time and dropout layers taking up around 36%. Interestingly,

Table 2. Teacher 4 time and memory costs for 1 batch (100 data) on CPU

OPERATION	TIME TOTAL (MS)	MEM (MB)
MODEL INFERENCE	275.721	0
CREATE TENSOR	165.000US	162.71
CONV2D	132.866	62.50
DROPOUT	99.021	125.02
LEAKY RELU	20.030	62.51
BATCH NORM	18.906	50.00
DATALOADING	11.968	1.17

Table 3. Teacher 6 time and memory costs for 1 batch (100 data) on CPU

OPERATION	TIME TOTAL (MS)	MEM (MB)
MODEL INFERENCE	437.044	0
CREATE TENSOR	470.000US	250.20
CONV2D	199.713	87.50
DROPOUT	169.428	175.02
LEAKY RELU	14.149	87.51
BATCH NORM	25.248	75.00
DATALOADING	17.719	1.17

while the parameter size of the teacher 3 model is around triple of the student model, the time and memory cost both only increase by twice.

3.1.3. MODEL SIZE

The chart outlines the number of trainable parameters in the Student model and three variations of the Teacher model, differentiated by the number of convolutional layers they contain, as indicated by the notation Teacher (X), where X represents the number of convolutional layers in addition to the initial layer present in all models.

The Student model is the simplest, with 11,444 parameters, suitable for faster training and lesser computational demands but with limited capacity for handling complex patterns in the data. As the complexity of the models increases with the addition of more convolutional layers, so does the number of parameters. Specifically, the Teacher (4) model, which includes 4 additional convolutional layers to the base configuration, houses 39,316 parameters. This is followed by the Teacher (6) with 6 additional layers totaling 57,876 parameters, and the Teacher (8) with 8 additional layers, totaling 76,436 parameters. Each additional convolutional layer in the Teacher models includes its own set of filters, activation functions (LeakyReLU), batch normalization, and dropout components, which cumulatively increase the model's parameter count.

Table 4. Teacher 8 time and memory costs for 1 batch (100 data) on CPU

OPERATION	TIME TOTAL (MS)	MEM (MB)
MODEL INFERENCE	594.629	0
CREATE TENSOR	435.000US	312.69
CONV2D	293.220	112.50
DROPOUT	211.366	225.02
LEAKY RELU	15.319	112.51
BATCH NORM	37.660	100.00
DATALOADING	13.316	1.17

Table 5. Number of trainable parameters for each neutral network

MODEL	PARAMETER SIZE
STUDENT	11,444
TEACHER 4	39,316
TEACHER 6	57,876
TEACHER 8	76,436
VIT PRETRAINED	85,806,346

3.2. Knowledge Distillation with Online Distillation

Online distillation has been introduced to enhance the performance of the student model, particularly in situations where a large-capacity, high-performance teacher model is unavailable. In online distillation, both the teacher and the student models are updated simultaneously, making the entire knowledge distillation framework trainable in an end-to-end manner. Any one network can be the student model and other models can be the teacher during the training process.

In this project, we'll also explore the performance difference and training efficiency between offline and online distillation for our teacher-student model.

3.3. Knowledge Distillation with Pretrained Model

Vision transformer has been introduced to leverage the transformer architecture and was often found to surpass the result of the traditional CNN models in image classification tasks. In this method, we utilize a vision transformer that was fine-tuned on the CIFAR-10 dataset to take on the role of the teacher. This approach uses an even more complex model of 85,806,346 parameters with accuracy of 0.9788 but suffers from inference time of 0.127 seconds per image, which is is more than 200 times that of the student model.

Without modifications to the knowledge distillation model, we would inference from the teacher model at each epoch to produce soft labels and calculate the KL divergence loss. However, this leads to many unnecessary computations when the weights of the pretrained model never changes and each epoch step takes more than an hour given the

size of the dataset. Hence, to mitigate this, we instead inference from the teacher model prior to training. Now, at each epoch, we would instead load the soft labels that were already stored as data.

