EasyChair Preprint

№ 15433

Distillation-Based Model Compression Framework for Swin Transformer

Mazen Amria, Aziz M. Qaroush, Mohammad Jubran, Alaa Zuhd and Ahmad Khatib

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

November 16, 2024

Distillation-based Model Compression Framework for Swin Transformer

Mazen Amria
Department of Electrical and Computer Engineering
Birzeit University
Ramallah, Palestine
mramria@birzeit.edu

Aziz M. Qaroush
Department of Electrical and Computer Engineering
Birzeit University
Ramallah, Palestine
aqaroush@birzeit.edu

Mohammad Jubran
Department of Electrical and Computer Engineering
Birzeit University
Ramallah, Palestine
mjubran@birzeit.edu

Alaa Zuhd
Department of Electrical and Computer Engineering
Birzeit University
Ramallah, Palestine
azuhd@birzeit.edu

Ahmad Khatib
Department of Electrical and Computer Engineering
Birzeit University
Ramallah, Palestine
1182828@student.birzeit.edu

***Abstract*—Vision Transformers (ViTs) have gained significant attention in computer vision due to their exceptional model capabilities. However, most ViT models suffer from high complexity, with a large number of parameters that demand considerable memory and inference time, limiting their applicability on resource-constrained devices. To address this issue, we propose a distillation-based framework for compressing large models for smaller datasets. The framework leverages fine-tuning and knowledge distillation to accelerate the training process of compressed models. To evaluate its effectiveness, two compressed Swin Transformer models, Swin-N and Swin-M, were introduced and tested on the CIFAR-100 dataset. Experimental results demonstrate that when trained using the proposed framework, both Swin-N and Swin-M exhibit significant improvements in accuracy compared to their counterparts trained from scratch, with Swin-N achieving an 18.89% increase and Swin-M showing a 20.10% improvement. Additionally, Swin-M closely approximates the accuracy of the Swin-T teacher model, further validating the effectiveness of the framework.**

***Index Terms*—ViT, Swin, model compression, knowledge distillation, fine-tuning**

I. Introduction

Vision Transformers (ViTs), inspired by the success of transformer-based architectures in Natural Language Processing (NLP), have become a leading paradigm in Computer Vision (CV), surpassing Convolutional Neural Networks (CNNs) in numerous applications. Despite their impressive performance, ViTs tend to have a large number of parameters to achieve better results, which in turn increases training and inference time, as well as the required space when deployed. For example, the Swin Transformer [1] employs between 28 million and 197 million parameters (depending on the version) and is pre-trained on either ImageNet-1K (over 1 million images) or ImageNet-22K (over 14 million images), yielding strong results in downstream detection and segmentation tasks. However, the extensive computational resources required for ViTs pose challenges for deployment on low-end devices, such as IoT and mobile devices, affecting their real-time responsiveness in sensitive applications [2], [3]. To address these challenges, recent research has focused on enhancing ViTs through compression techniques. These efforts aim to reduce the model size and increase throughput, making ViTs more suitable for deployment in resource-constrained environments. This work seeks to downsize vision transformers, focusing on compressing large models for use in simpler tasks on resource-constrained devices. Specifically, it explores how to effectively transfer knowledge from large-scale transformers to smaller models and leverage extensive data to enhance the representational capabilities of these compressed models.

Swin Transformer has multiple variations, ranging from the smallest model, Swin-T (Tiny), to the largest, Swin-L (Large). Although the model is pre-trained on the large ImageNet-1K dataset, fine-tuning it on smaller datasets like CIFAR-100 often results in a performant yet redundant model. This work introduces a framework for compressing large models for smaller datasets, with Swin Transformer on CIFAR-100 serving as a test case. The framework leverages fine-tuning and knowledge distillation to accelerate the training process of compressed models. To evaluate the framework, we introduced two compressed Swin transformer models designed for smaller tasks, Swin-N and Swin-M. The primary contributions of this paper are as follows:

1) Introducing a framework to accelerate the training of smaller Swin transformers on simpler tasks.
2) Developing smaller versions of Swin transformers with fewer parameters and higher throughput.
3) Demonstrating model redundancy in Swin-T when fine-tuned on simpler tasks like CIFAR-100, as the smaller model, Swin-M, was able to approximate its performance.
4) Proposing joint loss weights scheduling (α-scheduling)

and harmonic mean as mechanisms to dynamically control the contribution of each loss component in knowledge distillation.

The structure of this paper is as follows: Section 2 reviews related works. Section 3 details the proposed framework. Section 4 presents the experimental results and analysis. Finally, Section 5 concludes the paper and suggests potential directions for future research.

