EFFICIENT PRETRAINING OF VISION TRANSFORMERS: A LAYER-FREEZING APPROACH WITH LOCAL MASKED IMAGE MODELING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

UTKU MERT TOPÇUOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF MULTIMEDIA INFORMATICS

SEPTEMBER 2024

EFFICIENT PRETRAINING OF VISION TRANSFORMERS: A LAYER-FREEZING APPROACH WITH LOCAL MASKED IMAGE MODELING

submitted by **UTKU MERT TOPÇUOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Multimedia Informatics Department, Middle East Technical University** by,

Prof. Dr. Banu Günel Kılıç
Dean, **Graduate School of Informatics**

Assoc. Prof. Dr. Elif SÜRER
Head of Department, **Modeling and Simulation**

Assoc. Prof. Dr. Erdem Akagündüz
Supervisor, **Modeling and Simulation, Middle East Technical University**

Examining Committee Members:

Prof. Dr. Alptekin Temizel
Modeling and Simulation, Middle East Technical University

Assoc. Prof. Dr. Erdem Akagündüz
Modeling and Simulation, Middle East Technical University

Assist. Prof. Dr. İrem Ülkü
Computer Engineering Deperatment, Ankara Üniversitesi

Date: 03.09.2024

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname:	**Utku Mert Topçuoğlu**
Signature	**:**

ABSTRACT

EFFICIENT PRETRAINING OF VISION TRANSFORMERS: A LAYER-FREEZING APPROACH WITH LOCAL MASKED IMAGE MODELING

Topçuoğlu, Utku Mert
M.S., Department of Multimedia Informatics
Supervisor: Assoc. Prof. Dr. Erdem Akagündüz

September 2024, 47 pages

This thesis explores the acceleration of pre-training Vision Transformers (ViTs) for self-supervised learning by integrating progressive layer freezing with local masked image modeling. The study aims to address the significant computational demands and lengthy training times inherent in training ViTs when employing self-supervised methods like masked image modeling. The core contribution of this research lies in integrating the FreezeOut [1] method into the LocalMIM architecture, enhancing training efficiency by systematically freezing specific layers at strategic points during training.

We evaluate whether the FreezeOut method is as effective as proposed in the original paper across different optimizers, acknowledging that learning rate scheduling is optimizer-dependent. Our experimental results demonstrate that the proposed approach can reduce training time by approximately 12.5% with a minimal drop in top-1 accuracy (0.6%). Furthermore, we introduce and validate a novel learning rate scheduling method tailored for ViTs, which achieves an even more negligible accuracy drop of 0.1% with an 83.1% top-1 accuracy. We demonstrate that the number of training epochs and dataset complexity are critical factors for the effectiveness of the FreezeOut method and show that it performs even better with longer training epochs or simpler datasets. Our specially designed learning rate scheduling method showed greater robustness to fewer training epochs and more complex datasets, explaining its superior results in the 100 epoch IN-1K training setup.

This research offers a solution for enhancing the efficiency of ViT pre-training, making self-supervised learning more accessible in environments with constrained computational resources. The findings contribute to the broader field of computer vision by highlighting the potential of progressive layer freezing and adaptive learning rate scheduling in optimizing training processes for ViTs. The implementation of our approach is accessible here: `https://github.com/utkutpcgl/ViTFreeze`.

Keywords: Vision Transformers, Self-Supervised Learning, Local Masked Image Modeling, Progressive Layer Freezing, Computational Efficiency, Multi-Scale Reconstruction, Adaptive Learning Rate Scheduling, FreezeOut Learning Rate Scheduling, Training Time Reduction

ÖZ

GÖRSEL DÖNÜŞTÜRÜCÜLERİN VERİMLİ ÖN EĞİTİMİ: YEREL MASKELİ GÖRÜNTÜ MODELLEME İLE KATMAN DONDURMA YAKLAŞIMI

Topçuoğlu, Utku Mert
Yüksek Lisans, Çoklu Ortam Bilişimi Bölümü
Tez Yöneticisi: Doç. Dr. Erdem Akagündüz

Eylül 2024, 47 sayfa

Bu tez, yerel maskeli görüntü modelleme ile aşamalı katman dondurmanın entegrasyonu yoluyla kendi kendine denetimli öğrenme için Görüntü Dönüştürücülerin (ViT'ler) ön eğitimini hızlandırmayı araştırmaktadır. Çalışma, özellikle maskeli görüntü modelleme gibi kendi kendine denetimli yöntemler kullanıldığında, ViT'lerin eğitiminde mevcut olan önemli hesaplama gereksinimlerini ve uzun eğitim sürelerini ele almayı amaçlamaktadır. Bu araştırmanın ana katkısı, eğitim sırasında belirli katmanları stratejik noktalarda sistematik olarak dondurarak eğitim verimliliğini artıran FreezeOut [1] yönteminin LocalMIM mimarisine uygulanmasında yatmaktadır.

FreezeOut yönteminin, öğrenme oranı planlamasının optimize ediciye bağımlı olduğunu kabul ederek, farklı optimize edicilerde, orijinal makalede önerildiği kadar etkili olup olmadığını değerlendiriyoruz. Deneysel sonuçlarımız, önerilen yaklaşımın eğitim süresini yaklaşık %12,5 oranında azaltabileceğini ve top-1 doğrulukta yalnızca %0,6'lık bir düşüşle minimal bir kayba yol açtığını göstermektedir. Ayrıca, ViT'ler için uyarlanmış yeni bir öğrenme oranı planlama yöntemini tanıtıyor ve doğruluyoruz; bu yöntem, %0,1'lik daha önemsiz bir doğruluk düşüşü ile %83,1 top-1 doğruluğa ulaşmaktadır. Eğitim dönemlerinin sayısının ve veri kümesi karmaşıklığının FreezeOut yönteminin etkinliği için önemli faktörler olduğunu gösteriyor ve bu yöntemin daha uzun eğitim dönemlerinde veya daha basit veri kümelerinde daha iyi performans gösterdiğini kanıtlıyoruz. Özel olarak tasarlanmış öğrenme oranı planlama yöntemimiz, daha az sayıda eğitim dönemi ve daha karmaşık veri kümelerine karşı daha büyük bir dayanıklılık göstermiş ve bu da 100 dönemlik IN-1K eğitim kurulumundaki üstün sonuçlarını açıklamaktadır.

Bu araştırma, ViT ön eğitimini hızlandırmak için bir çözüm sunarak, kendi kendine denetimli öğrenmeyi sınırlı hesaplama kaynaklarına sahip ortamlarda daha erişilebilir hale getirmektedir. Bulgular, aşamalı katman dondurma ve uyarlamalı öğrenme oranı planlamasının ViT eğitim süreçlerini optimize etmedeki potansiyelini vurgulayarak bilgisayarla görü alanına katkılar sağlamaktadır. Projenin kaynak koduna buradan ulaşabilirsiniz: `https://github.com/utkutpcgl/ViTFreeze`.

Anahtar Kelimeler: Görsel Dönüştürücüler, Özdenetimli Öğrenme, Yerel Maskeli Görüntü Modellemesi, İlerlemeli Katman Dondurma, Hesaplama Verimliliği, Çok Ölçekli Yeniden Yapılandırma, Uyarlamalı Öğrenme Oranı Zamanlaması, Dondurma Öğrenme Oranı Zamanlaması, Eğitim Süresi Azaltma

I dedicate this thesis to my family and my beloved spouse, who have always supported me.

ACKNOWLEDGMENTS

I want to express my gratitude to my advisor, Assoc. Prof. Dr. Erdem Akagündüz, for his guidance and support throughout this thesis. I am also thankful to Kuartis for allowing me to use their GPUs during the research process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Networks
ViT	Vision Transformer
SSL	Self-Supervised Learning
MIM	Masked Image Modeling
MHA	Multi-Head Attention
IN-1K	ImageNet-1K
DNN	Deep Neural Network
SA	Simulated Annealing
SR-LR	Stage-wise Reduction of Learning Rate
LR-LR	Layer-wise Reduction of Learning Rate
VAE	Variational Auto Encoder
NLP	Natural Language Processing

CHAPTER 1

INTRODUCTION

In this study, the recently popularized, one of the most capable and data-hungry architectures, Vision Transformers (ViTs) [2], are taken into consideration in an even more demanding training setup in the pursuit of more efficient model training a layer freezing technique. This chapter focuses on the advantages and challenges of Vision Transformers (ViTs), self-supervised learning (SSL), and layer-freezing techniques. We highlight the efficiency issues in SSL training for ViTs, which require substantial computational resources and time.

1.1 The Need for Efficient Models

More data is being stored, better sensors are being produced as the world becomes more digitalized at a fast pace. This enables deep learning models to benefit from larger scales. Even though the model performance tends to increase logarithmically as the data increases exponentially [3], the benefits of model scaling remain the most simple and effective way to improve model performance.

As a result, reducing the compute requirement of a model allows it to consume less energy and possibly achieve better results by compressing larger models. Hence, efficient models not only enable faster and smaller models but also pave the way for better-performing models in a compute or energy-constrained environment.

An important factor in this has been hardware innovations. Thanks to model optimizations in this field, deep learning models' energy consumption has not increased exponentially, while the compute requirements have increased exponentially [4]. Even if the pace of hardware innovations slows down, methods that increase model efficiency will allow for larger models to outperform when trained on larger datasets, by enabling them to run larger models on compute-restricted settings.

Moreover, in developing countries, like Turkey, it is hard or even impossible to access HPC (high-performance computing) to train large SOTA models. Likewise, low-budget individuals lack the opportunity to train or contribute to the field. Energy/Cost/Time efficient models can remove this barrier and democratize deep learning.

These efficient models can perform faster, are less energy-consuming, and less memory requiring inference and training. Learning rate and layer convergence are fundamental in these methods, as in model convergence. A simple yet effective method is layer freezing, however, the interaction between layer convergence and layer freezing has not been understood clearly, though highly used.

1.2 Related Approaches and Challanges

It is important to understand the methods at hand with the challenges they face before describing the specific problem we tackle. Here we focus on these issues in the context of ViT, SSL, and layer freezing.

1.2.1 Vision Transformer

The transformer architecture, successful in NLP [5], has also excelled in computer vision, often outperforming or matching traditional CNNs [6] in various tasks. A key advantage of transformers is their ability to process multiple data modalities, enabling flexible and comprehensive analysis. Furthermore, vision transformers effectively capture long-range dependencies within images, leading to more accurate and detailed representations, making them ideal for complex vision tasks that demand a deep understanding of spatial relationships.

However, unlike convolutions, the MHA layer lacks inductive bias, leading to increased capacity and associated challenges [2]. These challenges include the need for more data, complex initialization techniques, and pre-training to mitigate overfitting. Additionally, global attention introduces quadratic computational complexity, where increasing the pixel count linearly results in a quadratic rise in computational demands. Consequently, ViTs require significantly more training time.

1.2.2 Self Supervised Learning and Masked Image Modeling

Another vision method that requires a lot of data and training is self-supervised learning, such as contrastive learning and masked image modeling. It removes the need for labeled data and allows models to learn robust and generalizable features from the data itself. This has significantly benefited Vision Transformers (ViTs), which thrive on diverse learning signals to mitigate their tendency to overfit.

Contrastive learning uses data augmentation to learn invariant, robust feature representations by contrasting augmented views of the same image [7]. Masked image modeling, inspired by masked language modeling in NLP, has also become a powerful approach to vision. It requires the model to predict the original content of masked image regions, thus fostering an understanding of contextual relationships [8].

Despite all its advantages, self-supervised learning techniques mostly require large datasets and lengthy training for the pre-trained model to surpass regular pre-trained models on popular datasets. They excel on large, diverse datasets and extended training iterations. For example, the pre-trained ResNet50 on ImageNet-1K already improves the performance of ResNet50 on most downstream tasks [2]. To surpass this, it is mostly required to train longer on datasets where many different views of an image are aided with complex augmentations. This is also true for masked image modeling, where it is hard to find the model converging even in long training iterations. This is why it is common to see 1600 epochs of training durations for the MIM task [8].

1.3 A Solution: Layer Freezing

A simple and effective method for increasing model training speed is layer freezing. Although mostly used in transfer learning, it has been also explored to train models efficiently from scratch. As early layers mostly learn simpler features [9] and later features build up on early activation outputs, mostly initial layers are preferably frozen earlier during this setup. Convergence of layers is particularly sensitive to the Learning rate, hence layer-wise learning rates have been proposed [1].

However, it is important to understand when a layer has converged, and freeze it if there is no harm to the learning process or the performance. Yet, it is not clear when a layer has converged, as even smaller gradients do not guarantee convergence [10] or that freezing a particular layer with a lower gradient does not harm the model performance. Even though there are studies that freeze layers to speed model training with minimal to no accuracy drop, the optimal timing and method to freeze remain unclear. Also, most layers are considered as a whole, as freezing a layer is much more beneficial than unit freezing due to the parallel structure of deep models.

1.4 Problem Statement

Considering the hunger for data and the requirement for long training epochs of these methods, MIM and ViT combined, need and benefit from even longer epochs and larger datasets. This enhances the importance of efficient and faster training methods utilized while training ViTs with SSL. Considering larger models and ViT variations are introduced gradually, this issue seems to remain and will continue to grow.