4. Experiments and Results

4.1. Training Assumption

The training of the models adheres to specific assumptions designed to ensure consistent and comparable results across different training sessions. Firstly, the training dataset comprises 45,000 samples, while validation is performed on 5,000 samples, providing a substantial volume for training and a sufficient quantity for performance evaluation. The batch size is set at 100, and the data is shuffled randomly for each epoch to promote model generalization and prevent learning biases associated with the order of data. Each model undergoes training over 100 epochs on the same computational device to ensure that comparisons of training time are fair and unaffected by hardware variations.

The AdamW optimizer, known for its effectiveness in deep learning by adjusting weights in a way that considers both momentum and the root mean square of the gradients, is used in its default configuration, ensuring that the optimization process is robust and efficient. The loss function employed is a combination of KL-divergence and cross-entropy loss. The KL-divergence loss measures how one probability distribution diverges from a second, expected probability distribution (in this case, the teacher's output compared to the student's), and is normalized by a temperature parameter to adjust the sensitivity of the outputs. The cross-entropy loss quantifies the difference between the true labels and the predictions made by the student model. The default loss use the arithmetic mean of the KL-divergence loss and cross-entropy loss.

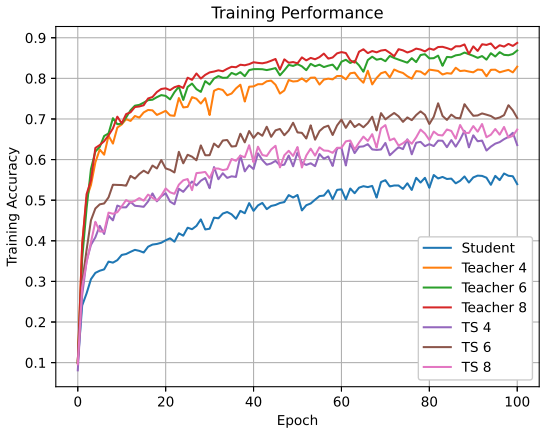

Figure 1. Performance of each model on training set

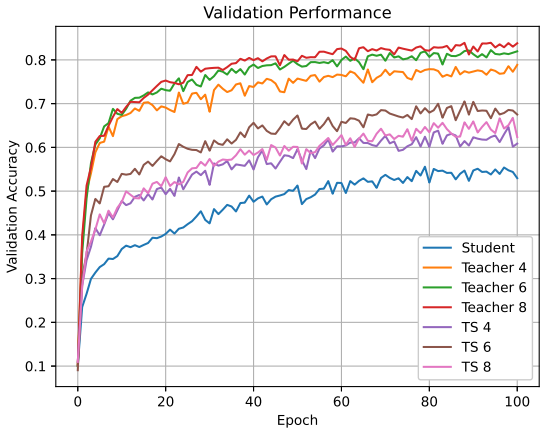

Figure 2. Performance of each model on training set

4.2. Experiment 1: Configurable Teacher Models

The experiment comparing the performance of various teacher-student models with standalone student and teacher models offers insightful observations on the influence of model complexity on knowledge distillation.

4.2.1. Metric 1: Accuracy

From the figure 1,2, we observe that the standalone Student model has the lowest performance, with accuracy significantly increasing in teacher models as additional convolutional layers are added (Teacher 4, 6, and 8). This suggests that more complex teacher models can better capture and process features from the training data.

However, the performance of the teacher-student models (TS 4, TS 6, TS 8) distilled from these teachers does not align directly with the complexity of the teacher models. Notably, the teacher-student model distilled from the Teacher 6 model (TS 6) achieves the highest accuracy among the distilled models, surpassing those distilled from both the less complex Teacher 4 and the more complex Teacher 8. This could be attributed to an optimal balance in the "model gap" between the teacher and the student. The largest teacher model, Teacher 8, does not translate to the highest performance in its distilled student (TS 8), potentially due to challenges in effectively transferring the complex patterns learned by the teacher to the simpler student model. This suggests that a middle ground in teacher complexity, as seen with Teacher 6, might offer the best conditions for effective knowledge distillation to a less complex student model.