II. Related Work

ViTs have demonstrated impressive performance in computer vision tasks, but their large model size and computational demands limit their practicality on resource-constrained devices. To address these challenges, this section explores four primary techniques for compressing ViTs: Efficient Architectures, Pruning, and Knowledge Distillation [2]–[4].

A. Efficient Architectures

Efficient Architectures focus on designing ViT models that are inherently smaller and more computationally efficient [5]–[7]. This involves optimizing the number of layers, attention heads, and feature dimensions to reduce model complexity without sacrificing performance.

Hierarchical models enhance the efficiency of ViTs by structuring visual processing into multiple levels, emulating the human visual system, which starts with coarse features and refines details. A significant advantage of these models is their optimization of the self-attention mechanism. Traditional self-attention processes an $N \times N$ matrix, resulting in quadratic complexity. In contrast, hierarchical self-attention partitions this matrix into smaller $\frac{N}{k} \times \frac{N}{k}$ blocks, calculating attention within each block separately, thus reducing complexity to linear and significantly improving efficiency.

The Swin Transformer exemplifies a hierarchical ViT, processing images in progressively smaller windows at each layer to capture local information. It employs shifted windows and hierarchical token merging to capture long-distance patterns effectively. This balance of efficiency and performance makes it suitable for resource-constrained environments. Figure 1 illustrates the architecture of the Swin Transformer, specifically Swin-T.

To address the computational limitations of standard ViTs, researchers have focused on developing lightweight models such as MobileViT [8] and LVT [9]. These models aim to reduce the model size and complexity while maintaining performance. MobileViT, a lightweight ViT that incorporates convolutional-like operations, was introduced in [8]. The MobileViT block consists of components similar to convolutional layers, including unfolding, folding, and matrix multiplication. By replacing the matrix multiplication with a stack of transformer layers, MobileViT combines the strengths of CNNs and ViTs. This innovative design enables MobileViT to outperform both CNN-based and ViT-based models across various tasks and datasets. LVT is a lightweight ViT that integrates two types of self-attention mechanisms [9]. It incorporates a convolutional layer to enhance low-level features and a recursive atrous self-attention layer to increase representational capacity.

B. Pruning

Pruning is a critical technique for optimizing ViTs by reducing size and computational complexity while preserving accuracy. This method involves selectively removing redundant components—such as weights, attention heads, or entire layers—to streamline ViTs, enhancing efficiency for deployment in resource-constrained environments. Pruning can be categorized into two types: unstructured and structured pruning [10]–[13].

Unstructured pruning selectively zeroes out individual weights, particularly those with minimal magnitudes, to reduce model size without altering its architecture [14]. This approach exploits sparsity by limiting computations to non-zero elements, leading to substantial reductions in computational cost. However, it may cause irregular computation patterns that can be inefficient on some hardware, limiting practical benefits [10].

Structured pruning removes entire groups of weights, such as attention heads or layers, streamlining the model for better hardware implementation [14], [15]. This method is more compatible with hardware, as it removes contiguous matrix blocks, improving efficiency. However, structured pruning generally achieves a lower compression ratio than unstructured pruning, as fewer zero elements are bypassed, which may limit model size reduction [10].

C. Knowledge distillation

Knowledge distillation is a powerful technique for compressing ViTs by transferring knowledge from a larger, more complex teacher model to a smaller, more efficient student model. This process enables the student model to achieve comparable performance with significantly reduced computational resources [16], [17]. This process is done by optimizing the loss between the teacher's and the student's outputs. The loss between the student's output and the hard labels is jointly optimized as well to ensure the correctness of the output. Equation 1 shows the equation of the joint loss where L is the total loss, L_d is the distillation loss, L_s is the hard-label loss, and α is the weight of the student loss.

$$L = (1 - \alpha) \cdot L_d + \alpha \cdot L_s \tag{1}$$

The Early Exit technique proposed in [18] involves training the model with multiple output points at different layers, creating a form of knowledge distillation where earlier exits serve as student models and later exits as teacher models. Knowledge is transferred through back-propagation, with a weighted sum of losses used to address branching paths.

Knowledge distillation enhances the learning capabilities of the smaller model by guiding it with knowledge from the larger model. However, the effectiveness of this technique can be limited by the knowledge transfer methods employed.