Therefore, the primary issue addressed in this thesis is the inefficiency of Mased Image Modeling training for Vision Transformers. Although ViTs have shown significant promise in handling diverse data modalities and achieving superior results in various vision tasks, their training processes are plagued by extensive computational requirements and long training times. The high-capacity nature of ViTs, combined with the absence of inductive biases, necessitates large datasets and complex pre-training techniques to prevent overfitting. Furthermore, the global attention mechanism used in ViTs introduces quadratic computational costs as image size increases [2]. As the model size increases, the slow training speed and data hunger problems increase further.

SSL methods benefit ViTs by removing the need for labeled data and helping the models learn robust features from the data itself. However, these methods also demand significant compute and time. For instance, training a ViT-L model using masked autoencoding on 128 TPU-v3 cores can take 31 hours for 1600 epochs, while another SSL method, MoCo v3, requires 36 hours for 300 epochs [8]. Such prolonged training times are impractical for many users and applications without access to large-scale compute.

Furthermore, applying layer freezing techniques to this complex setup has not been investigated to our knowledge. Apart from the efficiency gains, there are insight to be gained in this pursuit.

1.5 Aim and Scope of the Research

Primary Objective:

Accordingly, the primary objective of this research is to accelerate the pre-training process of Vision Transformers (ViTs) with minimal drop in accuracy by using a special layer freezing technique aided with a layer-wise learning rate method. Making SSL ViT training more accessible.

Scope:

This research focuses on self-supervised learning (SSL), particularly masked image modeling. The study is centered around the ViT-B model, examining its performance over 100 training epochs. This focused approach allows for a detailed and controlled examination of the proposed methodologies.

In addition, the research aims to explore and validate strategies such as progressive layer freezing and adaptive learning rate scheduling. By systematically freezing specific layers at strategic points during training, the study seeks to reduce the computational burden while maintaining model performance. The adaptive learning rate scheduling is intended to optimize the training process further, potentially improving the performance of ViTs.

1.6 Research Questions

The research questions guiding this thesis are designed to explore the effectiveness and applicability of progressive layer freezing and adaptive learning rate scheduling in the context of self-supervised learning for Vision Transformers. Specifically, the questions aim to address the efficiency, performance, and adaptability of these techniques:

- **Question 1:** Does FreezeOut effectively speed up model training without significantly reducing performance, and can it maintain accuracy while decreasing training time?
- **Question 2:** How does the FreezeOut learning rate schedule, which increases early layer learning rates, impact the performance? While originally tested with SGD, does this method work with other optimizers like AdamW [11]?
- **Question 3:** Can the FreezeOut technique be applied to the Vision Transformer architecture LocalMIM to accelerate the masked image modeling pre-training task? Does it enhance training efficiency for this specific architecture?
- **Question 4:** Does the learning rate scaling of FreezeOut improve the layer freezing method out of the box, or does it require adjustments for LocalMIM to achieve optimal results?
- **Question 5:** Considering that the original FreezeOut method was tested on CIFAR-10 and CIFAR-100 datasets, how do dataset size, complexity, and training epoch count affect the results when applied to IN-1K with LocalMIM?

1.7 Contributions of the Study

This thesis makes several contributions to Vision Transformers (ViTs) and self-supervised learning (SSL). The key contributions are as follows:

Novel Application of Methodology: We introduce an innovative approach that combines local masking with progressive layer freezing to enhance the efficiency of ViT training. To our knowledge, this is the first method that applies layer freezing to ViT in the SSL training setting.

New Dynamic Learning Rate Scaling Method: We propose a new dynamic learning rate scaling method to enhance FreezeOut's integration to LocalMIM, specific to the multi-stage structure and large gradients of initial layers.

Efficiency Gains: The study provides a detailed quantification of the training time reductions achieved through the proposed methodology. It systematically analyzes the impact of these efficiency gains on the model's accuracy, demonstrating that reductions in training time can be achieved with minimal performance drop.

Theoretical Findings and Implications:

- Better upstream task performance might not translate to downstream tasks directly.
- The FreezeOut method benefits from easy-to-converge training configurations, such as extended training epochs and simpler datasets.
- Multi-stage architectures might increase the early layer gradients too much when their losses contribute linearly to the total loss. This should be taken into consideration with special learning rate scaling methods.
- FreezeOut's learning rate scaling method might not drastically improve the performance (as stated in the paper). It might require case-specific modification for different architectures.
- Our results support that FreezeOut can also be used for other architectures. The original FreezeOut paper claims that skip-connections availability is required for the method to work, and our results support this.
- We show that LocalMIM converges much faster than regular MAE training and even a low number of epochs achieve competitive results on the classification downstream task. However, its contribution to FreezeOut is not apparent.

1.8 Organization of the Thesis

This thesis is structured into five chapters, each addressing different aspects of the research.

- **Chapter 1: Introduction** - Provides an overview of the research background, problem statement, objectives, research questions, and contributions.

- **Chapter 2: Related Work** - A review of the existing literature is presented in this chapter, situating the research within the current state-of-the-art in Vision Transformers (ViTs) and self-supervised learning (SSL).
- **Chapter 3: Methodology** - This chapter details the proposed methods, including the architecture of Vision Transformers, the principles of FreezeOut, and the specifics of Local Masked Image Modeling (LocalMIM), ViTFreeze, and learning rate scaling techniques.
- **Chapter 4: Experiments and Results** - Describes the experimental setup—datasets, training configurations, and evaluation metrics. Presents and analyzes the findings, providing a framework for replicating and assessing the proposed methods.
- **Chapter 5: Discussion** - Analyzes the results and discusses their implications.
- **Chapter 6: Conclusion** - Summarizes the research findings and their contributions and suggests future research directions.

CHAPTER 2

RELATED WORK

Rapid advancements in computer vision and deep learning have led to the development of numerous innovative architectures and methods. This chapter delves into these contributions. We will explore various strategies to enhance model efficiency and accelerate training.

2.1 Architectures

CNNs Since AlexNet [12], which demonstrated the power of deep learning in large-scale image classification, convolutional neural networks have been the gold standard for vision tasks. As a follow-up to AlexNet, VGG [13] introduced a simpler architecture with deeper layers using only 3x3 convolutions. This was further advanced by GoogleNet [14], adding inception modules with varying filter sizes combined. With ResNet [15], residual connections mitigated the problem of vanishing gradients, allowing for much deeper networks. DenseNet [16] improved gradient flow and parameter efficiency by ensuring direct connections between all layers, concatenating initial layer outputs to later layers. EfficientNet [17] optimizes the network's depth, width, and resolution in a balanced manner, maintaining high-resolution representations throughout the network. Additionally, depthwise separable convolutions, as seen in MobileNets [18], have made CNNs more efficient for mobile applications. In contrast, deformable convolutions [19] attend to more important locations in images, focusing on necessary parts of an image.

Vision Transfomers The Vision Transformer (ViT) represents a significant shift in vision backbones, directly applying Transformer architectures to non-overlapping image patches for image classification [2]. ViT achieves an impressive accuracy compared to traditional convolutional networks, though it initially required large-scale datasets like JFT-300M for optimal performance. DeiT introduced training strategies that enabled effective performance on smaller datasets, such as ImageNet-1K [20]. Despite these advancements, the original ViT architecture struggled with high-resolution inputs due to quadratic complexity with image size.

Some studies applied transformer techniques to dense vision tasks. One extended the CNN architecture with transformers for dense vision tasks, object detection, and semantic segmentation[21]. The other has used it for image super-resolution and denoising [22]. Concurrently, other research efforts have modified the ViT architecture to enhance image classification capabilities [23, 24]. The Swin Transformer notably achieves superior accuracy by processing shifted hierarchical windows to increase its field of view and patch diversity [23]. The Swin Transformer's localized operation and linear complexity have proven advantageous in modeling the high correlation in visual signals, achieving state-of-the-art accuracy on tasks like COCO object detection and ADE20K semantic segmentation. Another line of work explores building feature maps of multi-resolution (scale) patches on Transformers, diversifying the field of view and granularity of the extracted features, and applying a special transformer architecture used in NLP named LongFormer to the vision domain [25].

2.2 Self supervised learning

Unsupervised learning and autoencoders Unsupervised learning, a fundamental approach in machine learning, aims to learn patterns in data without using explicit labels. One of the most notable methods in this domain is the autoencoder, which learns efficient codings of input data. Autoencoders consist of an encoder that compresses the input into a latent-space representation and a decoder that reconstructs the input from this representation. This architecture benefits dimensionality reduction, feature learning, and data denoising.

Key developments in autoencoders include the introduction of the Variational Autoencoder (VAE), which integrates probabilistic graphical models and allows for more structured latent space representations [26]. Another important method is the Denoising Autoencoder, designed to reconstruct corrupted input, thereby learning robust features [27]. Additionally, the Sparse Autoencoder enforces sparsity in the latent representation, encouraging the model to learn useful features for downstream tasks [28].

Self-supervised learning in NLP

Self-supervised learning (SSL) in natural language processing (NLP) has revolutionized the field. One of the pioneering works in this domain is the development of word embeddings, such as Word2Vec [29] and GloVe [30]. These models leveraged context to predict word associations, laying the groundwork for more advanced methods.

The advent of transformers marked a significant advancement in SSL for NLP. The introduction of the BERT model showcased the effectiveness of bidirectional training by predicting masked words in a sentence, thus capturing deeper semantic relationships [31]. Subsequent models like RoBERTa improved on BERT by optimizing the pretraining process [32]. GPT-3 extended this paradigm by demonstrating the power of autoregressive models in generating coherent text and performing a wide range of NLP tasks with minimal fine-tuning [33].

Self-supervised learning in Vision. Self-supervised learning has become a significant trend in computer vision as well. Key methodologies include generative techniques like denoising autoencoders [27], image inpainting [34], and colorization [35], which reconstruct data from partial inputs. Contrastive learning, a form of discriminative approach, is notable for creating varied image views and ensuring consistent representations [36, 37], sometimes without contrasting negative pairs [7, 38]. Recent approaches in contrastive learning also incorporate pixel level granularity [39].

Masked image modeling Inspired by Masked Language Modeling (MLM) in NLP, Masked image modeling (MIM) involves predicting occluded or masked regions within images. Foundational methods such as [40] and [8] paved the way, with recent innovations [41] reducing the need for large decoders in the pre-training task, and [42] reconstructing HOG features instead of pixel values to improve the performance and efficiency. These MIM approaches, which use different target signals like normalized pixels [8], discrete tokens from a visual CodeBook like BERT [43], and handcrafted HOG features

[42], have addressed initial challenges related to computational demands and pre-training time. [44] further optimizes the encoding process utilizing only visible tokens, while ConvMAE combines ViTs with CNNs [45] enhancing hierarchical feature understanding.

"MixMAE" by Liu et al. [46] mixes two images and conducts dual reconstruction to improve efficiency. The work of Wang et al. [47], "Masked Image Modeling with Local Multi-Scale Reconstruction," introduces LocalMIM, the baseline model which is used in this thesis, applies local multi-scale and multi-stage reconstruction approach, enriching the representations learned for intermediate encoder stages. "FastMIM" [48], used again HOG features instead of pixels to reconstruct as [42] did, but additionally, they reduced the input resolution, making use of HOGs robustness to lower resolutions and loss of texture. Particularly notable is "Stare at What You See" by Xue et al. [49], which uses non-reconstruction-based image modeling by aligning features extracted from student and teacher models, further increasing efficiency. Furthermore, [50] decides what to mask for the student by using the attention maps generated by a teacher network, increasing the final performance and convergence speed.

2.3 Methods To Improve Speed

Pruning and quantization techniques. While quantization and pruning are mostly designed for inference, these methods also aim to increase training speeds. Quantization reduces the precision of the numbers used to represent model parameters to decrease computational requirements and memory usage. Techniques such as Mixed Precision Training and methods described in [51, 52] demonstrate how reducing numerical precision can accelerate training processes without substantial loss in model accuracy. On the other hand, pruning techniques focus on removing redundant or less significant parameters within neural networks to reduce their size and complexity. The efficacy of pruning in accelerating training is shown in [53, 54, 55]. Also, [56] shows dynamic approaches to pruning that adapt during training, further enhancing efficiency.

Speeding up Training with Layer Freezing and Dropping. Bengio et al. (2006) pioneered this field with their work on layer-wise training, which highlighted the complexity of training deep architectures [57]. Building on this, Brock et al. (2017) introduced FreezeOut, a method for progressively freezing layers, which showed time savings in training with minimal accuracy loss for WideResNets and architectures with skip connections in general [1]. Xiao et al. (2019) furthered this approach by proposing an intelligent layer freezing method based on normalized gradient differences [58], freezing layers by assigning a freezing rate per layer given the magnitude of the gradient of that layer. Zhang and He (2020) extended these concepts to Transformer-based language models, achieving substantial reductions in training time (achieving the same accuracy 2.5x faster) and computational costs (24% less per sample) through progressive layer dropping by using a gating mechanism to skip certain layers during training [59]. Chen et al. (2022) contributed by identifying a Layer Convergence Bias, indicating faster convergence in shallower layers of DNNs, which supports the idea of selective layer training [9]. This paper shows that early layers converge faster than deeper ones (even with smaller gradients), suggesting that early layers are more robust to larger learning rates as they are usually optimized on a flatter landscape. This also explains why FreezeOut's learning rate scaling method works well. The high-frequency content of the image (signal) is more repetitive and easier to learn by early layers (simpler features) and vice versa for low-frequency complex features. Wang et al. (2022) introduced a knowledge-guided layer freezing technique, using semantic knowledge from a reference model to freeze layers during training without sacrificing accuracy [60]. Liu et al. (2021) developed AutoFreeze, an adaptive system for accelerating model fine-tuning by selectively training layers (rather than freezing layers in order) and efficiently caching intermediate activations [61].

