# SimFreeze: Adaptively Freeze Vision Transformer Encoders with Token Similarity

Tianyi Shen, Chonghan Lee, Vijaykrishnan Narayanan
Pennsylvania State University
State College, Pennsylvania, USA
{tqs5537, cvl5361, vxn9}@psu.edu
*

## Abstract

*With the remarkable success of transformers in the field of Natural Language Processing (NLP), researchers from both industry and academia are actively seeking to design large language models with increased parameters and deeper transformer encoder and decoder layers. However, as the model size increases, the training and fine-tuning processes become more time-consuming, resulting in large computational costs and energy consumption. As for now, the training of these large models is predominantly undertaken by major corporations, typically spanning several weeks or even months. The fine-tuning stages of these models on downstream tasks also require large computational costs mainly due to the enormous model size and computation complexity of the state-of-the-art models such as Vision Transformer and BERT. In this paper, we introduce a novel algorithmic solution for adaptive layer-freezing to reduce the total fine-tuning duration and energy consumption exemplified by the vision transformer model. Our experiments with ViT-Base and ViT-Large show that the proposed strategies can accelerate fine-tuning by up to* $1.83\times$ *and* $2.02\times$*, at the cost of only a 0.438% and 0.210% drop in accuracy, which results in up to 28.6% and 37.4% speedup of previous SOTA algorithm solutions.*

## 1. Introduction

With the novel architecture that addresses key limitations of previous models including Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), transformer models [18] show state-of-the-art performance in the field of natural language processing (NLP). The advent of large-scale pre-trained transformers such as BERT [3] and GPT-3 [2] marks a significant milestone, showcasing extraordinary proficiency across various NLP tasks. These large language models are pre-trained using unsupervised learning on a vast amount of corpus data to capture semantic representations of the language. The pre-trained models are later fine-tuned for specialized downstream tasks. Their application spans a wide spectrum, including sentiment analysis, contextual similarity assessment, and question answering.

Inspired by the great success in NLP, the transformer models have been adapted to the computer vision domain. Vision transformers [4] have demonstrated their performance compared to the traditional CNN models in various computer vision tasks. While CNNs rely on local receptive fields, vision transformers leverage attention mechanisms to capture both local and global correlations across the entire image and achieve superior performance.

However, transformer models present substantial challenges in terms of training and fine-tuning efficiency. Training the gigantic GPT-3 with 175 billion parameters costs 1287 MWh of electricity, resulting in a million pounds of carbon emission, equivalent to driving 123 gasoline-powered cars for a year [11]. Additionally, the repeated process of designing and training on neural architecture search (NAS) model of a relatively modest 12-layer transformer generates 626,000 pounds of $CO_2$ emission, comparable to the lifetime emission of 5 cars [15]. It is critical to improve the training and fine-tuning efficiency of complex models such as transformers. Reducing training and fine-tuning time could lead to significant cost savings in terms of hardware and energy consumption either when training on cloud-based infrastructure or resource-constrained devices or IoT devices. For NAS, the time required for the repetitive process of designing and fine-tuning models suitable to the deployment constraints could be significantly reduced. Furthermore, in domains where data distribution changes rapidly, models may need to be retrained and fine-tuned frequently. Reducing training and fine-tuning time enables

*This work was supported in part by SRC/DARPA JUMP 2.0 PRISM Center.

faster adaptation to the changing data patterns.

There have been efforts to improve the efficiency of training and fine-tuning neural networks by freezing certain layers. Recent research discovered that freezing certain CNN layers at specific points during the fine-tuning process can significantly reduce the overall fine-tuning time and cost without notably compromising the model's accuracy. However, freezing layers of large transformer-based models have not been extensively studied in the existing works.

In this work, We developed a novel adaptive layer-freezing algorithm for the ViT model, by comparing the class token similarity before and after freezing the evaluating layer. This algorithmic approach resulted in an efficiency boost ranging from $1.57\times$ to $1.80\times$ for the ViT-B model, with a minimal accuracy reduction between $0.267\%$ to $0.438\%$. The ViT-L model achieves efficiency improvements ranging from $1.59\times$ to $2.02\times$, accompanied by an accuracy variation between a decrease of $0.210\%$ and an increase of $0.288\%$.

## 2. Related Works

It is observed that not all layers in neural networks need to be trained equally. Normally, the model will give large weight updates in the early stage of the fine-tuning and the greatest weight change usually occurs in certain few layers [1, 8, 20]. As the fine-tuning goes on, the model weight updates will gradually decrease and finally converge. However, from the perspective of fine-tuning efficiency and energy conservation, that process is not efficient. As the weight changes decrease, the fine-tuning itself will have less impact on the model performance whereas the computations and data movements remain the same. When approaching the end of the fine-tuning, since the model is almost converged and starts to overfit, most of the operations become redundant and keep calculating and writing floats with small differences. As the early termination technique alleviates that phenomenon from the model level, the non-equivalent layer weight updates induce the idea of ”layer freezing” which can be regarded as a layer-level early termination.

