Efficient Vision Transformers: Once-Train Optimization

Akshay Paralikar
New York University
NYC, USA
ap8235@nyu.edu

Rugved Mhatre
New York University
NYC, USA
rrm9598@nyu.edu

***Abstract*—Designing accurate and efficient Vision Transformers for diverse hardware platforms, particularly resource-constrained edge devices, is a significant challenge. Traditional approaches rely on either manual design or neural architecture search to discover specialized architectures, often requiring computationally expensive retraining for each scenario. In this work, we propose a once-trained (OT) network framework that decouples training from the search process, enabling efficient inference across a wide range of architectural settings. The OT network is trained once and allows rapid extraction of specialized sub-networks without the need for additional training. To achieve efficient training, we leverage Flash Attention, reducing overhead while maintaining high accuracy. Additionally, we design a NAS pipeline that incorporates attention head pruning, teacher-student distillation, and quantization techniques. This approach produces ViTs that meet specific hardware and latency constraints while preserving accuracy comparable to independently trained models. Our method significantly reduces the computational cost of deploying ViTs across heterogeneous devices, paving the way for scalable and efficient vision transformer solutions. github.com/rugvedmhatre/Efficient-ViT**

***Index Terms*—ViT, Flash Attention, Pruning, Quantization, Teacher-student Distillation, Neural Architecture Search**

I. Introduction

Vision Transformers (ViTs) [1] have achieved state-of-the-art accuracy in many image classification tasks, demonstrating their transformative potential in deep learning. However, their rapidly increasing model size and computational demands present significant challenges for efficient deployment across diverse hardware platforms. These platforms, ranging from high-end devices with dedicated neural accelerators to older devices with limited processing power, impose varying efficiency constraints such as latency and energy consumption.

To address these challenges, researchers have explored two primary approaches: designing compact, specialized models for specific scenarios [2], [3] or compressing existing models to improve efficiency [4], [5]. While effective to some extent, both methods come with significant drawbacks. Human-designed models and neural architecture search (NAS)-based solutions require repeated network design and retraining for each deployment scenario, incurring high computational and energy costs. As the number of deployment scenarios increases, the total cost scales linearly, making these methods unsustainable for large-scale or diverse applications.

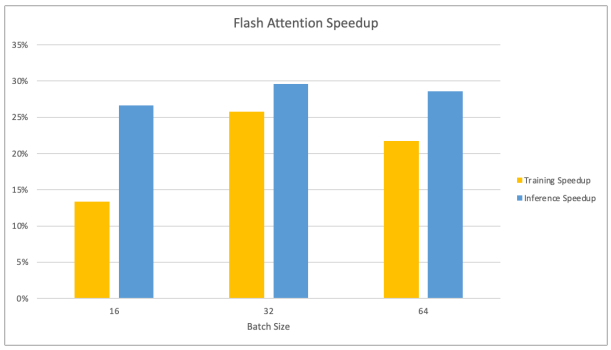

Fig. 1. Efficient-ViT Speedup compared to Baseline (T4 GPU)

Inspired by the Once-for-All paper for convolutional neural neworks [6], in this project we propose a once-trained (OT) ViT network capable of supporting diverse architectural configurations without the need for retraining. By decoupling the training phase from the neural architecture search phase, our approach amortizes training costs across multiple deployment scenarios. During the training phase, we aim to improve the accuracy of the OT network and use efficient training techniques such as Flash Attention [7] to reduce computational overhead. In the post-training phase, we employ attention head pruning [8], teacher-student distillation [9], and quantization [4] techniques to generate sub-networks tailored to specific hardware and latency constraints, while maintaining accuracy comparable to independently trained models. A lightweight NAS [10] is then used to select the optimal sub-network for a given deployment scenario, incurring negligible additional cost.

This approach significantly reduces the energy and computational demands of designing specialized neural networks, enabling scalable and efficient deployment of ViTs across a wide range of devices and scenarios.