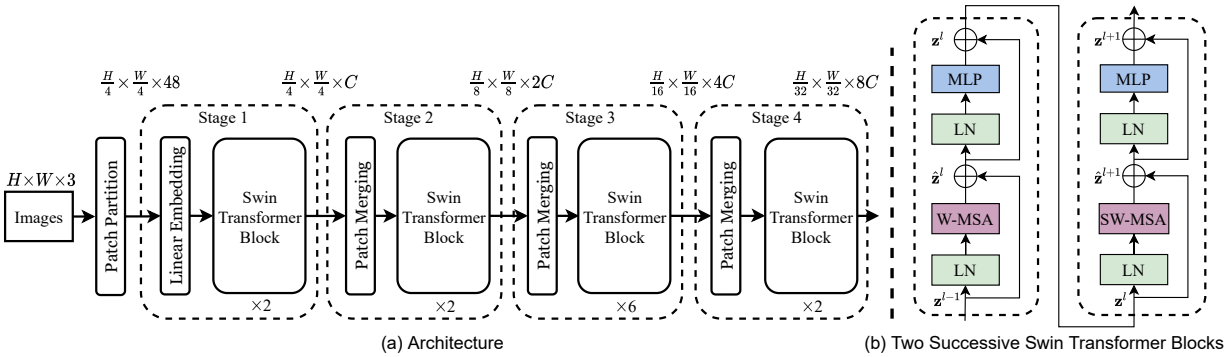


Fig. 1: Architecture diagram for Swin-T [1].

Careful selection and implementation of knowledge distillation techniques are crucial for achieving optimal results [19].

III. Proposed Work

In this paper, we propose a distillation-based framework to accelerate the training of smaller transformers on simpler tasks. To evaluate the framework, we introduced two compressed Swin transformer models, named *Swin-N* and *Swin-M*. Table I outlines the architectural differences between our models and the standard Swin models. The primary variation across the models lies in the depth of the third layer, specifically the number of blocks. Following this trend, our models feature depths of 2 and 4 in the third layer for Swin-N and Swin-M, respectively, compared to a depth of 6 in Swin-T. We utilized the CIFAR-100 dataset to test the framework.

TABLE I: Architecture of the existing and proposed models.

Model	Depths	Number of Heads
Swin-N	$[2, 2, 2, 2]$	$[3, 6, 12, 24]$
Swin-M	$[2, 2, 4, 2]$	$[3, 6, 12, 24]$
Swin-T	$[2, 2, 6, 2]$	$[3, 6, 12, 24]$
Swin-S	$[2, 2, 18, 2]$	$[3, 6, 12, 24]$
Swin-B	$[2, 2, 18, 2]$	$[4, 8, 16, 32]$

Our framework begins by fine-tuning a pre-trained large model on the target dataset, which in this case involves fine-tuning Swin-T on CIFAR-100. Next, the earlier blocks are transferred to the smaller model, specifically copying the first 6 blocks from the fine-tuned Swin-T to Swin-N. Finally, knowledge distillation is applied, where the fine-tuned large model (Swin-T) serves as the teacher and the smaller model (Swin-N) acts as the student. Figure 2 illustrates the process of compressing Swin-T into Swin-N for CIFAR-100. Algorithm 1 shows the steps of the framework.

The fine-tuned model can be divided into two parts: $\mathcal{F}(x)$, which represents the portion before the first cut point, and $\mathcal{G}(z)$, which is the remainder of the model. The full fine-tuned model can be expressed as $\mathcal{G} \circ \mathcal{F}$. Our framework suggests using the part after the second cut point, $\hat{\mathcal{G}}$, to approximate $\mathcal{G}$. Successfully approximating $\mathcal{G}$ means compressing it into $\hat{\mathcal{G}}$, resulting in a compressed model represented as $\hat{\mathcal{G}} \circ \mathcal{F}$. Therefore, the difference between the output of the compressed

Algorithm 1: Compressing a Swin transformer T into a smaller Swin transformer S.

Input: Pre-trained model T, Dataset, cut points p_1, p_2, loss weights w_d, w_s
$T \leftarrow$ FineTune(T, Dataset);
$S \leftarrow$ CreateNetworkLike(Compose($T[:p_1]$, $T[p_2]$));
$S[:p_1] \leftarrow$ CopyWeights($T[:p_1]$);
for *each batch* (X, Y) *in Dataset* **do**
  $O_S, O_T \leftarrow$ Forward(S, X), Forward(T, X);
  $L_d \leftarrow$ DistillationLoss(O_S, O_T);
  $L_s \leftarrow$ HardLabelLoss(O_S, Y);
  $L \leftarrow$ HarmonicMean(L_d, L_s, w_d, w_s);
  $S \leftarrow$ UpdateWeights(S, L);
end

model and that of the original model is *initially* driven by the gap between $\mathcal{G}$ and $\hat{\mathcal{G}}$. Through knowledge distillation, the outputs of both models are aligned, gradually forcing $\hat{\mathcal{G}}$ to approximate $\mathcal{G}$.