2.4 Model Convergence

Simulated Annealing and Learning Rates Effect on Model Convergence. Simulated annealing (SA) is a probabilistic optimization technique inspired by the annealing process in metallurgy, where controlled cooling of material reduces defects, achieving a more stable structure. SA has optimized learning rates in deep learning, enhancing training efficiency and convergence stability. The foundational work [62] highlights the applicability of SA in finding optimal weights in neural networks, demonstrating its effectiveness over traditional methods by escaping local minima and exploring a broader solution space. This is similar to trying to fit a key into a lock. One needs to jiggle the key with larger movements first, smoother, and smaller in the end to place it.

Learning rates are an important determiner of the size of this movement while trying to find the optimal point. The concept of super-convergence, as discussed in [63], demonstrates that training with a cyclical learning rate schedule that includes very high learning rates can drastically reduce training times while maintaining or improving model performance. This method exploits the ability of high learning rates to accelerate learning and subsequently uses lower rates to fine-tune the model, leading to efficient convergence, just as in simulated annealing. Large learning rates act like regularization, similar to simulated annealing in escaping local minima with larger oscillations.

Subsequent research has expanded, incorporating cyclical patterns and warm restarts to refine learning rate schedules further. For instance, [64] introduced a method to vary learning rates within a range, preventing premature convergence and improving model performance. Warm restarts, as detailed in [65], utilize SA principles to periodically reset the learning rate.

Moreover, optimization algorithms like Adam, described in [66], adapt the learning rate for each parameter individually, combining the advantages of two other popular methods: AdaGrad and RMSProp. By maintaining a moving average of the gradient and its square, Adam provides an effective and reliable means to adjust learning rates dynamically, contributing to stable and faster convergence. Unlike simulated annealing, momentum is lagging, hence warm-ups are necessary to move in the correct direction before applying large learning rates. Momentum enables escaping local minima, but the estimates of the first and second moment of the gradients need to be reliable before taking too large steps, showing the importance of warm-up epochs.

In addition, [67] explores the impact of learning rates on the model's ability to generalize versus memorizing the training data. The study suggests that appropriately tuned learning rates can help balance this trade-off, ensuring the model converges to a solution that generalizes well to unseen data. Increasing the learning rate can introduce regularization and enhance model generalization.

CHAPTER 3

METHODOLOGY

Here, we start with an in-depth examination of the Vision Transformer (ViT) architecture [2], emphasizing its adaptation for image classification and masked image modeling. Subsequently, we explore FreezeOut [1], a technique designed to speed up the training process through layer-wise learning rate scheduling and freezing. We then introduce LocalMIM [47], an enhancement over traditional masked image modeling that leverages multi-scale and multi-stage supervision to improve semantic understanding. Finally, we present ViTFreeze, our novel approach that integrates FreezeOut with LocalMIM to optimize training efficiency and performance with our proposed learning rate adjustment technique.

3.1 Vision Transformer (ViT)

The Vision Transformer (ViT) adapts the Transformer architecture [5] for image classification tasks by converting input images into a sequence of patches. Specifically, an image $x \in \mathbb{R}^{H \times W \times C}$ is reshaped into N patches $\{x_{p_i}\}_{i=1}^{N}$, where $N = HW/P^2$ and P is the patch size. Each patch $x_{p_i} \in \mathbb{R}^{P \times P \times C}$ is then flattened and projected into a D-dimensional embedding space using a trainable linear projection, resulting in a set of patch embeddings.

3.1.1 Patch Embeddings

Patch embeddings are generated by applying a linear projection E to the flattened patches, producing the initial sequence of embeddings z_0:

$$z_0 = [x_{class}; x_{p_1}E; x_{p_2}E; \cdots; x_{p_N}E] + E_{pos} \tag{1}$$

Here, x_{class} is a learnable class token prepended to the sequence. Each flattened patch $x_{p_i} \in \mathbb{R}^{P^2 \cdot C}$ is projected using $E \in \mathbb{R}^{(P^2 \cdot C) \times D}$. The positional embeddings $E_{pos} \in \mathbb{R}^{(N+1) \times D}$ are added to maintain spatial information, resulting in the sequence $z_0 \in \mathbb{R}^{(N+1) \times D}$.

3.1.2 Transformer Encoder

The Transformer encoder processes the patch embeddings using alternating layers of multi-headed self-attention (MSA) and MLP blocks, with Layernorm (LN) and residual connections:

$$z'_\ell = MSA(LN(z_{\ell-1})) + z_{\ell-1}, \quad \ell = 1 \dots L \tag{2}$$

$$z_\ell = MLP(LN(z'_\ell)) + z'_\ell, \quad \ell = 1 \dots L \tag{3}$$

Here, $z_{\ell-1}$ is the input to the ℓ-th layer, z'_ℓ is the output after the MSA layer and residual connection, and z_ℓ is the output after the MLP block and residual connection. The total number of layers is denoted by L, with each MLP block containing two layers with GELU non-linearity.

3.1.3 Classification Head

The output state corresponding to the class token z_L^0 from the final Transformer encoder layer is used for classification. A multi-layer perceptron (MLP) with one hidden layer is employed during pre-training, while fine-tuning uses a single linear layer:

$$y = LN(z_L^0) \tag{4}$$

where $z_L^0 \in \mathbb{R}^D$ represents the state of the class token after the final layer L, and $y \in \mathbb{R}^K$ is the output logits for K classes.

3.1.4 Inductive Bias

Unlike Convolutional Neural Networks (CNNs), ViT has significantly less image-specific inductive bias. CNNs inherently capture locality and translational equivariance, while ViT relies on MLP layers for locality and self-attention mechanisms for global context. The 2D neighborhood structure is minimally used, mainly during the initial patch division and positional embedding adjustment during fine-tuning.

3.2 FreezeOut

FreezeOut modifies the backpropagation and Stochastic Gradient Descent (SGD) process to decrease training time [1]. This method uses a layer-wise cosine annealing schedule, gradually reducing each layer L_i's learning rate to zero. The learning rate $\alpha_i(t)$ for layer i at iteration t is given by:

$$\alpha_i(t) = 0.5 \times \alpha_i(0) \times (1 + \cos(\frac{\pi t}{t_i})) \tag{5}$$

where $\alpha_i(0)$ is the initial learning rate for layer i, and t_i is the iteration at which its learning rate reaches zero. Layer freezing begins at a predefined iteration t_0, with subsequent layers freezing at later iterations. Once a layer's learning rate becomes zero, it switches to inference mode. It is excluded from further backward passes, resulting in per-iteration speedup proportional to the computational cost of the frozen layer. FreezeOut also allows varying the initial layer-wise learning rates and the relation of t_0 to other layers' freezing times, providing flexibility in training prioritization.

In FreezeOut, two aspects of the training strategy are varied experimentally. Firstly, the initial layer-wise learning rate $\alpha_i(0)$ is scaled relative to the base learning rate α and the time t_i at which the layer's learning rate reaches zero, defined as:

$$\alpha_i(0) = \frac{\alpha}{t_i} \tag{6}$$

where α is the base learning rate used for the final layer, and t_i ranges between 0 and 1 (with 1 representing the entire training duration). This scaling ensures that each layer's learning curve integrates to the same value, enabling equivalent distance traversal in weight space despite fewer training steps.

Secondly, the relationship between the initial freezing time t_0 and the freezing times t_i for other layers is adjusted. A linear scheduling rule is modified by cubing the t_i values, denoted as $t_{i,\text{cubed}} = t^3_{i,\text{linear}}$. This approach prioritizes training for later layers compared to a linear schedule. Both unscaled and scaled variants are explored, where the α_i values are either identical or scaled based on the cubed t_i values. For example, a user-selected $t_0 = 0.5$ would result in $t_{0,\text{cubed}} = 0.125$. These modifications introduce flexibility in the training process, potentially reducing the need for extensive hyperparameter tuning [1].

3.3 LocalMIM

Masked Image Modeling. Masked image modeling is a self-supervised learning method that entails randomly masking portions of input data and training a model to predict the missing segments [8, 40]. Formally, given an input image $x \in \mathbb{R}^{H\times W\times C}$, it is segmented into patches $\{x_p^i\}_{i=1}^N$, where $N = HW/P^2$ represents the number of patches with patch size P. A subset of these patches $\{x_p^m\}_{m=1}^M$, where $M < N$, is masked. The objective is to predict the masked patches $\hat{x}_p^m$ using the information from the unmasked patches. This technique is similar to masked language modeling in NLP, as used in BERT [68], and allows the model to learn detailed, contextual representations of the input data without needing labeled examples. This method is especially effective in vision transformers [2], where the transformer's ability to capture global context helps to reconstruct the masked patches.

Vision Transformers in Masked Image Modeling In Vision Transformers (ViT) for masked image modeling, an image $x \in \mathbb{R}^{H\times W\times C}$ is converted into a sequence of N 2D patches $x_p \in \mathbb{R}^{N\times(P^2\cdot C)}$, where $N = HW/P^2$ and P is the patch size. These patches are then projected to D dimensions, creating patch embeddings. A learnable class token is added to the sequence, and 1D positional embeddings are included to preserve spatial information. The Transformer encoder processes the sequence using multi-headed self-attention (MSA) and linear blocks with layer normalization (LN) and residual connections. For masked image modeling, some patches are randomly masked, and the model is trained to reconstruct these masked patches, learning contextualized representations in the process. Inspired by masked language modeling, this approach enables the ViT to capture complex visual patterns and long-range dependencies in image data.

Local Masked Image Modeling. In Masked Image Modeling (MIM), the lower encoder layers are essential for pre-training, as they convey significant semantic knowledge to the upper layers. However, these initial layers often struggle to learn inter-patch semantic relationships [47], particularly with global reconstruction loss. This shows that the early layers lack image representation by focusing on feeding valuable features for the later layers, which might overwhelm the later layers and utilize the initial layers ineffectively. A multi-scale method has been proposed to improve the learning of these relationships in the initial layers, diversifying and spreading the learning task throughout the model.

Local Masked Image Modeling (LocalMIM) [47] enhances the learning of initial layers in vision transformers by using multi-scale reconstruction to guide multiple layers in the model. This approach provides a nuanced understanding of the input at various scales. They the input image into non-overlapping regions, each offering supervision signals for the reconstruction task at different scales. Figure **??** illustrates this proposed approach.

In Local multi-scale reconstruction for Masked Image Modeling (MIM), an image $x \in \mathbb{R}^{H \times W \times C}$ is divided into regions $\{x_i \in \mathbb{R}^{p \times p \times C}\}_{i=1}^{HW/p^2}$. Supervision signals $y_i = \pi(x_i)$ are obtained using the HOG feature descriptor π, providing fine-scale supervisions that capture low-level details and coarse-scale supervisions that encapsulate high-level semantics [47].

The lower layers of the model focus on reconstructing fine-scale supervision, while the upper layers handle coarse-scale supervision. It has been observed that applying the coarse-scale reconstruction task alone does not improve model performance, but combining fine-scale with coarse-scale supervision yields the best results [47]. The decoder, consisting of Transformer blocks and Deconvolution/Pool layers, adjusts the predictions to match the supervision scale across four levels.

The training loss is defined as:

$$\mathcal{L}_{LocalMIM} = -\sum_{l \in \mathcal{I}} w_l \cdot \sum_{i=1}^{N_l} m_i^l \cdot \ln P(y_i^l | \hat{y}_i^l),$$

where $\mathcal{I}$ denotes the chosen layers, w_l represents the weight of each local loss, and m_l is the mask. In LocalMIM, the losses are weighted equally. This method effectively utilizes multi-scale supervision, improving the learning process and semantic understanding at different scales.

3.4 ViTFreeze: Applying FreezeOut to LocalMIM

Our approach aims to improve the learning efficiency and speed of the initial layers in Vision Transformer (ViT) based masked image modeling. By incorporating local reconstruction tasks, these layers achieve a deeper understanding of the input data more quickly. This strategy facilitates the earlier freezing of layers while minimizing any loss in model accuracy. The combination of advanced local masked image modeling and progressive layer freezing is designed to enhance both the learning quality and computational efficiency of ViTs.