Table 6. Model Performance After 100 epochs

MODEL	TRAINING ACC	VALIDATION ACC
STUDENT	53.95	52.96
TEACHER 4	82.88	78.86
TEACHER 6	86.87	81.96
TEACHER 8	88.74	83.80
TS 4	63.57	60.86
TS 6	70.25	67.51
TS 8	67.4	62.35

Table 7. Training Efficiency and Inference Speed (second/epoch)

MODEL	TRAINING	INFERENCE
STUDENT	62.59	13.88
TEACHER 4	213.54	37.13
TEACHER 6	321.31	60.22
TEACHER 8	429.12	80.93
TS 4	104.53	13.56
TS 6	125.81	13.94
TS 8	146.43	13.91

4.2.2. METRIC 2: TRAINING EFFICIENCY AND INFERENCE SPEED

Table 7 has shown all the information about training time and inference time for one epoch of data, meansured in one second wall clock time.

Training time per epoch increases substantially with the complexity of the model. The standalone Student model is the fastest, requiring only 62.59 seconds per epoch. This is expected due to its simpler architecture with fewer parameters. On the other hand, teacher models show a progressive increase in training times corresponding to their complexity: Teacher 4 requires 213.54 seconds, Teacher 6 needs 321.31 seconds, and Teacher 8 takes the longest at 429.12 seconds per epoch. This pattern is indicative of the higher computational demand due to the increased number of layers and parameters, which require more time for both forward and backward passes during training.

The standalone Student model, with the shortest training time of 62.59 seconds per epoch, operates independently of any additional inference overhead. In comparison, the teacher-student models not only perform their own training computations but also include the inference time of their respective teachers to derive the necessary predictions for distillation. For instance, consider the actual training times reported: TS4 is 104.53 seconds per epoch, TS6 is 125.81 seconds, and TS8 is 146.43 seconds. When factoring in the additional inference times of Teacher 4 (37.13 seconds), Teacher 6 (60.22 seconds), and Teacher 8 (80.93 seconds), the effective training overhead for distillation isn't significantly more than the standalone training time of these models.

The teacher-student models all have similar inference times close to that of the standalone student model. TS4, TS6, and TS8 have inference times of 13.56, 13.94, and 13.91 seconds per epoch, respectively. This suggests that while training these models involves a more complex setup due to the distillation process, the final models are streamlined enough to offer quick inference.

The overall efficiency in training and inference times highlights a crucial trade-off between model complexity and operational efficiency. While more complex models can achieve higher accuracy, as seen in the teacher models, their increased computational demands can be a limiting factor, especially in resource-constrained environments. Teacher-student models offer a middle ground, improving upon the simplicity of student models in terms of accuracy while maintaining a reasonable computational footprint that approaches the standalone student model in inference speed. This balance makes them attractive for scenarios where both performance and efficiency are required.

4.3. Experiment 2: Optimizing the loss function

The loss function we need have 2 hyperparameters: α, which refers to the weights of loss functions and temperature T. In this experiment, I want to optimize the model performance by optimizing these 2 hyperparameters through grid search.

$$\text{loss} = \alpha \cdot \text{KL}\left(\text{softmax}\left(\frac{y_s}{T}\right), \text{softmax}\left(\frac{y_t}{T}\right)\right) \cdot T^2 + (1-\alpha) \cdot \text{CEL}(y_s, y)$$

4.3.1. OPTIMIZE THE LOSS FUNCTION WEIGHTS

In this experiment, the performance of a teacher-student model distilled from the Teacher 6 model was optimized by tuning two hyperparameters in the loss function: the balance factor α and the temperature T, with T fixed at 1. The results of varying α from 0 to 1 demonstrate the effect of balancing the two components of the loss function: KL-divergence and cross-entropy loss.

The graph of training performance shows that the model achieves optimal training accuracy at $\alpha = 0.7$, with a peak accuracy of approximately 72.03%. This suggests that at this value, the blend of knowledge distillation via KL divergence and direct learning from the ground truth via cross-entropy loss is most effective for this specific model configuration and dataset. Notably, the model performs comparably well for α values between 0.5 and 0.8, indicating a robust range where the influence of the teacher model positively impacts learning without overshadowing the direct training from actual data. When $\alpha = 0$, the model essentially reverts to being a standard student model without distillation, relying solely on cross-entropy loss for learning, which is

reflected in its lower performance compared to when distillation is incorporated ($\alpha > 0$). This experiment clearly illustrates the critical role of carefully tuning distillation and task-specific learning components to maximize a model's performance.