II. Literature Review

Vision Transformers have emerged as a powerful alternative to convolutional neural networks, achieving state-of-the-art performance across numerous image classification tasks. However, their extensive computational and memory demands present challenges for deployment on resource-constrained devices such as edge hardware and mobile processors.

A. Efficient ViT Architectures

To address these limitations, researchers have explored two main strategies:

- **Designing Compact Models:** Approaches like MobileNetV2 [2] and ShuffleNet [3] introduce lightweight architectures tailored for low-power devices.
- **Model Compression:** Techniques such as pruning, quantization, and knowledge distillation have been applied to reduce model size and computational cost while maintaining acceptable accuracy [4], [5].

Despite their effectiveness, these methods often require repeated retraining for specific deployment scenarios, significantly increasing the computational overhead.

B. Once-Train Optimization Framework

The Once-for-All (OFA) paradigm [6] proposed for CNNs inspires an alternative approach to reduce this overhead. OFA decouples the training phase from the search phase, enabling the creation of a versatile network that supports diverse configurations without retraining. Applying this concept to ViTs offers the potential for scalable optimization across hardware platforms. This approach aligns with techniques like Flash Attention [7], which minimizes memory overhead during training, and structural pruning strategies such as DepGraph [8].

C. Post-Training Optimization Techniques

- **Pruning:** Selectively removes redundant components like attention heads to reduce computation without significantly compromising accuracy [8].
- **Knowledge Distillation:** Transfers knowledge from a large, accurate teacher model to a smaller student model, achieving efficiency gains with minimal accuracy loss [9].
- **Quantization:** Converts model weights to lower precision (e.g., INT8) to improve efficiency while preserving model structure [4].

D. Neural Architecture Search (NAS)

NAS frameworks [10] leverage automated exploration to discover optimal sub-network configurations tailored to deployment-specific constraints, such as latency or memory. While traditional NAS methods incur high computational costs, lightweight variants can integrate seamlessly with post-training techniques to identify efficient architectures with minimal overhead.

This body of work highlights the need for frameworks capable of efficiently tailoring ViTs for diverse scenarios, bridging the gap between high performance and practical deployability. The current project builds on these ideas, introducing an OT network pipeline incorporating Flash Attention, pruning, distillation, quantization, and NAS to optimize ViTs for real-world applications.

III. Methodology

A. Dataset

For training and evaluating our OT network, we chose the CIFAR-100 dataset [11], a widely recognized benchmark in computer vision research. CIFAR-100 comprises 60,000 32×32 color images categorized into 100 fine-grained classes, organized under 20 superclasses. Its fine-grained classification task adds complexity due to subtle inter-class differences, such as distinguishing between maple and oak trees.

CIFAR-100 offers several advantages: it provides a standardized benchmark for comparing with state-of-the-art methods, challenges model generalization across diverse categories, and mirrors real-world tasks requiring fine-grained distinctions. Despite being smaller than datasets like ImageNet [12], its complexity makes it an effective proxy for larger-scale applications. Its manageable size also facilitates efficient experimentation, enabling rapid iteration and refinement of training techniques without incurring high computational costs.

We pre-processed the images by standardizing them and applied data augmentation techniques such as random cropping, horizontal flipping, and cutout regularization to enhance robustness. Overall, CIFAR-100 serves as an ideal testbed for evaluating the adaptability and accuracy of our OT network, with future plans to scale the method to larger datasets like ImageNet.

B. Model Selection

For our OT network, we selected the ViT-Base [1] model as the backbone architecture. The ViT-Base model, with approximately 86 million parameters, represents a balanced trade-off between model complexity and computational efficiency, making it well-suited for our study.

ViT-Base has demonstrated strong performance on various image classification tasks, including achieving competitive top-3 accuracy (92%) on the CIFAR-100 dataset. Its design incorporates 12 transformer layers, 12 attention heads, and a sequence length of 256 (16×16 patches), making it capable of learning intricate patterns and relationships in image data.