Finally, we proposed using the *Harmonic Mean* for the knowledge distillation joint loss, as described in Equation 2, instead of the arithmetic mean.

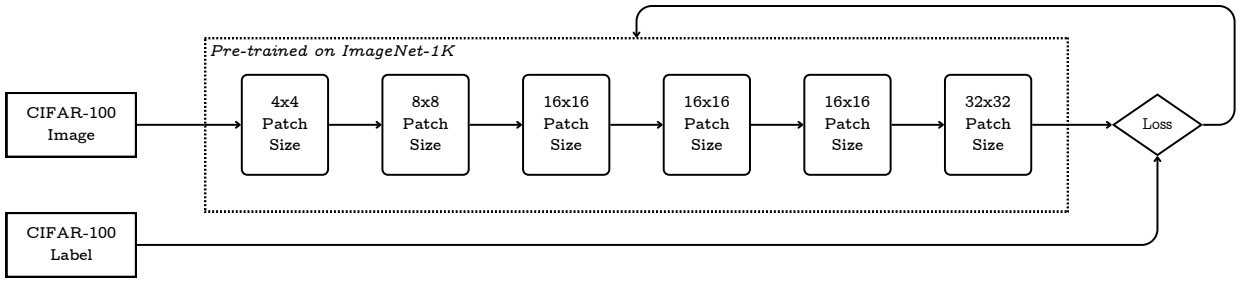
$$L(t) = \frac{2}{\frac{1}{L_d(t)} + \frac{1}{L_s(t)}} \tag{2}$$

where L is the total loss, L_d is the distillation loss, and L_s is the hard-label loss. The arithmetic mean can cause instability because a loss with a significantly larger magnitude may dominate the backpropagation process. In contrast, the harmonic mean provided more stable learning curves and better overall performance.

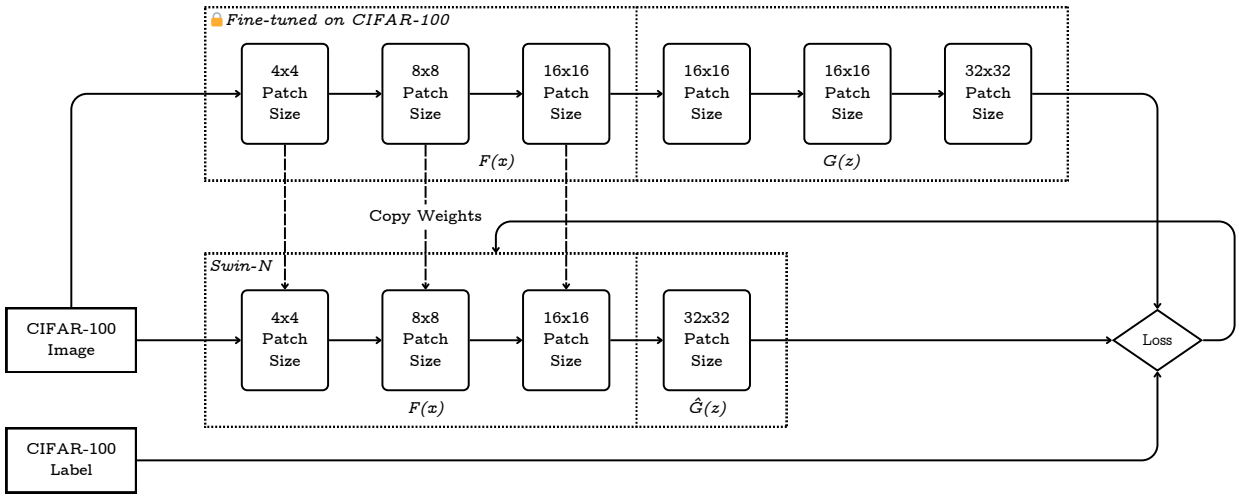
IV. Experiments and Results

To evaluate the framework, we first assess how each technique contributes to accelerating the training of the smaller models. Next, we examine the impact of combining these techniques on training speed. Finally, we compare all the results.

For a specific model, such as Swin-N, all experiments share the same model configurations, training algorithm, and



(a) Fine-tuning Swin-T (pre-trained originally on ImageNet-1K) on CIFAR-100.



(b) Copying the weights of the earlier blocks of Swin-T to Swin-N and using knowledge distillation to the Swin-N output distribution to Swin-T output distribution.

Fig. 2: Compressing Swin-T into Swin-N using the proposed framework.

parameters, including the number of epochs. Therefore, if a particular technique achieves higher performance within the same number of epochs compared to another, we can conclude that this technique *accelerates* the training process.