Layer freezing stops updating the weight of a layer in a deep learning model once that layer is already saturated, and the idea was first proposed by Brock et al. in the FreezeOut algorithm [1] where four strategies with configured freezing points along with layer-specific learning rate scheduling were applied to DenseNets [6], WideResNets [22], and VGG [14]. To make the process itself adaptive and eliminate the need to configure based on different datasets and models, Liu et al. proposed AutoFreeze [10] which utilized gradient norm as a metric to automatically determine the optimal time step for freezing. Inspired by that, numerous approaches are proposed: PipeTransformer [5] decided on freezing conditions based on the singular vector canonical correlation analysis (SVCCA) score that measures the similarity between different layers [12]; SmartFRZ [9] generated a special dataset using the Centered Kernel Alignment (CKA) similarity index and trained an attention-based predictor model to determine the timing for freezing layers; Egeria [20] employed a technique of comparing differences between intermediate activation tensors across layers; and SpFDE[21] utilized flops as an indicator for progressive layer freezing in sparse training domains.

However, FreezeOut [1] is not an adaptive freezing algorithm, which means its freezing parameters need to be adjusted based on the model and datasets and was verified working on DeiT [16] by Li et al. [9] with only simple datasets such as CIFAR-10 and CIFAR-100; AutoFreeze [10] has the assumption that transformer encoder layers will converge in order, which is not correct in vision transformer and the layer norm approach is not effective in certain scenarios with gradient fluctuation during model training, which provides a sub-optimal freezing pattern, causing a smaller speedup ratio [9]; calculating the SVCCA score for PipeTransformer [5] is too expensive which can even slow down the overall fine-tuning process, indicated by the $\leq 1$ speedup in its paper. Egeria [20] and SpFDE [21] were not implemented for ViT because ViT encoder layers have the same dimension and will converge at almost the same time. SmartFRZ [9] used additional layers to determine the freezing time which generates extra flops during the training, causing only small improvements compared with FreezeOut [1].

In our work, we propose an algorithmic approach that can be easily integrated into existing vision transformer training scripts with minimal structural modifications to the model. Specifically, our method involves adding an extra output to the model, the class token, allowing for an efficient and easily deployable solution for layer freezing. Compared with [1] which needs a manually decided freezing timestamp, our algorithm provides a dynamic and automatic solution for determining the moment to freeze layers.

## 3. Approach

### 3.1. Feature Memory

In a standard ViT model, the class token is fed to the classification head, which contains the overall context information among all the tokens. To circumvent the influence of the classifier head in the freezing evaluation processes, we introduced the class-wise feature memory.

The class-wise feature memory stores token features by classes to bypass the need for the classifier head in performance evaluation. The feature memory is an average of the raw distribution of the class token, encapsulating the essential characteristics of its respective class features. With those average distributions, we are able to assess the performance of the ViT backbone without using the classifier results.

### 3.2. Two State Freezing Evaluation Framework

To adaptively decide when to freeze layers based on the feature memory, we designed a two-state freezing evaluation framework. As outlined in 1, the framework alternatively switches between two states: recording and comparing.

During the recording state, the currently frozen layer will be restored from the frozen state, and the class-wise raw distributions from the class tokens will be used to update the feature memory. This aggregated and averaged class-wise feature representation reflects the model performance with the currently evaluating layer being active.

In the comparing state, the current layer to assess for freezing will enter the frozen state and the algorithm will compute the cosine similarity between the average distribution stored in the feature memory and the current input to compare the performance between the frozen and non-frozen cases. This process compares the model performance when the evaluating layer is idle to the performance when the evaluating layer is active and will generate an average similarity score at the end of each comparing interval. After completing this phase, the values in the feature memory are reset to collect new feature distributions to assess the next layer to freeze.

The entire layer freezing process initiates from the first encoder to the last encoder and the algorithm evaluates each layer by switching between the recording and comparing states.

A critical aspect of this evaluation system is the criterion for freezing a layer. The decision to freeze is based on the average similarity scores derived during the comparing state. From the observation shown in our experiments, the similarity score for each layer behaves like an inversed hyperbola which increases with a decreasing gradient (speed) and eventually converges and fluctuates along a horizontal line. We only consider the evaluating layer to be good enough to enter the frozen state as the similarity score converges when freezing the evaluated layer will not affect the model performance much. In other words, the algorithm will freeze the model once the similarity score fluctuates along a horizontal line. Then, layers are only frozen when they have sufficiently learned the necessary features, thus maintaining the performance and efficiency of the learning process.