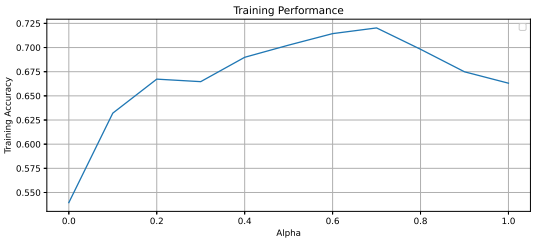

Figure 3. Model Performance with different balance factors

4.3.2. Optimize the Temperature

In a further exploration to optimize the distillation process, the hyperparameter temperature T was tested while keeping the balance factor α fixed at 0.5. The temperature parameter plays a critical role in softening the outputs of the teacher model during the calculation of the KL-divergence part of the loss function. The experiment evaluated the effect of varying T over a wide range, represented in a logarithmic scale from $T = 0.1$ to $T = 100$.

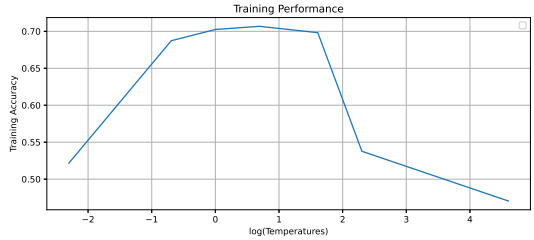

Figure 4. Model Performance with different temperature

The results, as depicted in the figure 4, show that the training accuracy of the teacher-student model remains relatively stable within the temperature range from 0.5 to 5, achieving peak performance around these values. This suggests that within this range, the softening effect provided by the temperature is optimal for transferring knowledge from the teacher to the student model without excessively diluting the information or making it too peaky, which would be less informative.

Outside this optimal range, particularly for temperatures higher than 5 and lower than 0.5, the model performance declines sharply. At very high temperatures, the softmax function approaches a uniform distribution, thereby reducing the effectiveness of the distillation as the soft labels become increasingly non-informative. Conversely, at very low temperatures, the softmax function becomes too peaky, which can exacerbate overfitting to the teacher's specific outputs rather than learning the general patterns.

This experiment highlights the importance of carefully tuning the temperature parameter in knowledge distillation setups to maintain a balance between preserving informative signals in the teacher's outputs and preventing the loss function from favoring overly confident, less-generalizable student predictions. Such tuning is crucial for maximizing the efficacy of the knowledge transfer from teacher to student model.

4.4. Experiment 3: Performance of Online Distillation

In this experiment, we want to see how the performance and training time will change if we apply online distillation rather than offline distillation.

4.4.1. Metric 1: Model Accuracy

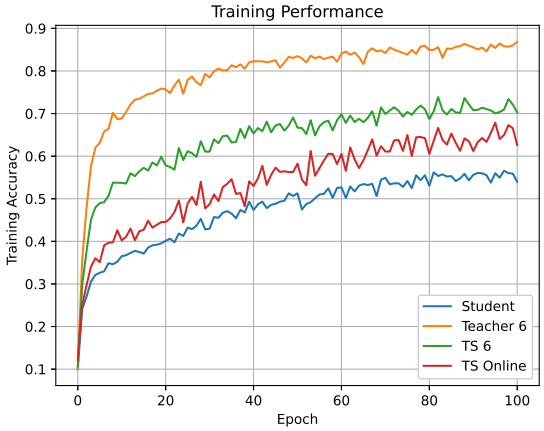

Figure 5. Model Performance with online and offline distillation

The graph illustrates the comparison between two distillation approaches for teacher-student models: offline distillation (TS 6) and online distillation (TS Online). The standalone Student model, as expected, shows the lowest performance due to its simpler architecture. The Teacher 6 model, with its complex structure, consistently achieves the highest training accuracy.