In our model, we utilize a transformer block design suitable for freezing. Specifically, in the ViT encoder model, each layer, starting from the patch embedding layer, is progressively frozen during training. For instance, the ViT-B model has 13 layers, including 12 transformer blocks and one patch embedding layer. When a layer is frozen, all its parameters, including weights and biases, are set to a non-trainable state. The unused stage decoder heads are removed after the prior encoder layers are frozen, as shown in 1

We selected a t_0 value of 0.8, followed by cubic scheduling and learning rate scaling, as our default strategy proposed in FreezeOut to be the lowest value with minimal accuracy drop. We use the cubed t_0 value of 0.512 (derived from the original $t_0 = 0.8$) as suggested in the FreezeOut paper.

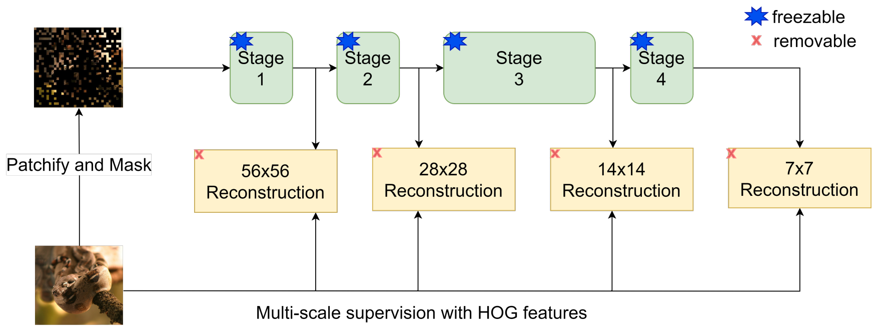

Figure 1: The proposed method involves progressively freezing layers (or stages) and eliminating decoders and reconstruction tasks when they're no longer needed. Each layer from the encoder stages is frozen individually. Once an encoder stage is frozen, the associated reconstruction task (including the decoder and hog layer) is completely removed.

The application of FreezeOut logic to ViT-B is illustrated in Figure 2. The learnable patch embedding layer is considered the first layer. There are 13 blocks, with the remaining 12 transformer encoder blocks.

Our method linearly warms up the learning rate of layers to their initial learning rate, as shown in Figure 3. Using the Adam optimizer, the learning rate gradually increases from a low value to the target rate, enhancing initial training stability. This approach initially prevents large, destabilizing weight updates, allowing for smoother and more effective convergence during the early optimization steps.

Pruning Decoders Our method further improves speed by dynamically removing intermediate decoders from the calculation once the encoder feeding them is frozen. By selectively removing these decoders, unnecessary computations are avoided, further enhancing the efficiency of the training process.

3.5 Improving LR schedule in ViTFreeze

The original FreezeOut method has a single-stage loss function, while ViTFreeze has 4 stages where the gradients can get larger for the initial stages. We speculate that the increase in gradients might be too much and can benefit from slightly adjusted learning rates.

To mitigate the potential problem of too-large gradients, we analyze the effects of different learning rate scaling methods in LocalMIM's multi-stage training approach. We hypothesize that the multi-stage approach increases the gradients too much in the initial layers. Therefore, reducing the learning rates more for early stages might enhance the performance of FreezeOut's layer-wise learning rate method.

The primary focus is on two aspects: 1. Finding the optimal base learning rate for IN-100, a smaller variation of IN-1K [69]. 2. Evaluating the performance of stage-wise learning rate reduction methods.

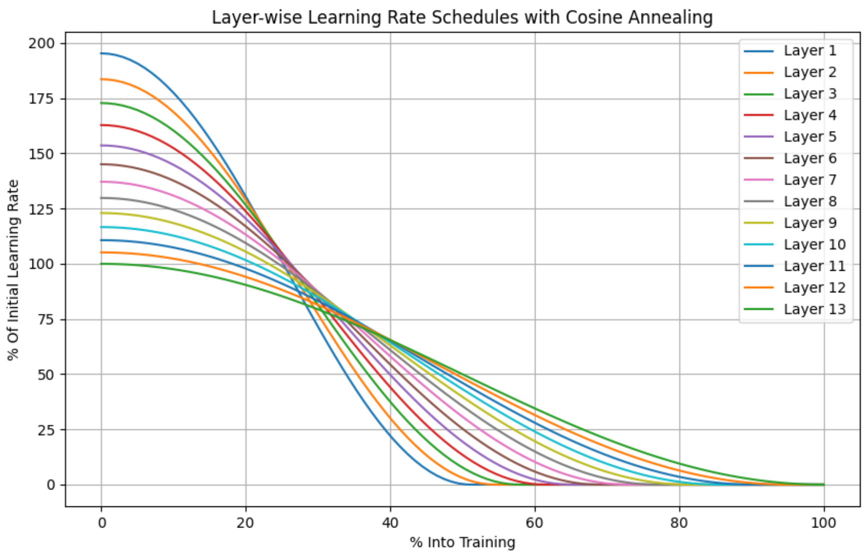

Figure 2: 13-layer cubic learning rate scheduling with $t_0 = 0.8$.

We have modified this method to better suit the multi-stage training of LocalMIM by trying two different methods.

3.5.1 Stage-wise Reduction of Learning Rate (SR-LR)

The initial learning rate is updated based on the number of output stages a layer contributes in the forward pass. This reduces the learning rate of the layers of the initial stages.

Let $s \in \mathbb{Z}^+$ denote the stage to which a layer belongs, where $S \in \mathbb{Z}^+$ is the total number of stages. Let $p \in \mathbb{Q}$ be the power hyperparameter. The optimal value of p has to be found, with larger values indicating a greater reduction in the learning rate for the initial layers. When $p = 0$, SR-LR has no impact.

The modified initial learning rate for each layer, $\alpha_i(0)$, is computed as:

$$f(s, S, p) = \left(\frac{1}{S - s + 1}\right)^p$$

$$\hat{\alpha}_i(0) = \alpha_i(0) \times f(s, S, p)$$

Where the modified FreezeOut learning rate per iteration becomes (t_i is the iteration at which its learning rate reaches zero):

$$\alpha_i(t) = 0.5 \times \alpha_i(0) \times f(s, S, p) \times \left(1 + \cos\left(\frac{\pi t}{t_i}\right)\right) \tag{7}$$

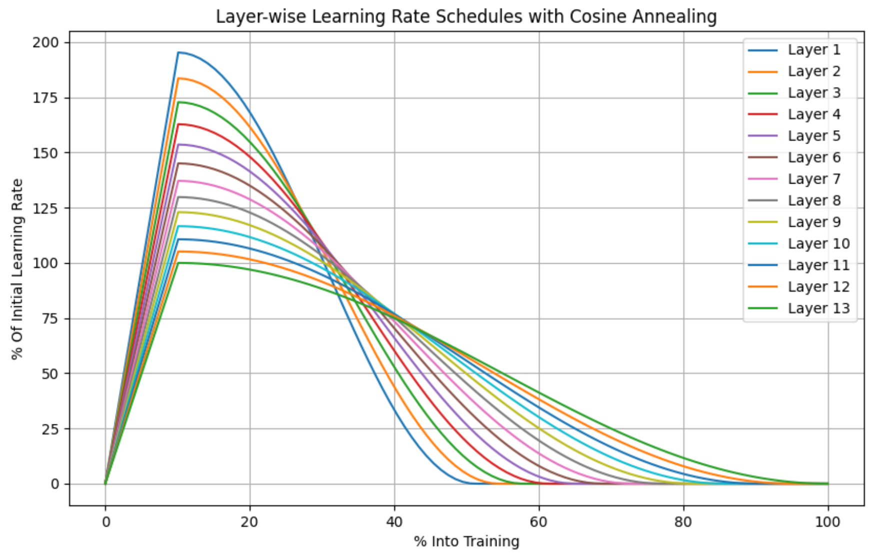

Figure 3: 13-layer cubic learning rate scheduling for $t_0 = 0.8$ with 10% warm-up.

The method calculates learning rate multipliers for each layer, denoted as L, consisting of N layers divided into S stages. $p == 0$ is equal to not applying the method.

3.5.2 Layer-wise Reduction of Learning Rate (LR-LR)

In this approach, the learning rates of the initial layers are reduced independently of the stage they belong to. The modified initial learning rate for each layer is given by:

$$\hat{\alpha}_i(0) = \frac{\alpha_i(0)}{t_i^p}$$

Where $p == 1$ is equal to not applying the method.

3.6 Analyzing Dataset Size and Training Epochs Effect

The original FreezeOut paper demonstrates its achievements on CIFAR 10 and 100 with the WideResNet model for 100 epochs. We speculate that it might be easier for this model to converge due to these reasons:

- Unlike regular classification tasks, masked image modeling can generate different supervision signals for a single image by creating different randomly masked versions of images. This might allow more prosperous learning signals to occur, which can benefit from longer training times than tasks with constant learning signals like classification.

Dataset	**Training Set**
IN-1K (ImageNet-1K)	1,281,167
IN-100 (ImageNet-100)	128,000
CIFAR-10	50,000
CIFAR-100	50,000

Table 1: Number of images in various datasets

- CIFAR 10/100 is more straightforward than IN-1K, which has 100/10 times more classes than CIFAR. If we neglect the number of iterations necessary to train for one epoch, it might be possible that training LocalMIM with the FreezeOut method might require more epochs to converge. However, increasing the number of iterations can also mitigate this problem.

Model	**Number of Parameters**
ViT Base (Vision Transformer Base)	86 million
WideResNet-50	68.9 million

Table 2: Number of parameters in ViT Base and WideResNet-50 models

- ViT architecture does not have an intrinsic inductive bias as the CNN-based WideResNet model has. Also, it has more parameters, which might require longer training time compared to WideResNet given similar setups.

To verify these assumptions and the impact of training epochs and dataset size on the performance of FreezeOut in LocalMIM, we conducted a series of experiments using the ViTFreeze model. The training setup involved variations of the ImageNet-100 dataset, with the dataset size and training duration being the primary variables.

As the requirement of better convergence was the primary investigation, the individual correctness of each argument above was not verified. Still, we analyzed whether easier convergence led to better performance. Limited compute availability and time limitations were factors in this decision.

CHAPTER 4

EXPERIMENTS

This chapter presents the experiments conducted to evaluate the effectiveness of layer freezing in accelerating the pretraining of Vision Transformers (ViTs) [2] for masked image modeling. Our primary goal is to investigate the impact of layer freezing on training efficiency and model performance. The experiments are structured to validate the benefits of layer freezing, compare different learning rate schedules, and assess the scalability of our approach across different datasets and training settings. Through these investigations, we aim to offer valuable insights into optimizing the pretraining process of ViTs.

4.1 Experimental Setup

4.1.1 Datasets

The transformer-based LocalMIM model was initially trained on the ImageNet-1K (IN-1K) [69] and the ImageNet-100 (IN-100) datasets and on variations of IN-100 that included only selected classes. To ensure comparability with the original LocalMIM paper, IN-1K was used for baseline results. Due to time and computational constraints, smaller subsets of IN-100 were utilized for extensive hyperparameter tuning in some experiments. Each class has 1300 images, and the dataset can be determined by multiplying this with the number of classes. IN-X notation is utilized here, where X represents the number of classes, where the classes have been chosen randomly.

Experiments involving the original FreezeOut method with WideResNets were conducted exclusively on the CIFAR-100 dataset.

Table 3: Class Codes for ImageNet 1K Variations

Dataset	**Class Codes**
IN-5	n01443537, n01496331, n01773157, n01978455, n01984695
IN-10	n01498041, n01644900, n01664065, n01695060, n01729977, n01770081, n01798484, n01847000, n01883070, n02002556
IN-25	n01498041, n01514668, n01592084, n01632458, n01632777, n01667778, n01687978, n01698640, n01728572, n01729322, n01735189, n01742172, n01756291, n01775062, n01796340, n01798484, n01818515, n01824575, n01828970, n01843383, n01847000, n01944390, n01950731, n01968897,

Table 3: (continued)

Dataset	**Class Codes**
	n02028035
IN-100	n01440764, n01443537, n01484850, n01491361, n01494475, n01496331, n01498041, n01514668, n01514859, n01531178, n01537544, n01560419, n01582220, n01592084, n01601694, n01608432, n01614925, n01622779, n01630670, n01632458, n01632777, n01644900, n01664065, n01665541, n01667114, n01667778, n01675722, n01677366, n01685808, n01687978, n01693334, n01695060, n01698640, n01728572, n01729322, n01729977, n01734418, n01735189, n01739381, n01740131, n01742172, n01749939, n01751748, n01753488, n01755581, n01756291, n01770081, n01770393, n01773157, n01773549, n01773797, n01774384, n01774750, n01775062, n01776313, n01795545, n01796340, n01798484, n01806143, n01818515, n01819313, n01820546, n01824575, n01828970, n01829413, n01833805, n01843383, n01847000, n01855672, n01860187, n01877812, n01883070, n01910747, n01914609, n01924916, n01930112, n01943899, n01944390, n01950731, n01955084, n01968897, n01978287, n01978455, n01984695, n01985128, n01986214, n02002556, n02006656, n02007558, n02011460, n02012849, n02013706, n02018207, n02018795, n02027492, n02028035, n02037110, n02051845, n02058221, n02077923

4.1.2 Evaluation Metrics

For the CIFAR-100 dataset, the validation error metric was employed to enable direct comparison with results from the original paper. For the IN-1K dataset, both Top-1 accuracy and Top-5 accuracy metrics were used.