The choice of ViT-Base was motivated by several factors. First, it strikes a practical balance between scalability and performance. With its 86 million parameters, the model is large enough to capture the complexity of the dataset but still manageable for training and deployment across diverse hardware platforms. Second, ViT-Base's patch-based image representation, where input images are divided into non-overlapping patches (16×16), allows for efficient processing while maintaining spatial information. This property is particularly useful when tailoring sub-networks for deployment scenarios with varying resource constraints.

Additionally, ViT-Base has demonstrated compatibility with optimization techniques such as quantization and attention head pruning, which are central to our specialization pipeline. These techniques, combined with the model's inherent architectural flexibility, enable us to efficiently generate sub-networks that meet specific hardware and latency constraints without significant loss of accuracy.

C. Pre-Training Optimizations

The training phase of our OT network incorporates advanced optimizations to enhance efficiency and scalability. A

central component of these optimizations is the use of Flash Attention [7], a state-of-the-art attention mechanism specifically designed to minimize memory overhead and improve computational efficiency in Transformers.

Flash Attention redefines the way attention is computed within the transformer architecture by restructuring the attention computation process to reduce memory usage significantly. Traditional attention mechanisms can incur substantial memory costs, particularly when working with large models like ViT-Base, as they involve storing intermediate results for backpropagation. Flash Attention addresses this challenge by adopting a memory-efficient approach that computes attention directly without the need to store redundant intermediate values. This not only minimizes memory consumption but also speeds up the overall attention process, making it particularly well-suited for training large-scale ViTs.

By integrating Flash Attention during the pre-training stage, we successfully reduce the computational burden typically associated with training ViTs while maintaining the model's ability to learn complex patterns and relationships in the data. This optimization allows us to accelerate training times without sacrificing accuracy, enabling us to handle the high parameter count of ViT-Base effectively.

To quantify the benefits of Flash Attention, we conducted a comprehensive evaluation of our model's training and inference times. Using PyTorch profiling tools, we compared the training performance of our Flash Attention-enabled ViT-Base model (Efficient-ViT) with a baseline ViT-Base model that does not incorporate Flash Attention. This comparison highlighted the speedup achieved through the optimized attention mechanism and demonstrated its impact on both computational efficiency and memory usage.

These training optimizations were instrumental in ensuring that our OT network could be trained efficiently, reducing the overall computational cost of the methodology. By enabling faster and more memory-efficient training, the optimizations not only facilitated the development of our model but also set the stage for its seamless adaptation to diverse deployment scenarios.

D. Post-Training Optimizations

After successfully training our OT network with the proposed optimizations and achieving competitive accuracy on the CIFAR-100 dataset, we save the model weights and proceed with post-training optimizations. These optimizations are critical for tailoring the trained network to meet specific hardware constraints and deployment scenarios, especially for resource-constrained edge devices. The primary focus during this phase is applying neural architecture search and optimization techniques such as pruning, quantization, and teacher-student distillation to balance efficiency and performance.

1) ***Pruning****:* Pruning is the first post-training optimization method we apply to reduce the computational complexity of the model. Specifically, we focus on pruning attention heads in the ViT architecture based on [8]. This approach leverages the redundancy often found in multi-head attention, where certain attention heads contribute less to the overall performance of the model.

Our pruning process uses the L2 norm of grouped weights as an importance score to identify and eliminate less critical attention heads. The procedure involves the following steps: The pruning procedure involves several key steps to effectively reduce the number of attention heads while maintaining network functionality. We establish a pruning configuration using GroupNormImportance for calculating attention head importance scores and MetaPruner for managing the process. The model is then pruned iteratively, gradually reducing attention heads to allow for incremental adaptation and mitigate sudden accuracy drops. We also redefine the forward function of the ViTSelfAttention module to accommodate pruned attention heads, ensuring seamless operation post-pruning. Finally, after each pruning step, we adjust the attention head size and all-head size parameters to maintain network consistency with the reduced architecture

This iterative pruning strategy balances the trade-off between model size reduction and accuracy preservation. While more aggressive pruning can result in smaller models, it may lead to larger accuracy drops. By carefully tuning the pruning configuration and iteration process, we achieve an optimal balance that meets hardware constraints while retaining strong classification performance.