In terms of the teacher-student models, TS 6 (offline distillation) demonstrates superior performance over TS Online (online distillation). This outcome aligns with typical expectations in the field of knowledge distillation. Offline distillation benefits from a stable and fully-trained teacher model, which provides a consistent and optimized target

for the student model to emulate. The knowledge extracted from this static teacher is generally more refined and beneficial for the student's learning process.

Conversely, TS Online, which involves the student model learning concurrently with the ongoing training of the teacher model, shows lower performance. This method might be affected by the teacher model's initial instability and gradual progression toward optimal performance, which could lead to less effective teaching signals being passed to the student model during the early and possibly more volatile phases of training.

Thus, the experiment underscores that while online distillation offers the advantage of dynamic learning and potential adaptability, it may not always provide the robustness or consistency in performance achieved through offline distillation, where the quality and stability of the teacher's knowledge are assured. This finding could guide the choice of distillation techniques based on the specific requirements and constraints of training scenarios.

4.4.2. Metric 2: Training Time

The training efficiency of offline and online distillation methods presents a nuanced picture when considering their wall clock times and the potential for optimization in implementation. Currently, the online distillation method demonstrates a training efficiency edge with a total wall clock time of 11.7 hours, compared to 12.42 hours required for offline distillation. This advantage in the online method is primarily due to the concurrent training of both the teacher and student models, which, despite the need for constant adjustment and feedback between the two, seems to streamline the process overall.

However, it's important to consider the nature of the data handling in these models. In the existing setup, each model needs to perform inference on the training data within each epoch due to the shuffling of the training set. This requirement inherently increases the computation time per epoch as each model must recompute the outputs for the entire dataset.

With a more optimized implementation, where both the teacher and student models only need to perform inference once per training cycle, significant efficiencies could be realized. This change would effectively equalize the training times for both online and offline distillation methods, as each would only need to compute the teacher's outputs once, rather than repeatedly per epoch. Such an optimization would not only simplify the training process but also reduce the computational overhead, potentially bringing the training times of the online and offline methods much closer together. Thus, while the current scenario shows a slight efficiency edge for online distillation, with proper implementation adjustments, both methods could achieve similar efficiencies in training duration.

4.5. Experiment 4: Pretrained model knowledge distillation

In this experiment, we want to see how the performance and training time will change when there exists a high gap between the number of parameters and the model accuracy between the teacher and the student.

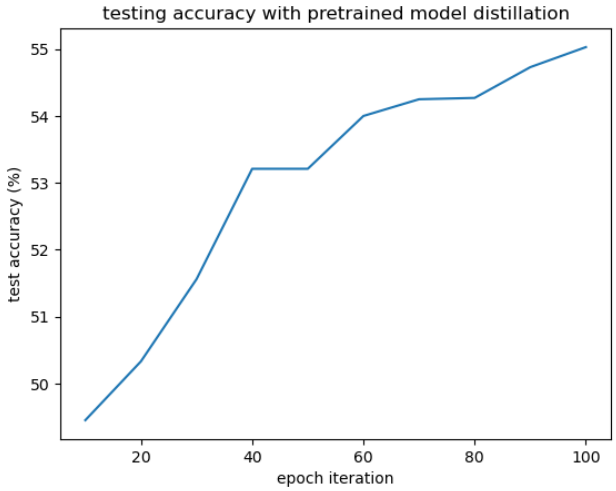

Figure 6. Performance of student model

In our experiment using a pretrained Vision Transformer (ViT) model as the teacher, the performance of the student model achieves an of accuracy 53.82% (starting from 21% accuracy). The pretrained model approach was less effective compared to our custom teacher-student models. The custom teacher-student models achieved higher accuracy improvements while maintaining efficiency. The substantial difference in performance might be attributed to the significant capacity gap between the pretrained ViT model and the student model. Moreover, the training time for the student model with the pretrained teacher was nearly eight times longer than that of our custom teacher-student setup, taking 862 seconds per epoch compared to much shorter times for the custom models. This significant increase in training time represents a considerable waste of computational resources, underscoring that our custom-designed teacher-student models are more effective and efficient for knowledge distillation in resource-constrained environments.