Top-1 accuracy, also known as accuracy, refers to the percentage of times the model's highest-confidence prediction matches the true label. It is a stringent metric indicating how often the model predicts the correct class in a single guess. Top-5 accuracy, on the other hand, measures the percentage of times the true label is among the model's top five predictions. This metric is useful for evaluating model performance in scenarios with multiple similar classes, providing a more lenient measure of model accuracy.

4.1.3 Experimental Settings

All experiments were implemented in Python using PyTorch on NVIDIA A100 GPUs (unless stated otherwise), each with 64 GB of RAM. The experiments were conducted using CUDA Version 12.1 and PyTorch Version 1.13.1.

4.2 Validating FreezeOut Method

Here, we investigate the effectiveness of the original FreezeOut method by comparing the performance of a ResNet [15] model trained with and without FreezeOut. The training procedure for both scenarios is the same as the original FreezeOut paper as follows:

- **Model:** ResNet
- **Optimizer:** SGD
- **Weight Decay:** 0.0001
- **Learning Rate:** 0.1
- **Epochs:** 100

We aim to determine whether FreezeOut can reduce training time while maintaining or improving model performance.

The experiments involve training the ResNet model under three different configurations for each optimizer (six experiments in total). Here we only consider SGD training without learning rate scaling alterations and leave those experiments to the following sections:

Model	**Optimizer**	**Method**	**FreezeOut t_0**	**Epochs**	**Scale Learning Rate**
ResNet	SGD	FreezeOut	0.8	100	No
ResNet	AdamW	FreezeOut	0.8	100	No
ResNet	SGD	FreezeOut	0.8	100	Yes
ResNet	AdamW	FreezeOut	0.8	100	Yes
ResNet	SGD	No FreezeOut	1.0	100	-
ResNet	AdamW	No FreezeOut	1.0	100	-

Table 4: Summary of Experiments Involving ResNet, Different Optimizers, and FreezeOut Method

The models' performance is evaluated based on the validation error at different epochs, focusing mainly on 100 epochs to see if FreezeOut allows the model to achieve similar performance in fewer epochs.

The following results in 5 were obtained from the experiments (AdamW experiments are left for the following subsection to compare them).

4.2.1 Brief Analysis

The results demonstrate that the FreezeOut method can effectively reduce the training time while achieving comparable or better performance. Specifically, the model trained with FreezeOut achieves its best validation error earlier than the one trained without. This suggests that FreezeOut can be a viable technique for accelerating the training process of deep neural networks without sacrificing performance.

Model	**Optimizer**	**Method**	**Epoch**	**Validation Error (%)**
ResNet	SGD	FreezeOut ($t_0 = 0.8$)	100	21.72
ResNet	SGD	FreezeOut ($t_0 = 0.8$)	96	21.46
ResNet	SGD	FreezeOut ($t_0 = 0.8$)	80	22.30
ResNet	SGD	No FreezeOut ($t_0 = 1.0$)	100	21.98
ResNet	SGD	No FreezeOut ($t_0 = 1.0$)	80	23.58

Table 5: Validation errors at different epochs for ResNet models trained with SGD under different methods.

By comparing the difference between 80 and 100 epoch-trained models with and without FreezeOut, it is visible that the model trained with FreezeOut converges earlier. This suggests that further training time reduction is possible with early stopping.

4.3 Analyzing FreezeOut LR Schedule for SGD and AdamW

4.3.1 Experiment Goal

The primary objective of this study is to evaluate the effectiveness of FreezeOut's learning rate scheduling compared to regular learning rate scheduling. The learning rate scheduling in FreezeOut may not be effective, as the improvements shown in the FreezeOut paper are subtle. The effectiveness might also change for different optimizers, where SGD is used in the FreezeOut paper, and AdamW is used in LocalMIM.

4.3.2 Experiment Setup

We implemented both FreezeOut and regular learning rate scheduling for the original FreezeOut paper using ResNet to test this. The experiments were conducted with SGD and AdamW optimizers to compare their performance under different learning rate scheduling methods. As the default hyperparameters were optimized for SGD, a grid search was undertaken to find AdamW-optimized hyperparameters. To see how FreezeOut affects the model under suboptimal learning rate and weight decay, the AdamW-optimized setup was also applied to SGD model training.

Given that the recommended learning rate (LR) for AdamW is 0.001, we adopted this value for our experiments. The optimal weight decay (WD) was determined through preliminary trials in 6.

Based on these results, we used WD = 0.001 for AdamW experiments. The experiments conducted for both optimizers are analyzed below.

4.3.3 AdamW Experiments

Experiments conducted with the AdamW optimizer are shown in 7.

Optimizer	Learning Rate (LR)	Weight Decay (WD)	Validation Error (%)
AdamW	0.001	0.1	Did not converge
AdamW	0.001	0.05	Did not converge
AdamW	0.001	0.01	Did not converge
AdamW	0.001	0.0005	31.82
AdamW	0.001	0.005	50.84
AdamW	0.001	0.001	30.60

Table 6: Preliminary trials to determine the optimal weight decay (WD) for AdamW optimizer with a fixed learning rate (LR) of 0.001.

Table 7: Validation Errors for Different Experiments with AdamW Optimizer

Experiment	Epoch 80	Epoch 100	Best Validation Error
No FreezeOut	34.04	31.42	31.26 (Epoch 99)
FreezeOut t_0=0.8	31.08	30.7	30.58 (Epoch 85)
No scale LR, FreezeOut t_0=0.8	34.14	33.12	32.94 (Epoch 95)

4.3.4 SGD Experiments with SGD Optimized LR (0.1) and WD (0.0001)

Experiments conducted with the SGD optimizer using the original FreezeOut weight decay and learning rate are shown in 8.

Table 8: Validation errors for different experiments with SGD optimizer. The original hyperparameters from the FreezeOut method was used (optimized for ResNet).

Experiment	Epoch 80	Epoch 100	Best Validation Error
No FreezeOut	23.58	21.98	21.98 (Epoch 100)
FreezeOut t_0=0.8	22.3	21.72	21.46 (Epoch 96)
No scale LR, FreezeOut t_0=0.8	21.94	21.16	21.10 (Epoch 87)

4.3.5 SGD Experiments with AdamW Optimized LR and WD

In 9 the experimental results indicate that FreezeOut learning rate scheduling is somewhat inferior to regular learning rate scheduling when using the SGD optimizer. It only works without the learning rate scheduling. However, FreezeOut appears to be slightly more effective for the AdamW optimizer.

4.3.6 Brief Analysis

The observations from these experiments suggest that the effectiveness of FreezeOut learning rate scheduling depends on the optimizer used. For SGD, regular learning rate scheduling appears to be

Table 9: Validation errors for different experiments with SGD optimizer. The parameters optimized for AdamW were used.

Experiment	Epoch 80	Epoch 100	Best Validation Error
No FreezeOut	34.12	23.04	22.8 (Epoch 99)
FreezeOut t_0=0.8	28.12	24.08	23.38 (Epoch 91)
No scale LR, FreezeOut t_0=0.8	25.32	22.76	22.76 (Epoch 91)

Table 10: Lowest validation errors achieved by different optimizers and FreezeOut configurations. SGD* uses AdamW-optimized hyperparameters, while SGD uses the original SGD-optimized hyperparameters.

Configuration	Lowest Validation Error	Epoch
SGD + no FreezeOut (t_0=1)	21.98	100
SGD + FreezeOut (t_0=0.8)	21.46	96
SGD + FreezeOut (t_0=0.8) (no scale LR)	21.10	87
SGD* + no FreezeOut (t_0=1)	22.8	99
SGD* + FreezeOut (t_0=0.8)	23.38	91
SGD* + FreezeOut (t_0=0.8) (no scale LR)	22.76	98
AdamW + no FreezeOut (t_0=1)	31.26	99
AdamW + FreezeOut (t_0=0.8)	30.58	85
AdamW + FreezeOut (t_0=0.8) (no scale LR)	32.94	95

more effective. However, for AdamW, FreezeOut demonstrates a slight improvement in validation error.

These results are consistent with the findings reported in the original FreezeOut paper, which indicated insignificant to no improvement from using cubic and linear learning rate scaling methods compared to no scaling for both WideResNets and DenseNets while using SGD [1]. Thus, the benefits of learning rate scaling methods remain ambiguous and may vary based on different experimental setups and configurations.

Repeating the experiments with the original SGD setup without FreezeOut learning rate scaling further supports the argument that FreezeOut can be inferior under certain conditions.

Intriguingly, AdamW achieves its top performance in earlier epochs than SGD, mostly. This shows that AdamW is converging faster than SGD in most setups as suggested in the paper [11].

4.4 ViTFreeze: Applying FreezeOut to LocalMIM

Here, we explore the impact of FreezeOut on local masked image modeling (LocalMIM) pre-training for Vision Transformers (ViTs). We hypothesize that FreezeOut accelerates the pre-training process without significant performance degradation. As stated before, the learning rate scheduling might not be superior for ViTFreeze either.

Table 11: Comparing 100 epoch fine-tuning accuracy on ImageNet-1K with LocalMIM ViT-B with 50, 70, 90, and 100 epoch pre-training. Bold indicates faster or better. All setups' training time were measured on the same device (a single V100 GPU).

Model	**PT Epoch**	**Hours/Ep.**	**Total Hours**	**Top-1**	**Top-5**
LocalMIM-HOG	**50**	0.48	**24**	83.09	96.45
LocalMIM-HOG	70	0.48	33.6	83.17	96.45
LocalMIM-HOG	90	0.48	43.2	**83.38**	96.45
LocalMIM-HOG	100	0.48	48	83.2	**96.6**
ViTFreeze	100	**0.42**	42	82.66	96.18
ViTFreeze w/o LR scaling	100	**0.42**	42	82.94	**96.36**

Table 12: Comparing 100 epoch fine-tuning accuracy and different pre-training epoch losses on ImageNet-1K with LocalMIM ViT-B. Bold indicates better. Even though the 100-epoch pre-trained model has lower reconstruction error, it has worse Top-1 accuracy than the 90-epoch pre-trained model.

Model	**PT Epoch**	**Top-1**	**Top-5**	**Pre-training loss**
LocalMIM-HOG	**50**	83.09	96.45	0.1111
LocalMIM-HOG	70	83.17	96.45	0.1093
LocalMIM-HOG	90	**83.38**	96.45	0.1080
LocalMIM-HOG	100	83.2	**96.6**	0.**1078**

To evaluate the effectiveness of FreezeOut, we conducted the following experiments:

- **Baseline Experiment:** Pre-training a ViT model with standard local masked image modeling for 100 epochs without any FreezeOut or layer-wise learning rate scheduling.
- **FreezeOut Experiment:** Pre-training a ViT model with FreezeOut applied at $t_0 = 0$ for 100 epochs.
- **Epoch Reduction Experiment:** Pre-training the ViT model for fewer epochs (50, 60, 70, etc.) without FreezeOut to compare the results against the FreezeOut model.

The preliminary results of our experiments are summarized as follows:

FreezeOut Experiment Results:

- Applying FreezeOut at $t_0 = 0$ for 100 epochs reduced pre-training time by 12.5%.
- Without layer-wise learning rate scheduling, the accuracy drop was minimal (0.2%).

4.4.1 Brief Analysis

Our experiments demonstrate that FreezeOut can accelerate the pre-training process for LocalMIM in ViTs without significant performance degradation. The preliminary results show a 12.5% reduction in pre-training time with only a minor drop in accuracy (0.26% without using LR-scaling). This suggests that FreezeOut is an effective technique for speeding up pre-training.

The impact of layer-wise learning rate scheduling on accuracy is still under evaluation. Initial observations suggest that it might harm accuracy, aligning with our hypothesis. However, more comprehensive experiments are needed to draw definitive conclusions.

Since model pertaining benefits not only from model convergence but from the model learning weights that will translate well to downstream tasks (a generalization of the model), longer pre-training might not always produce the optimal results in downstream tasks, as the model might overfit to the pre-training task instead of learning generalizable features.

Likewise, the upstream task loss shows no clear correlation between how well the model converges during pre-training and how well it performs in the downstream classification task. Even though the 90-epoch pre-training loss is slightly more than the model trained with 100 epochs for the 100-epoch LocalMIM training setup, the results of the 90-epoch pre-trained model in the classification downstream task are superior. This might be unique to the downstream task or LocalMIM's fast converging architecture, which should be considered a separate study.

4.5 Improving LR schedule in ViTFreeze

Due to the extensive nature of the IN1K dataset, experiments were conducted on IN-100 to evaluate different learning rate scaling methods.

After finding suitable values for the proposed method, the experiment was repeated once for the original IN-1K setup.

4.5.1 Finding Optimal Base Learning Rate

Previous results for FreezeOut with AdamW on the CIFAR dataset showed that FreezeOut can be sensitive to learning rate. A grid search was performed to identify the optimal base learning rate, as the default rate of 0.0002 for IN1K was suboptimal for IN-100. A base learning rate of 0.0001 was found to be better, achieving an accuracy of 86.56%. This value has been used for the rest of the experiments with IN-100.