2) ***Distillation****:* As a key component of our post-training optimization pipeline, we employed knowledge distillation to transfer the learned knowledge from a larger, pre-trained ViT-Base model to a smaller and more efficient DeiT-tiny model [13]. This technique significantly reduces the network size while preserving high accuracy, making it particularly suitable for deployment on resource-constrained devices.

The distillation process utilizes the ViT-Base model, with approximately 86 million parameters, as the teacher. We train the student model using hard loss (Cross Entoropy) and soft loss (KL divergence from probabilistic outputs), enabling the student model to learn accurate classifications and grasp the subtle decision boundaries of the teacher. The DeiT-tiny model, with only 5.5 million parameters, acts as the student. This lightweight architecture is designed to dramatically lower memory and computational demands while maintaining effectiveness.

Knowledge distillation brings several benefits to the optimization pipeline. It allows the smaller DeiT-tiny model to inherit the high-performance attributes of the teacher, resulting in a substantial reduction in model size and computational overhead. This optimization makes the student model highly practical for edge-device deployment, where resource constraints are a critical factor.

In conclusion, knowledge distillation serves as an essential step in our post-training optimization strategy. It enables us to achieve a significant reduction in model complexity and size without compromising accuracy. This enhances the adaptability and efficiency of our OT network, making it suitable for diverse deployment scenarios, particularly in real-world applications with stringent hardware constraints.

*3) **Quantization**:* Quantization is a vital component of our post-training optimization strategy, aimed at reducing the memory footprint and computational requirements of the ViT model while maintaining a high level of accuracy.

Each attention layer in a ViT consists of three linear layers: the query, key, and value layers. Additionally, each ViT layer includes three other linear layers in conjunction with the attention mechanism. These linear layers are the primary targets for quantization. By converting their weights from 32-bit floating-point representations to 8-bit integers, we effectively compress the model while preserving its overall structure and operational fidelity.

We used the PyTorch's dynamic quantization was applied using the `quantization.quantize_dynamic()` function. Dynamic quantization is applied post-training and does not require calibration data, making it a straightforward and efficient approach for optimizing ViT models. This method is particularly advantageous because it reduces the computational overhead of matrix multiplications during inference without altering the model's structure or requiring any retraining.

Quantization provides several benefits in the context of model deployment, especially on resource-constrained devices. By reducing the weight precision, we achieve a significant decrease in model size, which makes the model more suitable for storage-limited environments. Additionally, the use of INT8 operations can speed up inference on hardware platforms optimized for lower-precision arithmetic, such as mobile processors and neural network accelerators.

Although quantization typically introduces a slight drop in model accuracy, the trade-off is often negligible in comparison to the advantages gained in efficiency and deployability. In our experiments, the quantized ViT models maintained robust accuracy on the CIFAR-100 dataset while achieving considerable reductions in both memory usage and inference time.

In summary, quantization is a crucial step in our post-training optimization pipeline, enhancing the efficiency of the ViT model for deployment on diverse hardware platforms. By employing Quantization, we ensure flexibility and compatibility with a wide range of deployment scenarios, making the model highly practical for real-world applications.

*4) **Neural Architecture Search**:* As the final component of our post-training optimization pipeline, we implemented a simplified Neural Architecture Search (NAS) framework to identify the most suitable sub-network configurations for deployment across various hardware platforms. The NAS framework builds upon the earlier optimization methods—pruning, distillation, and quantization—to dynamically select and refine the architecture based on deployment constraints and performance goals.