The design of the teacher-student architecture is crucial in distillation learning, as it directly influences the efficiency and effectiveness of the knowledge transfer process. A well-matched teacher-student pair ensures that the student model can effectively learn from the teacher without being

overwhelmed by the complexity or the sheer number of parameters of the teacher model. This balance is essential for achieving optimal performance improvements while maintaining computational efficiency. In our experiments, we observed that custom-designed teacher-student models with moderate complexity provided the best results. The custom models distilled from a teacher with a balanced architecture not only outperformed the standalone student model but also did so with significantly lower training times and resource requirements compared to models distilled from an excessively large pretrained teacher. This highlights the importance of carefully designing the teacher-student architecture to align with the specific constraints and objectives of the deployment environment, ensuring that the benefits of distillation learning are fully realized without incurring unnecessary computational costs.

5. Conclusion

In this project, we explored the development and optimization of lightweight deep learning models using knowledge distillation techniques, aimed at efficient deployment on resource-constrained edge devices. Our primary goal was to create a student model with fewer than 12,000 parameters, training times under 2 hours, and inference times less than 0.05 seconds per batch, all while maintaining high accuracy.

We began by implementing various knowledge distillation methodologies, including traditional teacher-student setups, online distillation and the use of pretrained models such as the Vision Transformer (ViT). Our experiments were designed to compare these methods in terms of accuracy improvement, training efficiency, and computational resource requirements.

Our findings indicate that custom-designed teacher-student models provide a significant performance boost over standalone student models. Specifically, the teacher-student models improved the baseline student's accuracy by approximately 30%, showcasing the effectiveness of the distillation process. Among the custom models, those distilled from a teacher model with moderate complexity achieved the highest accuracy, highlighting the importance of selecting an appropriately balanced teacher-student pair. The Teacher 6 model, in particular, emerged as the optimal choice, providing a harmonious balance between model complexity and performance gains for the student.

In contrast, the use of a pretrained Vision Transformer (ViT) as the teacher model, despite its high accuracy and complexity, did not yield proportional benefits for the student model. The significant capacity gap between the ViT and the student model likely contributed to this discrepancy. Moreover, the training time for the student model with the pretrained ViT teacher was nearly eight times longer than that of our custom teacher-student models, taking 862 seconds per epoch. This substantial increase in training time underscores the inefficiency and resource wastage associated with using excessively large pretrained models for distillation in resource-constrained environments.

Additionally, our experiments with online distillation revealed that while this method offers dynamic learning and adaptability, it did not achieve the same level of performance consistency as offline distillation. The stable and fully-trained teacher in offline distillation provided a more reliable and refined target for the student model, leading to better overall performance.

Our work emphasizes the critical role of teacher-student architecture design in the knowledge distillation process. A well-matched teacher-student pair ensures effective knowledge transfer without overwhelming the student model, facilitating optimal performance improvements and computational efficiency. This balance is essential for practical applications, particularly in environments with stringent resource constraints.

In conclusion, our project demonstrates that custom teacher-student models can significantly enhance the performance of lightweight student models, making them viable for deployment on edge devices with limited computational resources. The insights gained from this research contribute to a deeper understanding of knowledge distillation mechanisms and offer practical guidelines for developing high-performance, resource-efficient models. Future work could further investigate the scalability of these methods across different datasets and more complex tasks, continuing to refine the balance between model complexity and deployment feasibility.

6. Supplementary Materials

Link to code repository

References

Ba, J. and Caruana, R. Do deep nets really need to be deep? 2014.

Chen, G., Choi, W., Yu, X., Han, T., and Chandraker, M. Learning efficient object detection models with knowledge distillation. 2017.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. 2015.

Hu, C., Li, X., Liu, D., Chen, X., Wang, J., and Liu, X. Teacher-student architecture for knowledge learning: A survey. 2022.

Kim, J., Park, S., and Kwak, N. Paraphrasing complex network: Network compression via factor transfer. 2018.

Urban, G., Geras, K. J., Kahou, S. E., Aslan, O., Wang, S., Caruana, R., Mohamed, A., Philipose, M., and Richardson, M. Do deep convolutional nets really need to be deep and convolutional? 2017.

Zhang, L., Song, J., Gao, A., Chen, J., Bao, C., and Ma, K. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. 2019.