4.5.2 Analyzing Learning Rate Scaling Methods

Using the optimal learning rate, the following values for SR-LR were tested. First SR-LR 14, then LR-LR 15 and then their combination 16 were tested. The optimal values for SR-LR were kept while the LR-LR hyperparameter was tuned in the combined setup.

The experiments showed that SR-LR with a power parameter of 0.8 yielded the most promising results 14. LR-LR also yielded better results than regular ViTFreeze training 15. When combining stage layer LR scaling and LR power-based reduction, the results were not as good as expected 16.

Table 13: Testing different base learning rates for top-1 accuracy with cubic learning rate scaling using AdamW on ViT-B on IN-100.

t_0	Base Learning Rate	Top-1 Accuracy (%)
1.0	1.0e-4	**86.56**
1.0	1.2e-4	85.32
1.0	1.4e-4	72.94
1.0	2e-4	80.62
1.0	2.2e-4	75.78
1.0	2.4e-4	75.82
1.0	2.6e-4	50.54

Table 14: Testing different SR-LR values with t_0= 0.8 for top-1 accuracy with cubic learning rate scaling using AdamW on ViT-B with base learning rate 1.0e-4 on 1N-100.

SR-LR Multiplier	Top-1 Accuracy (%)
0.2	83.36
0.4	83.96
0.6	85.70
0.8	**86.44**
0.9	86.12
0.75	86.14
0.85	86.30
1.0	85.74
1.2	85.60

4.5.3 Applying SR-LR to ViTFreeze on IN1K

The effectiveness of SR-LR was tested on the original IN1K training setup. Its superiority is reflected in the results in 17.

4.5.4 Brief Analysis

As demonstrated in the results in 14, the hypothesis that reducing the learning rates more for early stages improves the performance of the FreezeOut layer-wise learning rate method was verified. The stage-dependent layer learning rate scaling method, particularly with a power parameter of 0.8, showed improved performance.

The LR-LR method improved the results individually in 15, but slightly less than SR-LR. This makes sense, as the gradient increase should be stage-wise in the multi-stage architecture. The combination of SR-LR and LR-LR did not provide a further performance boost, showing that the reduction from SR-LR was sufficient.

Table 15: Testing different Layer-wise Reduction of Learning Rate (LR-LR) values for top-1 accuracy with cubic learning rate scaling using AdamW on ViT-B with base learning rate 1.0e-4 on IN-100.

LR-LR Multiplier	Top-1 Accuracy (%)
0.2	85.40
0.3	85.68
0.4	82.98
0.5	83.90
0.6	85.60
0.7	**85.82**

Table 16: Testing different Layer-wise Reduction of Learning Rate (LR-LR) values where SR-LR is 0.8 on IN-100. Reporting top-1 accuracy with cubic learning rate scaling using AdamW on ViT-B with base learning rate 1.0e-4.

LR-LR Multiplier	Top-1 Accuracy (%)
0.2	85.78
0.3	85.78
0.4	85.92
0.5	85.88
0.6	**86.10**

The superior results of SR-LR for the IN-100 dataset translated well to the IN-1K dataset as demonstrated in 17. There is an **82%** reduction in top-1 accuracy drop compared to the bare FreezeOut method applied to LocalMIM (ViTFreeze) and **62%** reduction in top-1 accuracy drop compared to the FreezeOut method without LR Scaling applied to LocalMIM (ViTFreeze without LR Scaling).

Unlike 12, the results for 100 epoch pre-training in 18 show a correlation between the upstream task loss and the downstream task performance. This behavior aligns with theoretical expectations and makes it easier to conclude in this study. This was not the case for the early epoch success of LocalMIM. Hence, its incoherent nature requires further investigation.

The findings from these experiments contribute to our understanding of effective learning rate scheduling in multi-stage training setups and provide a foundation for further improvements in the training of multi-stage Vision Transformers with layer-freezing techniques.

4.6 Analyzing Dataset Size and Training Epochs Effect

4.6.1 Dataset Creation

The ImageNet-100 dataset was subsetted into minor variations containing 5, 10, 25, and 100 classes. This was done to simulate varying levels of complexity, assuming that as the number of classes in-

Table 17: Comparing 100-epoch pre-training followed by fine-tuning accuracy on ImageNet-1K with LocalMIM and ViTFreeze for ViT-B. Bold indicates faster or better. Total GPU hours can be found by multiplying GPU Hours/ep. by 100.

Model	**GPU Hours/Ep.**	**Top-1**	**Top-5**	**Top-1 Drop**
LocalMIM-HOG	0.48	83.2	**96.6**	0
ViTFreeze	**0.42**	82.66	96.18	0.54
ViTFreeze w/o LR scaling	**0.42**	82.94	**96.36**	0.26
ViTFreeze w/ SR-LR	**0.42**	**83.10**	96.33	**0.1**

Table 18: Comparing 100 epoch fine-tuning accuracy and 100-epoch pre-training epoch losses on ImageNet-1K with LocalMIM ViT-B. Bold indicates better. The correlation between pre-training losses and fine-tuning accuracies is not apparent.

Model	**PT Epoch**	**Top-1**	**Top-5**	**Top-1 Drop**	**Pre-training loss**
LocalMIM-HOG	0.48	**83.2**	**96.6**	0	**0.0083**
ViTFreeze	**0.42**	82.66	96.18	0.54	0.0087
ViTFreeze w/o LR scaling	**0.42**	82.94	**96.36**	0.26	**0.0085**
ViTFreeze w/ SR-LR	**0.42**	**83.10**	96.33	**0.1**	**0.0085**

creases, the complexity and difficulty of achieving convergence also increase. The selected classes were sampled without considering intra-class complexity.

4.6.2 Training Setup

The ViTFreeze model was trained for different durations for each dataset, specifically chosen to represent close-to-edge cases to reduce the number of experiments. Training durations were selected to be either five times longer or five times shorter than a baseline epoch duration. The iteration count and batch size were kept constant for each training set-up to ensure consistency.

4.6.3 Hyperparameters

The following hyperparameters were used during training:

Table 19: Hyperparameters Used During Training

Hyperparameter	**Value**
t_0	0.8,1.0
LR-SR value	0.0, 0.8

The experiments were structured to analyze the effect of varying dataset sizes and training epochs on the performance of FreezeOut. The main variables were the dataset size and the number of training epochs. As the quality of the transferred representation from the pre-training task can not be solely evaluated by a strict number of fine-tuning iterations, different numbers of epochs for fine-tuning were used for evaluation.

4.6.4 Dataset Variations

The dataset variations used in the experiments included subsets of ImageNet-1K [69] with 5, 10, 25, and 100 classes.

4.6.5 Training Durations

The training durations were set to 5 times longer or five times shorter than a baseline duration, ensuring a range of training epochs to observe the effects on FreezeOut.

4.6.6 Performance Metrics

The performance of the ViTFreeze model was evaluated using accuracy metrics at various stages of training. The results were recorded for each combination of dataset size and training duration.

The following tables summarize the performance results of the ViTFreeze model across different dataset sizes and training durations. Each entry represents the accuracy of the model at various training stages.