A. Baseline Models

First, we trained the smaller models on CIFAR-100 *from scratch* to establish a baseline for comparison. We followed the configurations provided by [20], which include using the AdamW optimizer with a weight decay factor of 0.01, a cosine learning rate scheduler with 5 warmup epochs, and an initial learning rate of 0.0002. These configurations were also applied to train the other models. Since our primary goal is to evaluate how the framework accelerates model training, we limited training to 60 epochs to observe how much progress can be made within such many epochs. Table II presents the *top-1 accuracy* scores for the baseline models.

TABLE II: Accuracy of the baseline models.

Model	Accuracy
Swin-N	66.32%
Swin-M	67.04%

B. Knowledge Distillation

Each of the smaller models is subsequently trained using knowledge distillation. To implement knowledge distillation, three parameters need to be configured:

1) **Teacher Model**
The teacher model is a pre-trained model used to compute the distillation joint loss. For this purpose, we fine-tuned three Swin models originally trained on ImageNet-1K on CIFAR-100. These models are Swin-T, Swin-S, and Swin-B. Table III displays the top-1 accuracy scores for these models both before fine-tuning (on ImageNet-1K) and after fine-tuning on CIFAR-100.

TABLE III: Accuracy of the Fine-tuned models.

Model	Original Accuracy	Fine-tuned Accuracy
Swin-T	81.2%	87.35%
Swin-S	83.2%	89.38%
Swin-B	83.5%	92.01%

2) **SoftMax temperature** τ
This parameter smoothes the SoftMax output, as the output in well-trained models may be sharp enough to look like a hard label, whereas, in knowledge distillation, we're interested in learning the distribution of the

teacher model [21]. In theory, [22] proved that when applying the Kullback-Leibler divergence loss (KLDiv), the attention moves from the label matching to the logits matching as τ grows, and vice versa. Furthermore, the same study found that when logits matching is used, performance improves. Furthermore, it was proved in that work that the mean squared error (MSE) loss surpasses the Kullback-Leibler divergence loss as τ approaches infinity. Based on these results, MSE should be used instead of KLDiv, eliminating the need to tune τ.

3) **Student loss weight in the joint loss α**
We determined the value of α through tuning. Specifically, we tested five values for α, uniformly distributed over the range $[0, 1]$. These values were 0.0 (training only with the teacher model), 0.25, 0.5, and 0.75, with 1.0 representing standard training without distillation (baseline).

Each model was trained with every combination of α and teacher model, resulting in a total of 24 different experiments. Table IV presents the results of these knowledge distillation experiments. Due to limited computational resources, the experiments were conducted for only 30 epochs instead of 60. Since our primary focus is on training speed, 30 epochs were sufficient to indicate which configuration accelerates training the most.

TABLE IV: Accuracy of the distilled models after training them for half runs.

Model	α	Swin-T	Swin-S	Swin-B
Swin-N	0	70.06%	67.69%	68.78%
	0.25	**71.74%**	69.29%	68.60%
	0.5	70.27%	67.33%	67.67%
	0.75	68.64%	66.49%	66.98%
Swin-M	0	71.06%	70.11%	69.33%
	0.25	**72.56%**	69.21%	69.64%
	0.5	70.27%	68.51%	68.87%
	0.75	68.30%	67.24%	67.54%

The results in Table IV show that using smaller teacher models, which are more similar in complexity to the compressed models, yields better performance than using larger models. Additionally, a larger gap between the teacher and student models tends to decrease the effectiveness of knowledge transfer, as the student model struggles more to learn the distribution of the teacher model due to the capacity mismatch [19], [23]. Furthermore, smaller values of α generally led to better performance, although a value of 0 resulted in poorer performance because the student model no longer benefited from the hard label loss. The results in Table IV suggest that using Swin-T as the teacher model and setting α to 0.25 is optimal for both compressed models. Table V shows the results of using these configurations and training the models for the full duration. Comparing the results with those in Table II, it is evident that knowledge distillation has accelerated the training process. The student models achieved higher performance within the same number of epochs, with accuracy improvements of $\mathbf{8.98}\%$ and $\mathbf{8.91}\%$.

TABLE V: Accuracy of the distilled models after full runs on tuned parameters.

Model	Accuracy
Swin-N	75.30% (+8.98%)
Swin-M	75.95% (+8.91%)

C. Scheduling α

The parameter α controls the contribution of each loss component based on its importance. It can also be used to adjust the magnitudes of the components, as the two different loss functions might be scaled differently. Figure 3 illustrates the behaviour of each loss component throughout the training process. Each curve represents a separate training run for that specific loss component alone, without the joint loss, to eliminate any interaction between the behaviours of the two components.