**Algorithm 1:** Adaptive Token Similarity Freezing

```
$T_i^{cls}$: Class token from the last ViT encoder at
  iteration i.
$y_i^{cls}$: Ground truth label of the input image at
  iteration i.
$\eta_k$: Accumulated similarity score at k-th comparing
  interval.
$L_n$: n-th encoder from ViT model.
for number of iterations do
    if in recording state then
        unfreeze($L_n$);
        while in recording interval do
            feature_memory.update($T_i^{cls}$, $y_i^{cls}$);
        end
        state = Comparing;
        continue;
    else
        freeze($L_n$);
        while in comparing interval do
            feature_memory.cos_similarity($T_i^{cls}$,
              $y_i^{cls}$);
        end
        $\eta_k$ = feature_memory.average_similarity();
        feature_memory.clear_distribution();
        state = Recording;
        if $\eta_k \leq \eta_{k-1}$ then
            $L_n$ = $L_{n+1}$;
        end
    end
end
```

## 4. Experiment Results

### 4.1. Setup

In our study, we tested our layer freezing algorithm on two configurations of the Vision Transformer model: the ViT-Base (ViT-B/16) and the ViT-Large (ViT-L/16), both of which were pre-trained on the Imagenet21K dataset[13]. And the performance of the model will be evaluated on three Fine-Grained Visual Categorization (FGVC) datasets: CUB-200-2011 [19], Stanford Dogs [7], and NAbirds [17].

During the fine-tuning, we utilized the SGD optimizer, with a momentum of 0.9 along with a cosine annealing scheduler. The initial learning rate was set at 0.003, and the batch size was set to 16. All the experiments were conducted on a Nvidia Tesla A100 GPU.

### 4.2. Performance Evaluation

As we talked about in the related work section, algorithms like Egeria [20] and SpFDE [21] can not be implemented for ViT because ViT encoders maintain the same dimension and converge at the same time, and PipeTransformer [5] requires too much computation on SVCCA score. AutoFreeze [10] assumes that transformer encoders will converge in order, which is not the case for ViT, and it doesn't give satisfying speedup on DeiT fine-tuning from the experiment of Li et al. [9]. SmartFRZ [9] requires extra training

| | CUB | | Stanford Dogs | | NAbirds | |
|---|---|---|---|---|---|---|
| **Algorithm** | **ΔAcc.** | **Speedup** | **ΔAcc.** | **Speedup** | **ΔAcc.** | **Speedup** |
| **ViT-B** | 0% | 1x | 0% | 1x | 0% | 1x |
| **ViT-B-SLF** | -0.345% | 1.14x | -0.828% | 1.24x | -1.364% | 1.10x |
| **ViT-B-ULF** | -0.570% | 1.18x | -1.049% | 1.23x | -1.149% | 1.14x |
| **ViT-B-SCF** | -0.553% | 1.46x | -1.340% | 1.48x | -0.459% | 1.28x |
| **ViT-B-UCF** | -0.518% | 1.42x | **-0.128%** | 1.40x | **-0.244%** | 1.32x |
| **ViT-B-ATSF(ours)** | **-0.267%** | **1.70x** | -0.268% | **1.80x** | -0.438% | **1.57x** |

Table 1. Layer Freezing Algorithm Evaluation on ViT-B/16.

| | CUB | | Stanford Dogs | | NAbirds | |
|---|---|---|---|---|---|---|
| **Algorithm** | **ΔAcc.** | **Speedup** | **ΔAcc.** | **Speedup** | **ΔAcc.** | **Speedup** |
| **ViT-L** | 0% | 1x | 0% | 1x | 0% | 1x |
| **ViT-L-UCF** | -1.243% | 1.50x | -0.361% | 1.47x | -1.307% | 1.40x |
| **ViT-L-ATSF(ours)** | **-0.121%** | **1.59x** | **-0.210%** | **2.02x** | **+0.288%** | **1.95x** |

Table 2. Layer Freezing Algorithm Evaluation on ViT-L/16.

with generated Centered Kernel Alignment data, which is not released in public. Then, the only work we can make a proper comparison is FreezeOut [1] which shows great speedup on DeiT fine-tuning (as good as SmartFRZ) with tolerable accuracy drop. This led us to re-implement four distinct algorithms that were originally presented in FreezeOut [1].

## 4.3. Results

### 4.3.1 Evaluation of Algorithms

We first evaluated our algorithm: Adaptive Token Similarity Freezing (ATSF) on the ViT-B/16 model. To benchmark its effectiveness, we also implemented and analyzed four algorithms from [1]: Scaled Linear Freezing (SLF), Unscaled Linear Freezing (ULF), Scaled Cubic Freezing (SCF), and Unscaled Cubic Freezing (UCF).