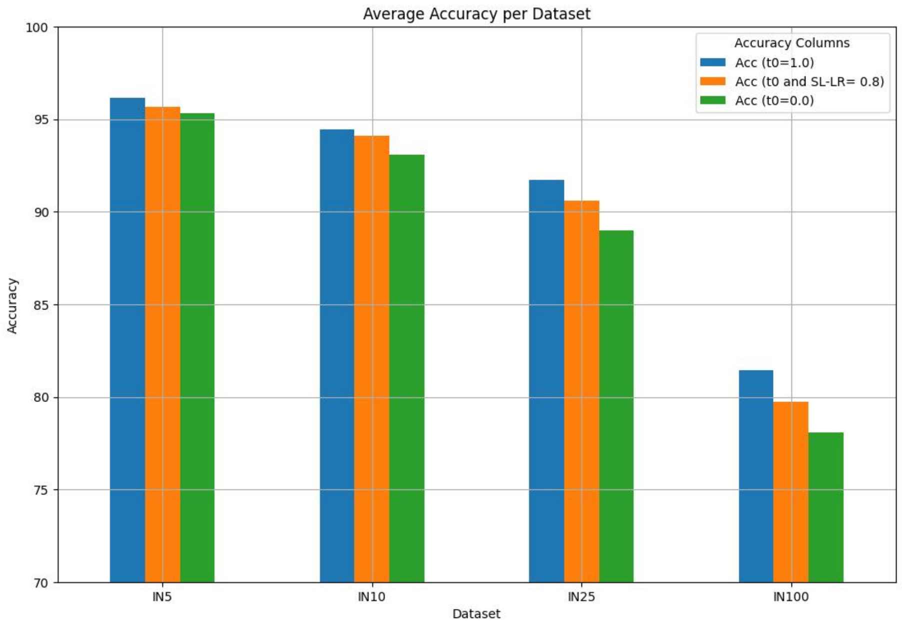

Figure 4: Dataset Size

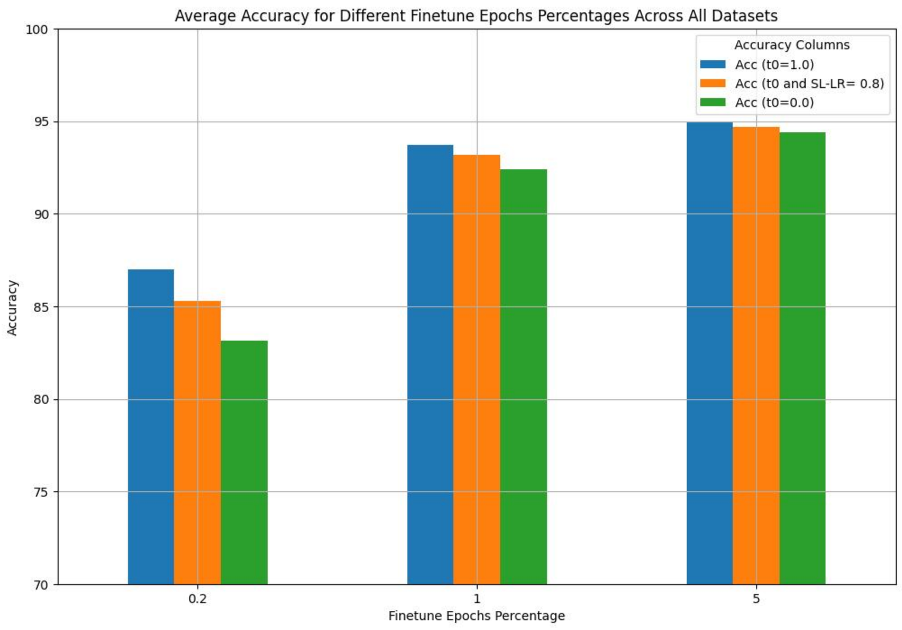

Figure 5: Finetune Epoch Graphs

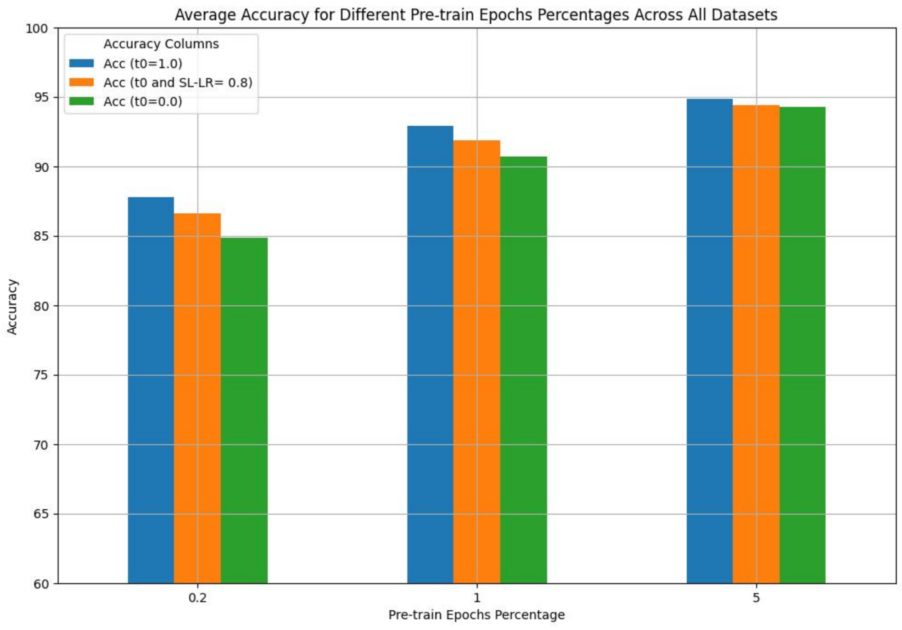

Figure 6: Pretrain Epoch Graphs

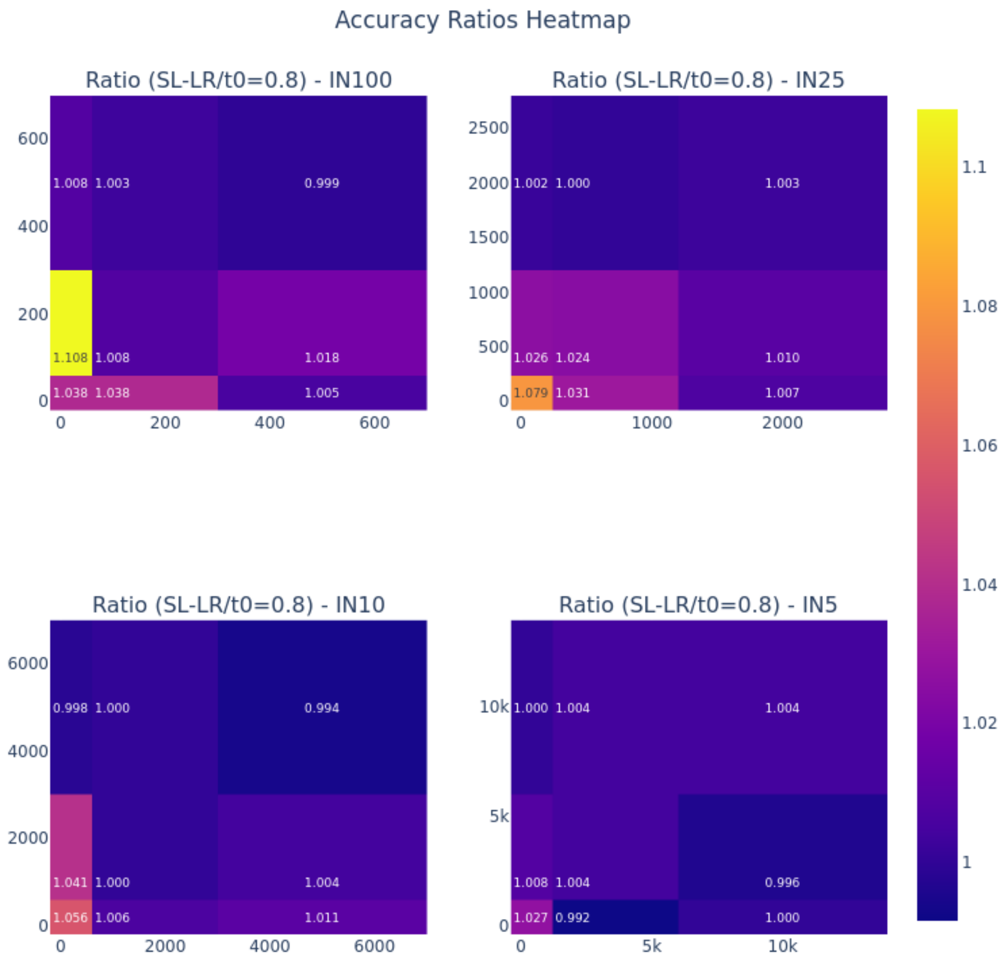

Figure 7: Ratio (SR-LR, t_0=0.8)

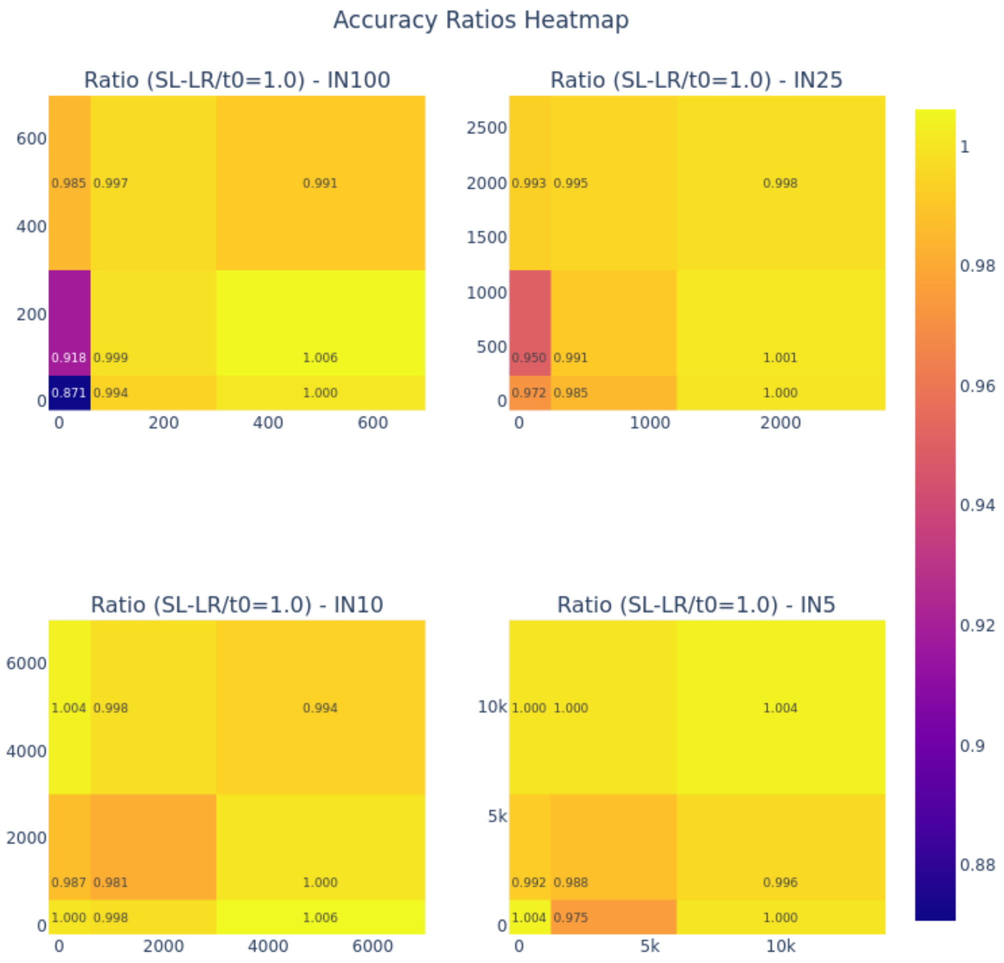

Figure 8: Ratio (SR-LR, t_0=1.0)

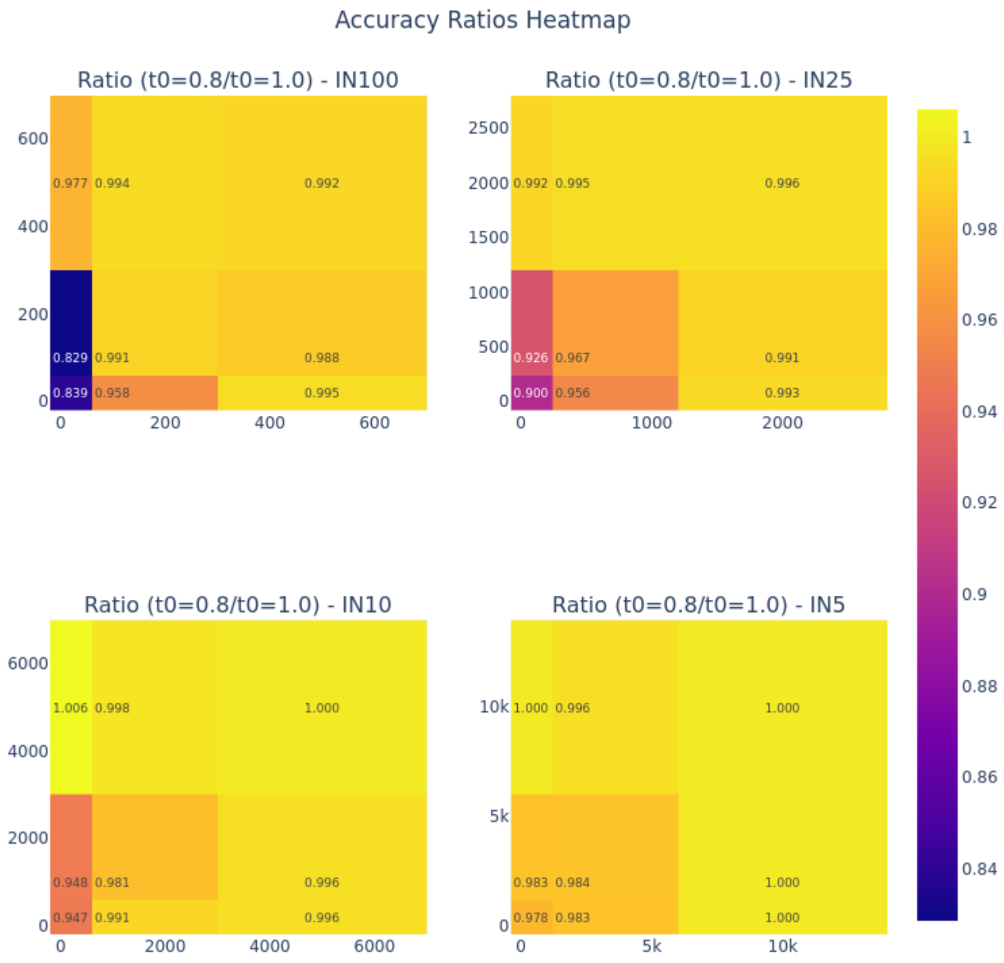

Figure 9: Ratio (t_0=0.8, t_0=1.0)

4.6.7 Brief Analysis:

The results indicate that the performance of FreezeOut is significantly influenced by both the dataset size and the number of training epochs. Key observations include:

- Dataset complexity negatively affects the performance of all models regardless of the setup (4)).
- As the dataset complexity increases from IN5 to IN100, the models' performance with layer freezing degrades faster than regular training. This shows that when keeping the number of iterations constant, FreezeOut can worsen the performance of more complex datasets (4).
- All experiments benefited from longer pertaining and training epochs (5 and 6).
- The models trained with FreezeOut require longer pre-training or fine-tuning to match the performance of regular training (5 and 6).
- The SR-LR method helps FreezeOut maintain performance, especially for lower fine-tuning or pre-training epochs. This becomes more noticeable in complex datasets (7).
- FreezeOut with SR-LR can still be inferior to regular training in more complex datasets, especially for lower pre-training and fine-tuning epochs (8).

These show that FreezeOut combined with SR-LR is more robust to more complex datasets and lower pre-training or fine-tuning epochs than FreezeOut. However, each variation of FreezeOut seems to benefit from longer training epochs. This suggests the possibility of ViTFreeze with SR-LR achieving better results for longer pre-training or fine-tuning epochs.

Indeed, training ViTFreeze with SR-LR on IN-1K with the original setup yielded superior performance compared to regular ViTFreeze 17. There is a small difference of 0.1% accuracy drop compared to the initial 0.54% drop of the original ViTFreeze.

Dataset	Pre. Ep.	Fine. Ep.	Acc ($t_0 = 0.8$)	Acc (t_0 and SR-LR= 0.8)	Acc (t_0=1.0)
IN100	500	500	88.62	88.56	**89.36**
IN100	500	100	88.4	88.64	**88.94**
IN100	500	20	82.04	82.72	**84.0**
IN100	100	500	87.16	**88.76**	88.26
IN100	100	100	85.74	86.44	**86.56**
IN100	100	20	60.32	66.86	**72.8**
IN100	20	500	87.4	**87.88**	87.84
IN100	20	100	77.68	80.62	**81.1**
IN100	20	20	45.28	47.0	**53.98**
IN25	2000	2000	95.76	96.0	**96.16**
IN25	2000	400	95.76	95.76	**96.24**
IN25	2000	80	93.76	93.92	**94.56**
IN25	400	2000	94.56	**95.52**	95.44
IN25	400	400	91.44	93.68	**94.56**
IN25	400	80	81.36	83.44	**87.84**
IN25	80	2000	94.24	**94.88**	**94.88**
IN25	80	400	87.68	90.4	**91.76**
IN25	80	80	66.56	71.84	**73.92**
IN10	5000	5000	**97.8**	97.2	**97.8**
IN10	5000	1000	97.6	97.6	**97.8**
IN10	5000	200	**96.8**	96.6	96.2
IN10	1000	5000	96.4	**96.8**	**96.8**
IN10	1000	1000	94.8	94.8	**96.6**
IN10	1000	200	88.0	91.6	**92.8**
IN10	200	5000	95.2	**96.2**	95.6
IN10	200	1000	92.4	93.0	**93.2**
IN10	200	200	78.8	**83.2**	**83.2**
IN5	10000	10000	97.2	**97.6**	97.2
IN5	10000	2000	96.8	**97.2**	**97.2**
IN5	10000	400	**96.4**	**96.4**	**96.4**
IN5	2000	10000	**97.2**	96.8	**97.2**
IN5	2000	2000	95.6	96.0	**97.2**
IN5	2000	400	94.8	95.6	**96.4**
IN5	400	10000	**96.8**	**96.8**	**96.8**
IN5	400	2000	95.2	94.4	**96.8**
IN5	400	400	88.0	**90.4**	90.0

Table 20: Performance results of ViTFreeze model on various dataset sizes and training durations. t_0=1.0 means there is no FreezeOut. Top-1 accuracy is demonstrated. "Pre. Ep." stands for pretrain epochs, while "Fine. Ep." stands for finetune epochs.

CHAPTER 5

DISCUSSION

Here we discuss the theoretical findings and implications of the experimental results in more depth. Each experiment is evaluated separately by linking interpretations across them.