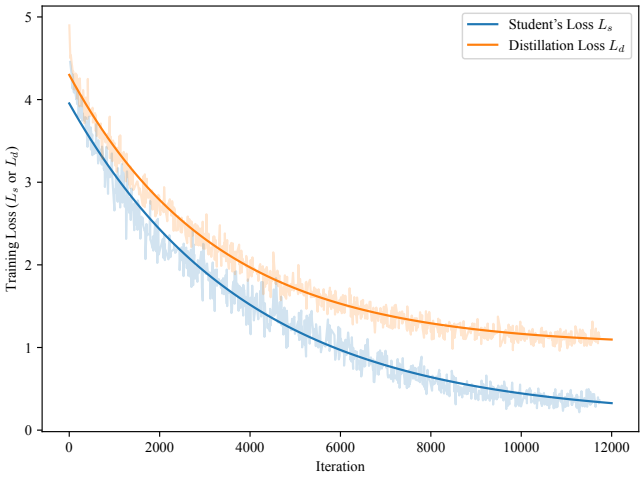


Fig. 3: Student and Distillation losses in Swin-N.

The figure indicates that the student loss magnitude decreases more rapidly than the distillation loss. To balance the magnitudes over time, a dynamic value for α, based on the ratio between the distillation loss and the total loss, can be utilized. Formally, α can be defined as:

$$\alpha(t) = \frac{L_d(t)}{L_s(t) + L_d(t)} \tag{3}$$

However, this approximation of α is highly sensitive to noise in the loss curves. To address this, two exponential functions ($A_s e^{B_s t} + C_s$ and $A_d e^{B_d t} + C_d$) were used to fit the loss curves. The approximation of α is then calculated based on these fitted curves. Figure 4 displays the ratio plot.

The figure shows that the approximation of α behaves almost like a linear function of time. Consequently, a linear scheduler for α was tested. Although the plotted range for α is $[0.50, 0.75]$, various ranges were explored to account for the interaction between the loss components. Table VI presents the results of these experiments on Swin-N.

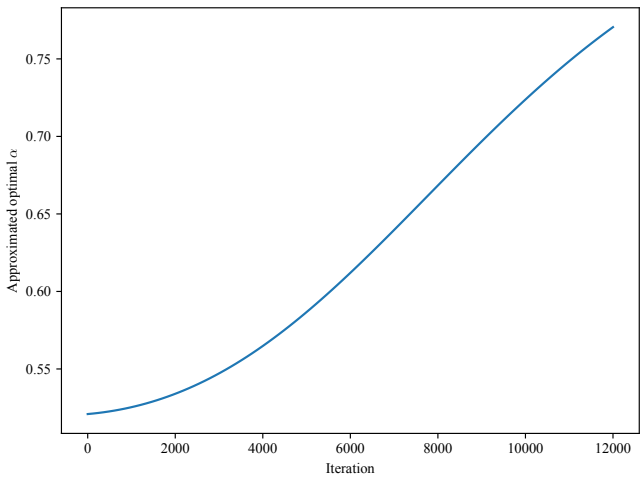


Fig. 4: Distillation Loss to Joint Loss Ratio.

TABLE VI: Accuracy of Scheduling α Linearly over different intervals.

α **Interval**	**Accuracy**
$[0.5, 1.0]$	72.83% $(+6.51\%)$
$[0.0, 1.0]$	74.45% $(+8.13\%)$
$[\mathbf{0.0}, \mathbf{0.5}]$	$\mathbf{75.43\%}$ $(+\mathbf{9.11\%})$

D. Harmonic Mean

Returning to the approximation of α, substituting it into the joint loss defined by Equation 1 yields the harmonic mean joint loss function, defined by Equation 2. This approach can be beneficial compared to using the arithmetic mean, as it mitigates the impact of mismatched magnitudes while still allowing for weighting based on the importance of each component. Equation 4 shows how weighting can be applied to the harmonic mean joint loss, where w_d is the weight of the distillation loss L_d and w_s is the weight of the hard-label loss L_s.

$$L(t) = \frac{w_d + w_s}{\frac{w_d}{L_d(t)} + \frac{w_s}{L_s(t)}} \tag{4}$$

Table VII shows the results of tuning the weights for the harmonic mean joint loss, specifically for **Swin-N** with Swin-T as the teacher model. **Swin-M** was also trained using the tuned configurations of Swin-N, which involved the harmonic mean with w_d set to 13. The highest accuracy achieved by the model was $\mathbf{77.02\%}$ which is an improvement of $\mathbf{9.98\%}$ over the baseline model.