The goal of our NAS is to balance multiple objectives: model accuracy, inference latency, and memory usage, ensuring that the resulting sub-network adheres to the constraints of the target deployment environment (e.g., edge devices, mobile processors). To achieve this, the NAS pipeline includes the following steps:

1) Defining the Search Space: The search space is derived from configurations generated during pruning, distillation, and quantization. It includes variations in attention head pruning, student-teacher distillation setups, and quantization levels (e.g., full-precision vs. INT8), representing trade-offs in accuracy, memory, and latency.
2) Scoring Function: A flexible scoring formula evaluates configurations based on metrics like accuracy, latency and memory. We can weight each metrix as per our requirement. Currently we have used:

$$\text{score} = \text{accuracy}(\%) - 0.1 \cdot \text{latency} - \text{memory used (GB)}.$$

3) Optimization Constraints: NAS refines the search space by adhering to user-specified deployment thresholds like memory, latency or CPU-usage, which reduce computational overhead and accelerate convergence.
4) Search Process: Implemented as a modular framework, the NAS uses a lightweight random search that evaluates configurations without retraining, leveraging the once-trained (OT) network to minimize lead time.

The integration of NAS into our optimization pipeline provides a systematic approach to customizing ViT sub-networks for deployment. By automating the search for optimal configurations, NAS eliminates the need for manual trial-and-error experimentation, saving time and computational resources.

In our experiments, NAS successfully identified configurations that balanced accuracy, memory, and latency based on deployment-specific constraints. This represents the final refinement stage in our post-training optimization methodology. By leveraging the outputs of pruning, distillation, and quantization, NAS enables the deployment of efficient, high-performing models across diverse hardware platforms. Its flexibility and adaptability make it a crucial step for ensuring the practical utility of our OT network in real-world applications.

IV. Experimental Results

For training, we utilized two different GPUs, the NVIDIA T4 and NVIDIA A100, to explore the impact of hardware on our proposed methods. For inference testing, we focused on CPUs to simulate deployment in resource-constrained environments, aligning with our goal of optimizing models for such scenarios.

A. Pre-training Optimization

To assess the efficiency of our pre-training optimizations, we conducted detailed benchmarks on both the T4 and A100 GPUs. On the T4 GPU, we observed the training and inference times with limited memory availability, while on the A100 GPU, the larger memory capacity allowed us to test with significantly higher batch sizes. The results, summarized in Tables I and II, highlight the differences in performance across these hardware configurations.

We designed a comprehensive benchmarking process where each experiment involved 55 iterations, including 5 warm-up iterations to stabilize GPU performance and 50 additional iterations to measure steady-state training and inference

TABLE I
COMPARISON OF TRAINING AND INFERENCE TIME FOR EFFICIENT-VIT (VIT WITH FLASH ATTENTION) AND VIT-BASE ON T4 GPU

Batch Size	Training Time (ms)		Inference Time (ms)		Train Speedup	Inference Speedup
	Efficient-ViT	ViT-B	Efficient-ViT	ViT-B		
16	170.009636	192.740746	43.768541	55.416231	1.13	1.27
32	292.703498	368.063443	88.014841	114.093797	1.26	1.30
64	582.278291	708.950535	169.086962	217.473467	1.22	1.29

TABLE II
COMPARISON OF TRAINING AND INFERENCE TIME FOR EFFICIENT-VIT (VIT WITH FLASH ATTENTION) AND VIT-BASE ON A100 GPU

Batch Size	Training Time (ms)		Inference Time (ms)		Train Speedup	Inference Speedup
	Efficient-ViT	ViT-B	Efficient-ViT	ViT-B		
16	53.038760	57.209185	29.826680	29.917025	1.07	1.00
32	80.881161	94.161342	49.698082	49.633290	1.16	1.00
64	156.295521	183.497359	93.735401	93.849544	1.17	1.00
128	297.264711	353.873496	177.331204	177.867273	1.19	1.00
256	581.151282	689.016176	341.334433	341.536947	1.19	1.00

times. This approach ensured consistency and reliability in our results. To explore the relationship between batch size and efficiency, we varied the batch size across multiple runs, demonstrating the scalability of our method on hardware with differing memory capacities.