Our findings in Table 1 show that the Cubic Freezing techniques (SCF and UCF) generally offered a higher speedup ratio, ranging from 1.28× to 1.42×, than Linear Freezing methods (SLF and ULF) which gives the speedup ratios between 1.10× and 1.18× across all datasets. This difference in efficiency is attributed to the Cubic Freezing's tripled split factor $\alpha(\alpha \leq 0)$, which initiates layer freezing earlier in the training process.

Compared with existing techniques, our algorithm (ATSF) exhibited impressive results. It achieved a significantly higher speedup ratio, ranging from 1.57× to 1.80×. It proves that the feature memory distribution is a good pre-classifier indicator to determine the optimal point for layer freezing. From a performance standpoint, our algorithm resulted in a tolerable accuracy degradation within 0.5%, which is comparable to the best-performing static freezing algorithm. That great tradeoff for efficiency underscores the potential of our algorithm as the new state-of-the-art approach for accelerating vision transformer-based model fine-tuning.

### 4.3.2 Scalability of Algorithms

In our initial experiments, the adaptive algorithm showcased impressive results with the ViT-B model. However, given the current trend in neural network research towards constructing larger and deeper networks, it was imperative to ascertain whether this technique could maintain its efficacy across a broader range of fine-tuning scenarios. To address this, we expanded our analysis to the more complex ViT-L/16 model.

For a comprehensive comparison, we selected the Unscaled Cubic Freezing (UCF) method—identified as the most efficient among static freezing algorithms—as a benchmark for our adaptive algorithm.

The results in Table 2 were enlightening. Our adaptive method not only achieved higher speedups in the larger ViT model across some datasets (2.02× in the Stanford Dog dataset and 1.95× in the Nabirds dataset) but also manifested a smaller overall accuracy drop (-0.121% in the CUB-200-2011 dataset and -0.210% in the Stanford Dog dataset). Remarkably, in the Nabirds dataset, our method even demonstrated an accuracy boost of +0.288%. In contrast, while the static freezing algorithms showed similar speedup ratios, they incurred greater accuracy decreases.

## 5. Conclusion

In this work, we introduced a layer-freezing framework designed to adaptively stop encoder updates during the fine-tuning stage of the vision transformer model. Our method utilizes the class token after the final encoder layer. The token distribution is aggregated and averaged for each class of the dataset in the feature memory and our framework measures the similarity scores that represent the learning condition of the model to help determine the layer freezing point. Our proposed freezing framework achieved a speedup ranging from 1.57× to 1.80× on ViT-B and a speedup ranging from 1.59× to 1.95× on ViT-L without compromising the model performance much. This advancement in layer-freezing methodology holds promising implications for reducing fine-tuning costs for LLM and LVM.

## References

[1] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Freezeout: Accelerate training by progressively freezing layers. *ArXiv*, abs/1706.04983, 2017. 2, 4

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901. Curran Associates, Inc., 2020. 1

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. 1

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 1

[5] Chaoyang He, Shen Li, Mahdi Soltanolkotabi, and Salman Avestimehr. Pipetransformer: Automated elastic pipelining for distributed training of large-scale models. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4150–4159. PMLR, 2021. 2, 3

[6] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Weinberger. Densely connected convolutional networks. 2017. 2

[7] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, 2011. 3

[8] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. *ArXiv*, abs/1905.00414, 2019. 2

[9] Sheng Li, Geng Yuan, Yue Dai, Youtao Zhang, Yanzhi Wang, and Xulong Tang. SmartFRZ: An efficient training framework using attention-based layer freezing. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 3

[10] Yuhan Liu, Saurabh Agarwal, and Shivaram Venkataraman. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning. *ArXiv*, abs/2102.01386, 2021. 2, 3

[11] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021. 1

[12] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Narain Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Neural Information Processing Systems*, 2017. 2

[13] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021. 3

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 2014. 2

[15] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13693–13696, 2020. 1

[16] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 2

[17] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 595–604, 2015. 3

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. 1

[19] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds200-2011 dataset. *Advances in Water Resources - ADV WATER RESOUR*, 2011. 3

[20] Yiding Wang, Decang Sun, Kai Chen, Fan Lai, and Mosharaf Chowdhury. Egeria: Efficient dnn training with knowledge-guided layer freezing. *Proceedings of the Eighteenth European Conference on Computer Systems*, 2022. 2, 3

[21] Geng Yuan, Yanyu Li, Sheng Li, Zhenglun Kong, Sergey Tulyakov, Xulong Tang, Yanzhi Wang, and Jian Ren. Layer freezing & data sieving: Missing pieces of a generic framework for sparse training, 2022. 2, 3

[22] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2017. 2