E. Fine-tuning

In this experiment, the weights from the trained Swin-T are used for the smaller models instead of initializing them randomly. The weights before the cut point are retained, as they are typically trained to extract features and encode the input image. However, the weights after the cut point need to be trained to accommodate different inputs. Table VIII presents the results after applying fine-tuning.

TABLE VII: Accuracy of using different sets of weights in Weighted Harmonic Mean.

L_d **Weight** (w_d)	L_s **Weight** (w_s)	**Accuracy**
1	1	69.62% $(+3.30\%)$
2	1	71.17% $(+4.85\%)$
3	1	72.85% $(+6.53\%)$
5	1	73.87% $(+7.55\%)$
10	1	75.27% $(+8.95\%)$
12	1	75.82% $(+9.50\%)$
13	**1**	$\mathbf{76.13\%}$ $(+\mathbf{9.81\%})$
15	1	75.65% $(+9.33\%)$

TABLE VIII: Accuracy of the Fine-tuned models (i.e., preserving blocks' weights from the larger model).

Model	**Accuracy**
Swin-N	82.71% $(+16.93\%)$
Swin-M	85.10% $(+18.06\%)$

Fine-tuning has significantly accelerated the training of the smaller models. This improvement arises because the models are now only adjusting the randomly initialized part $\hat{\mathcal{G}}$, and the initial parameter values are much closer to the convergence point.

F. Full framework and Comparison

Finally, both fine-tuning and knowledge distillation (using the optimal configurations) are combined. Table IX presents the results of these combined experiments, compared to all the results from previous experiments.

TABLE IX: Comparing results of normal training, knowledge distillation, fine-tuning, and a combination of fine-tuning and knowledge distillation.

Model	**Initialization**	**Distillation**	**Accuracy**
Swin-N	Random Initialization	No	66.32%
	Random Initialization	Yes	76.13%
	Fine-tuning	No	82.71%
	Fine-tuning	**Yes**	$\mathbf{85.21\%}$
Swin-M	Random Initialization	No	67.04%
	Random Initialization	Yes	77.02%
	Fine-tuning	No	85.10%
	Fine-tuning	**Yes**	$\mathbf{87.14\%}$

The results demonstrate that combining fine-tuning with knowledge distillation further enhanced the training speed, as the accuracy improvements over the baseline model were $\mathbf{18.89\%}$ and $\mathbf{20.10\%}$ for Swin-N and Swin-M respectively. This allowed the models to achieve performance close to that of the teacher model. In particular, Swin-M reached an accuracy of $\mathbf{87.14\%}$, while the fine-tuned Swin-T achieved $\mathbf{87.35\%}$ on CIFAR-100. This suggests that, for smaller tasks, certain layers in the model may be redundant, and the model can be compressed effectively by removing these layers,

retaining the weights of the earlier layers, and learning the distribution of the original (pre-compression) model.

G. Models Size

Additionally, this section provides an intuition of the size of the proposed models and the effectiveness of the compression. Table X shows the throughput and the number of parameters in each model on CIFAR-100.

TABLE X: Performance on CIFAR-100.

Model	Throughput (img/s)	Num. Of Params.	Accuracy
Swin-N	573.8	$20.5M$	85.21%
Swin-M	494.5	$24.0M$	87.14%
Swin-T	422.5	$27.6M$	87.35%
Swin-S	246.5	$48.9M$	89.38%
Swin-B	165.3	$86.9M$	92.01%

In Table X, Swin-N shows a significant improvement in throughput (573.8 img/s) compared to Swin-T, although its accuracy (85.21%) is lower. This suggests that either the optimal size for the smallest Swin model in solving CIFAR-100 lies between Swin-N and Swin-M, or that Swin-N may benefit from additional training epochs. Since our focus is on accelerating training, we aimed to achieve accuracy improvements within the same number of epochs, rather than focusing solely on maximizing the model's performance. Swin-M, while offering a smaller gain in throughput (494.5 img/s), is able to closely match Swin-T's accuracy (87.14% vs. 87.35%).

V. Conclusion

This paper presents a novel framework that significantly accelerates the training of compressed ViTs, particularly Swin Transformers, for resource-constrained environments. By leveraging knowledge distillation and fine-tuning techniques, we effectively expedite the training process of Swin-N and Swin-M models, achieving substantial performance gains on the CIFAR-100 dataset.

Our framework introduces innovative techniques, such as joint loss weight scheduling and harmonic mean, to dynamically control the knowledge distillation process and balance the contribution of each loss component.