Another critical aspect of our experiments was comparing the performance of 32-bit and 16-bit floating-point precision. Flash Attention, one of the core components of our optimization strategy, was evaluated under both precision settings to determine its effect on training speed and memory usage. On the A100 GPU, Flash Attention delivered significant reductions in training time, particularly for larger batch sizes, without compromising accuracy. Even on the more constrained T4 GPU, Flash Attention enabled measurable improvements, highlighting its suitability for resource-limited hardware.

Figure 1 illustrates the speedup achieved through Flash Attention on the T4 GPU. The results demonstrate the robustness of our pre-training optimizations, which consistently improved training efficiency and inference latency. These findings underscore the adaptability of our methods to various hardware configurations, ensuring that our approach remains effective across a wide range of deployment scenarios.

TABLE III
PRUNING PERFORMANCE COMPARISON

Model	**Parameters (M)**	**Top-1 Acc. (%)**	**Top-3 Acc. (%)**
Original	85.87	81.88	93.97
Level 1	72.05	70.15	86.95
Level 2	68.95	67.26	84.76
Level 3	63.70	56.60	75.51
Level 4	53.79	23.42	38.54

B. Post-training

For each model, we calculate total latency for running inference for 10,000 test images from CIFAR-100, top-1 accuracy, top-3 accuracy and total CPU memory usage. Additionally, we run torch profiling for cpu time as well as memory.

1) Pruning: Comparison between number of parameters of the model and top-1 accuracy as well as top-3 accuracy can be seen in the Table III. These results demonstrate a trade-off between model size reduction and accuracy preservation, with more aggressive pruning leading to significant accuracy drops. As the number of parameters in the model drop, we see a drop in inference time as well as the memory occupied by the model.

TABLE IV
DISTILLATION PERFORMANCE COMPARISON

	Teacher model	**Student model**
Number of Parameters	85.87M	5.5M
Top-3 Accuracy	93.97%	92.56%
Total Latency	210 sec	77 sec
Memory used	11.13 GB	2.87 GB

2) Teacher-Student Distillation: We can see the comparison characteristics in the Table IV. The distillation process has shown promising results, with the student model achieving comparable performance to the teacher model while being significantly smaller in size. Even with 15 times less parameters, the student model gives nearly the same accuracy on the test dataset. This can be attributed to the architecture of the DEiT model which is made for the distillation with the distillation tokens.

TABLE V
QUANTIZATION PERFORMANCE COMPARISON

	Original model	**Quantized model**
Top-3 Accuracy	93.97%	91.50%
Total Latency	210 sec	201 sec
Memory used	11.13 GB	16.9 GB
Self CPU time per iteration	2.082 sec	1.169 sec
Storage Memory	327 MB	85 MB

3) Quantization: We can see the comparison of original and quantized model in the Table V.

With a minimal drop in accuracy, we see a drop in total latency and per iteration CPU time from torch profiling. The

most significant advantage of quantization is seen in storage space, where we see 75% reduction in storage space due to quantization. We see an increase in CPU utilization because of the increased overhead of calculations due to quantization.

TABLE VI
NAS PERFORMANCE COMPARISON

	Parameter Limit : 70M	**Parameter Limit : 5.3M**
Top-3 Accuracy	88.69%	78.50%
Total Latency	200 sec	89 sec
Memory used	25 GB	5.1 GB

4) Neural Architecture Search: Table VI shows the comparison between two extreme cases where we use our NAS to find the optimum model within the specified parameter limits. We can see that for a 80% drop in memory usage and 55% drop in latency, we just lose 10% of accuracy.

V. DISCUSSION

A. Pre-Training Optimizations

While Flash Attention [7] reports significant speedups, our experimental results indicate more modest gains. Upon analysis, we discovered that this discrepancy arises due to the relatively small sequence length used in our experiments. Flash Attention is designed to optimize computation for large sequence lengths by minimizing memory overhead and computation time. However, with smaller sequences, the computational workload is already minimal, and the process becomes memory-bound rather than compute-bound.