5.1 Validating FreezeOut Method

The method works in the original WideResNet-based FreezeOut setup, achieving slightly better results than the typically trained model. It is also visible that the model trained with FreezeOut converges earlier than the customarily trained model, possibly due to the learning rate increment in FreezeOut.

5.2 Analyzing FreezeOut LR Schedule

The FreezeOut LR Schedule does not improve performance in general, and it might even reduce it. This has been observed mainly for SGD, even though it was the optimizer used in the FreezeOut paper.

5.3 ViTFreeze: Applying FreezeOut to LocalMIM

The model without FreezeOut learning rate scaling achieves results close to the original training setup. The ability of LocalMIM to achieve better results with fewer epochs shadows the effectiveness of FreezeOut for LocalMIM for the 100-epoch pretraining setup. However, we consider the early convergence behavior of LocalMIM a subject of another research based on these possible explanations:

- The goal of model pre-training is not model convergence only; the aim is also to achieve a model initialization that has more potential (more generalizable) on downstream tasks. Hence, evaluating the FreezeOut method's impact on the pre-training task performance might be better. This has been supported by "Exploring the Limits of Large Scale Pre-training" (Abnar et al.), where it has been shown that better upstream task performance might not translate to better downstream task performance.

- 100 epochs might be insufficient for FreezeOut to work fine as the method benefits from further convergence and longer training epochs. Hence, for longer pre-training epochs, it might be more difficult for early training epochs to surpass the fully trained ViTFreeze model.
- The classification task might favor features produced with fewer training epochs, while other downstream tasks might behave differently. Other downstream tasks might prefer more convergence in the upstream task.

As the results from the lower pre-training epoch are unexpected, they need further investigation on more downstream tasks and further training epoch setups. The superior early epoch might be misleading due to the points made above. Hence, in the given isolated setup for 100-epoch pre-training, ViTFreeze shows promising results with minimal performance degradation and a decent speed-up.

Even if the early epoch weight of the pre-training task has superior performance in general, the ViTFreeze method removes the need to search for the optimal pre-training epoch for the pre-training task. This advantage can also apply to other downstream tasks and datasets.

5.4 Improving LR schedule in ViTFreeze

The SR-LR method improved ViTFreeze's performance further, achieving very close results to the original 100-epoch pre-trained LocalMIM. It reduced the accuracy drop by **82%** and **62%** for bare ViTFreeze and ViTFreeze w/o LR Scaling. Regarding upstream task performance, ViTFreeze with SR-LR achieves slightly better results than without applying learning rate scaling. As mentioned, we keep the complex interpretation of better early epoch pre-training results out of our scope.

5.5 Analyzing Dataset Size and Training Epochs Effect

Several conclusions can be made in this study:

- The FreezeOut method benefits from easy convergence. Hence, this method can shine in longer pre-training or fine-tuning epochs and simpler datasets.
- SR-LR mainly compensates FreezeOut's performance drop in the difficult-to-converge cases (lower training and complex datasets). Making it more robust in diverse and challenging training setups.

5.6 Limitations

We had limited time and GPU for the experiments. Hence, the dataset size, the model, and the number of training epoch parameters had to be kept within acceptable constraints.

CHAPTER 6

CONCLUSION

This research aimed to speed up Vision Transformers (ViTs) pre-training for masked image modeling by progressively freezing layers. Our goal was to achieve this acceleration with minimal performance drop. To this end, we developed and validated the FreezeOut method using local masked image modeling (LocalMIM) and explored different learning rate scaling methods.

6.1 Achievements of the Study

6.1.1 Methodology and Implementation

- We employed the FreezeOut method with LocalMIM, progressively freezing layers to reduce computational demands.
- Various learning rate scaling methods were tested to find the most effective approach for our experiments.
- Both SGD and AdamW optimizers were used to validate the effectiveness of FreezeOut.

6.1.2 Validation and Results

- FreezeOut demonstrated significant efficiency, achieving a notable reduction in training time by approximately 12.5% with minimal performance drop.
- The results showed that FreezeOut performed better with AdamW than with SGD. Most probably due faster convergence of AdamW.
- By using Stage-wise Reduction of Learning Rate (SR-LR), further improvements in training performance were achieved.
- Fine-tuning early epoch weights for the pre-training tasks achieved considerable results (for 50-70-90 epochs). Still, it was kept out of the scope of this work due to a lack of data and incoherent results, as the 90 epoch pretrained LocalMIM even surpassed the performance of 100 epoch pretrained LocalMIM in the downstream classification task, without having superior performance (lower loss) in the upstream reconstruction task.w

6.1.3 Implications and Insights

- FreezeOut can be effectively applied to ViTs or other architectures with internal skip connections.
- The method offers potential performance boosts with minimal degradation, especially with longer training epochs and simpler datasets.
- SR-LR works better than bare FreezeOut when training for fewer epochs and improves training performance in diverse settings.

6.2 Questions for Future Research

Further questions have arisen from the observations in this work. These related questions here show potential directions:

- Does FreezeOut achieve superior results with larger datasets, training durations, and models?
- Would it be better to apply FreezeOut solely to model fine-tuning rather than pre-training, as it is more common to freeze initial layers for fine-tuning?
- Why and how does LocalMIM converge this much faster than regular ViT MAE training?
- How does FreezeOut affect self-supervised learning performance on other downstream tasks?
- How does regular ViT MAE training perform when FreezeOut is applied? Do early epoch checkpoints of pre-trained weights still achieve good results?
- Could curriculum learning be applied to FreezeOut LocalMIM training by focusing more on the initial stage reconstruction task and progressively giving more weight to the final reconstruction task to increase the convergence speed and model performance?

6.3 Concluding Remarks

This thesis presents an innovative approach to enhancing the efficiency of Vision Transformers in self-supervised learning. We addressed the challenge of lengthy training times and high computational demands by integrating local masked image modeling with progressive layer freezing. Our findings underscore the potential of FreezeOut as a viable method for speeding up pre-training with minimal performance loss, paving the way for more efficient and accessible self-supervised learning models in computer vision.

REFERENCES

[1] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Freezeout: Accelerate training by progressively freezing layers," *arXiv preprint arXiv:1706.04983*, 2017.

[2] 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," *arXiv preprint arXiv:2010.11929*, 2020.

[3] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

[4] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Compute and energy consumption trends in deep learning inference," *arXiv preprint arXiv:2109.05472*, 2021.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21271–21284, 2020.

[8] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

[9] Y. Chen, A. Yuille, and Z. Zhou, "Which layer is learning faster? a systematic exploration of layer-wise convergence rate for deep neural networks," in *The Eleventh International Conference on Learning Representations*, 2023.

[10] D. Zhou, J. Chen, Y. Cao, Y. Tang, Z. Yang, and Q. Gu, "On the convergence of adaptive gradient methods for nonconvex optimization," *arXiv preprint arXiv:1808.05671*, 2018.

[11] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[16] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” *arXiv preprint arXiv:1608.06993*, 2016.

[17] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” *arXiv preprint arXiv:1905.11946*, 2019.

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.

[19] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, “Deformable convolutional networks,” *arXiv preprint arXiv:1703.06211*, 2017.

[20] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, “Training data-efficient image transformers distillation through attention,” *arXiv preprint arXiv:2012.12877*, 2020.

[21] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” *arXiv preprint arXiv:2005.12872*, 2020.

[22] H. Chen, Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu, and W. Gao, “Pre-trained image processing transformer,” *arXiv preprint arXiv:2012.00364*, 2020.

[23] 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,” *arXiv preprint arXiv:2103.14030*, 2021.

[24] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z. Jiang, F. E. Tay, J. Feng, and S. Yan, “Tokens-to-token vit: Training vision transformers from scratch on imagenet,” *arXiv preprint arXiv:2101.11986*, 2021.

[25] P. Zhang, X. Dai, J. Yang, B. Xiao, L. Yuan, L. Zhang, and J. Gao, “Multi-scale vision longformer: A new vision transformer for high-resolution image encoding,” *arXiv preprint arXiv:2103.15358*, 2021.

[26] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[27] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the 25th international conference on machine learning*, pp. 1096–1103, ACM, July 2008.

[28] B. A. Olshausen and D. J. Field, “Sparse coding with an overcomplete basis set: A strategy employed by v1?,” *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[30] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, October 2014.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[32] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.

[33] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.

[34] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by inpainting,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544, 2016.

[35] G. Larsson, M. Maire, and G. Shakhnarovich, “Colorization as a proxy task for visual understanding,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6874–6883, 2017.

[36] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[37] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

[38] L. Huang, C. Zhang, and H. Zhang, “Self-adaptive training: Bridging supervised and self-supervised learning,” *arXiv preprint arXiv:2101.08732*, 2021.

[39] X. Wang, R. Zhang, C. Shen, T. Kong, and L. Li, “Dense contrastive learning for self-supervised visual pre-training,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3024–3033, 2021.

[40] H. Bao, L. Dong, and F. Wei, “Beit: Bert pre-training of image transformers,” *arXiv preprint arXiv:2106.08254*, 2021.

[41] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu, “Simmim: A simple framework for masked image modeling,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9653–9663, 2022.

[42] C. Wei, H. Fan, S. Xie, C.-Y. Wu, A. Yuille, and C. Feichtenhofer, “Masked feature prediction for self-supervised visual pre-training,” *arXiv preprint arXiv:2112.09133*, 2021.

[43] X. Dong, J. Bao, T. Zhang, D. Chen, W. Zhang, L. Yuan, D. Chen, F. Wen, and N. Yu, “Peco: Perceptual codebook for bert pre-training of vision transformers,” *arXiv preprint arXiv:2111.12710*, 2021.

[44] L. Huang, S. You, M. Zheng, F. Wang, C. Qian, and T. Yamasaki, "Green hierarchical vision transformer for masked image modeling," *arXiv preprint arXiv:2205.13515*, 2022.

[45] P. Gao, T. Ma, H. Li, J. Dai, and Y. Qiao, "Convmae: Masked convolution meets masked autoencoders," *arXiv preprint arXiv:2205.03892*, 2022.

[46] J. Liu, X. Huang, J. Zheng, Y. Liu, and H. Li, "Mixmae: Mixed and masked autoencoder for efficient pretraining of hierarchical vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6252–6261, 2023.

[47] H. Wang, Y. Tang, Y. Wang, J. Guo, Z.-H. Deng, and K. Han, "Masked image modeling with local multi-scale reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2122–2131, 2023.

[48] J. Guo, K. Han, H. Wu, Y. Tang, Y. Wang, and C. Xu, "Fastmim: Expediting masked image modeling pre-training for vision," *arXiv preprint arXiv:2212.06593*, 2022.

[49] H. Xue *et al.*, "Stare at what you see: Masked image modeling without reconstruction," in *CVPR*, 2023.

[50] I. Kakogeorgiou, S. Gidaris, B. Psomas, Y. Avrithis, A. Bursuc, K. Karantzalos, and N. Komodakis, "What to hide from your students: Attention-guided masked image modeling," in *European Conference on Computer Vision*, pp. 300–318, Springer, 2022.

[51] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," *arXiv preprint arXiv:1502.02551*, 2015.

[52] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *International Conference on Machine Learning*, pp. 2849–2858, PMLR, 2016.

[53] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[54] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *arXiv preprint arXiv:1506.02626*, 2015.

[55] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[56] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[57] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, 2006.

[58] X. Xiao, T. B. Mudiyanselage, C. Ji, J. Hu, and Y. Pan, "Fast deep learning training through intelligently freezing layers," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1225–1232, IEEE, 2019.

[59] M. Zhang and Y. He, "Accelerating training of transformer-based language models with progressive layer dropping," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14011–14023, 2020.

[60] Y. Wang, D. Sun, K. Chen, F. Lai, and M. Chowdhury, "Egeria: Efficient dnn training with knowledge-guided layer freezing," *arXiv preprint arXiv:2201.06227*, 2022.

[61] Y. Liu, S. Agarwal, and S. Venkataraman, "Autofreeze: Automatically freezing model blocks to accelerate fine-tuning," *arXiv preprint arXiv:2102.01386*, 2021.

[62] L. M. R. Rere, M. I. Fanany, and A. M. Arymurthy, "Simulated annealing algorithm for deep learning," *Procedia Computer Science*, vol. 72, pp. 137–144, 2015.

[63] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," *arXiv preprint arXiv:1708.07120*, 2017.

[64] L. N. Smith, "Cyclical learning rates for training neural networks," *arXiv preprint arXiv:1506.01186*, 2015.

[65] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[67] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, "A closer look at memorization in deep networks," in *International Conference on Machine Learning*, pp. 233–242, PMLR, 2017.

[68] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, 2019.

[69] 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*, pp. 248–255, Ieee, 2009.

CHAPTER 7

INSTRUMENTS AND ETHICAL CLEARANCE

Use of AI Tools: In the preparation of this thesis, the AI language model ChatGPT, developed by OpenAI, was utilized to assist with drafting and refining certain sections of the document.

All information generated by the AI was critically reviewed and edited by the author to ensure accuracy and alignment with the thesis's research objectives. Additionally, the author remains fully responsible for the content and conclusions presented in this thesis.