We demonstrate the effectiveness of our approach by comparing the performance of our trained models to those trained from scratch. Our results show that Swin-M achieves comparable accuracy to the larger Swin-T model after only 60 epochs of training. Moreover, both Swin-N and Swin-M exhibit significant improvements in accuracy compared to their counterparts trained from scratch, with Swin-N achieving a $\mathbf{18.89}\%$ increase in accuracy and Swin-M showing a $\mathbf{20.10}\%$ improvement.

For future work, we plan to explore the use of skip connections with joint optimization as an alternative to model duplication. This approach aims to reduce memory usage during training by maintaining a single model in memory while still producing multiple outputs. By jointly optimizing these outputs, the shorter path can approximate the output of the full path, resulting in an internally compressed model.

Acknowledgment

To enhance readability and language quality, all sections of this paper were grammatically revised and edited using the OpenAI ChatGPT system. The authors then reviewed and edited all content to ensure accuracy and alignment with our research objectives.

References

[1] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," 2021.

[2] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in Vision: A Survey," *ACM Comput. Surv.*, vol. 54, sep 2022.

[3] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A Survey on Vision Transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 87–110, 2023.

[4] L. Papa, P. Russo, I. Amerini, and L. Zhou, "A Survey on Efficient Vision Transformers: Algorithms, Techniques, and Performance Benchmarking," *IEEE transactions on pattern analysis and machine intelligence*, vol. PP, 2023.

[5] H. Cai, J. Li, M. Hu, C. Gan, and S. Han, "EfficientViT: Multi-Scale Linear Attention for High-Resolution Dense Prediction," 2024.

[6] H. You, Y. Xiong, X. Dai, B. Wu, P. Zhang, H. Fan, P. Vajda, and Y. C. Lin, "Castling-ViT: Compressing Self-Attention via Switching Towards Linear-Angular Attention at Vision Transformer Inference," 2024.

[7] Y. Li, C.-Y. Wu, H. Fan, K. Mangalam, B. Xiong, J. Malik, and C. Feichtenhofer, "MViTv2: Improved Multiscale Vision Transformers for Classification and Detection," 2022.

[8] S. Mehta and M. Rastegari, "MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer," 2022.

[9] C. Yang, Y. Wang, J. Zhang, H. Zhang, Z. Wei, Z. Lin, and A. Yuille, "Lite Vision Transformer with Enhanced Self-Attention," 2021.

[10] A. Kumar, "Vision Transformer Compression with Structured Pruning and Low Rank Approximation," *arXiv preprint arXiv:2203.13444*, 2022.

[11] L. Yu and W. Xiang, "X-Pruner: eXplainable Pruning for Vision Transformers," 2023.

[12] H. Yang, H. Yin, M. Shen, P. Molchanov, H. Li, and J. Kautz, "Global Vision Transformer Pruning with Hessian-Aware Saliency," 2023.

[13] M. Zhu, Y. Tang, and K. Han, "Vision Transformer Pruning," 2021.

[14] T. Chen, Y. Cheng, Z. Gan, L. Yuan, L. Zhang, and Z. Wang, "Chasing Sparsity in Vision Transformers: An End-to-End Exploration," 2021.

[15] Z. Lin, J. Z. Liu, Z. Yang, N. Hua, and D. Roth, "Pruning Redundant Mappings in Transformer Models via Spectral-Normalized Identity Prior," 2021.

[16] H. Lin, G. Han, J. Ma, S. Huang, X. Lin, and S.-F. Chang, "Supervised Masked Knowledge Distillation for Few-Shot Transformers," 2023.

[17] S. Ren, Z. Gao, T. Hua, Z. Xue, Y. Tian, S. He, and H. Zhao, "Co-advise: Cross Inductive Bias Distillation," 2021.

[18] S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks," *CoRR*, vol. abs/1709.01686, 2017.

[19] J. H. Cho and B. Hariharan, "On the Efficacy of Knowledge Distillation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4794–4802, 2019.

[20] K. Wu, J. Zhang, H. Peng, M. Liu, B. Xiao, J. Fu, and L. Yuan, "TinyViT: Fast Pretraining Distillation for Small Vision Transformers," 2022.

[21] G. Hinton, O. Vinyals, J. Dean, *et al.*, "Distilling the Knowledge in a Neural Network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[22] T. Kim, J. Oh, N. Kim, S. Cho, and S.-Y. Yun, "Comparing Kullback-Leibler Divergence and Mean Squared Error Loss in Knowledge Distillation," 2021.

[23] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved Knowledge Distillation via Teacher Assistant," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34(04), pp. 5191–5198, 2020.