In our case, the shorter sequence lengths led to faster calculations, limiting the extent of speedup Flash Attention could achieve. For tasks with longer sequences, Flash Attention can load a large sequence into memory, perform extensive computations in one go, and store the results efficiently, leveraging its design fully. However, for our use case of image classification on the CIFAR-100 dataset, longer sequences do not align well with the model's requirements, as they fail to provide the accuracy improvements needed for this task.

This insight is valuable for future work involving ViTs with longer sequence lengths, particularly for tasks outside image classification, where larger sequences may enhance performance. Although the benefits of Flash Attention were constrained in this specific use case, our analysis highlights its potential for broader applications where sequence lengths and computational demands are higher.

B. Post-Training Optimizations

Our experiments demonstrate that post-training optimization techniques like pruning, quantization, and knowledge distillation are effective across various model architectures, including ViTs, CNNs, etc. While our methods achieved significant reductions in model size and computational requirements, there were instances where aggressive optimization led to noticeable drops in accuracy. Finding the right balance is crucial for practical applications.

As model sizes in the field of deep learning continue to grow, it's crucial to consider how well our post-training optimization techniques scale to very large models, such as those with billions of parameters. This scalability aspect presents both challenges and opportunities for further research and development. As model complexity increases, preserving accuracy while applying aggressive optimizations becomes more challenging. The intricate interactions between billions of parameters may make it harder to identify redundant or less important components without affecting overall performance. For these issues, solutions like Advanced Distillation Techniques, Adaptive Quantization, Hardware-Aware Optimizations could be tried and analyzed. By addressing these challenges and exploring these potential solutions, we can work towards extending the applicability of our post-training optimization techniques to the increasingly large models that are becoming prevalent in the field of deep learning.

VI. CONCLUSION

In conclusion, our research on Efficient Vision Transformers demonstrates the effectiveness of a once-trained (OT) network framework for deploying Vision Transformers across diverse hardware platforms. By decoupling the training phase from the neural architecture search phase, we have successfully developed a methodology that significantly reduces the computational and energy costs associated with deploying specialized ViT models. The experimental results demonstrate the efficacy of our approach. We achieved significant reductions in model size, memory usage, and inference latency while preserving competitive accuracy on the CIFAR-100 dataset.

Our research demonstrates the potential of deploying a once-trained network on any resource-constrained device without the need of additional training. The required constraints can be met while still maintaining a sufficient accuracy on the dataset. Additionally, this can be looked upon as getting a pre-trained network of any required dimension, memory use, latency which can be readily used for other similar applications.

This work paves the way for more scalable and efficient deployment of Vision Transformers across a wide range of hardware platforms, from high-end devices to resource-constrained edge environments. By enabling rapid extraction of specialized sub-networks without additional training, our OT network framework offers a practical solution to the challenges of deploying state-of-the-art vision models in diverse real-world scenarios.

REFERENCES

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019. [Online]. Available: https://arxiv.org/abs/1801.04381

[3] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," 2017. [Online]. Available: https://arxiv.org/abs/1707.01083

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2016. [Online]. Available: https://arxiv.org/abs/1510.00149

[5] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, *AMC: AutoML for Model Compression and Acceleration on Mobile Devices*. Springer International Publishing, 2018, p. 815–832. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-01234-2_48

[6] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," 2020. [Online]. Available: https://arxiv.org/abs/1908.09791

[7] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," 2023. [Online]. Available: https://arxiv.org/abs/2307.08691

[8] G. Fang, X. Ma, M. Song, M. B. Mi, and X. Wang, "Depgraph: Towards any structural pruning," 2023. [Online]. Available: https://arxiv.org/abs/2301.12900

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015. [Online]. Available: https://arxiv.org/abs/1503.02531

[10] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2017. [Online]. Available: https://arxiv.org/abs/1611.01578

[11] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[13] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," 2021. [Online]. Available: https://arxiv.org/abs/2012.12877