Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Eigenvector-Masked Autoencoders

Bachelor's Thesis

Anja Buchmann

abuchmann@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Till Aczel, Andreas Plesner
Prof. Dr. Roger Wattenhofer

August 22, 2025

Acknowledgements

I sincerely want thank my supervisors, Till Aczel and Andreas Plesner, for their continued support. I am grateful for your guidance and perspective, I've profited greatly from our discussions.
I would also like to thank Prof. Dr. Roger Wattenhofer and the Distributed Computing Group for the opportunity to carry out this project and providing access to the group's computing cluster.

Abstract

Self-supervised visual representation learning is commonly performed by masking patches of input images and training a model to reconstruct them. Recently, a more successful approach has been proposed [1]: input images are mapped to principal component (PC) space, some positions are removed and the remaining coefficients are mapped back to image space for reconstruction of the missing coefficients.
In this work, we propose an architecture that performs masking, learning, and reconstruction entirely within PC space, avoiding repeated projections between PC and image space and leveraging the sequential nature of principal components more effectively.
Although this approach did not achieve competitive accuracy in a linear probe setting, it highlights key challenges in adapting transformer-based masked modeling to PC representations.

Contents

Chapter 1

Introduction

Self-supervised learning has revolutionized visual representation learning, with masked image modeling [2, 3, 4] emerging as a dominant paradigm. The Masked Autoencoder relies on random masking and reconstruction in pixel space to compel models to learn meaningful visual features. However, recent theoretical analysis challenges the assumption that pixel-space reconstruction inherently leads to perceptually useful representations. Specifically, [5] demonstrate a misalignment between reconstructive objectives and perceptual utility: models trained to minimize an image reconstruction error will learn reconstructing high-variance before low-variance components. However, focusing on lower-variance components can actually yield better performance, since they are more relevant for perceptual tasks.

Accordingly, the Principal Masked Autoencoder (PMAE [1]) adapts a Masked Autoencoder (MAE [4]) to mask in principal component (PC) space, encouraging the model to capture global image structure and high-level semantics. Empirically, PMAE has been shown to improve representation quality relative to traditional pixel-space masking.

However, the PMAE still trains in pixel space. This approach requires repeated projections between pixel and PC space throughout the training pipeline, which introduces computational overhead. We propose a novel architecture that operates entirely in PC space; the model performs masking, encoding, and reconstruction directly on PC coefficients with only a single initial projection.

Chapter 2

Background

2.1 Unsupervised Learning

In contrast to supervised learning, unsupervised learning algorithms leverage unlabeled data to learn meaningful structure or representations. For example, clustering methods group data into classes based on measures of similarity without requiring predefined labels.
In recent years, self-supervised learning, a subclass of unsupervised methods, has gained significant attention for its ability to generate supervisory signals from the data itself [6]. These approaches define pretext tasks to learn powerful data representations without manual annotations. In this work, we focus on representation learning through such self-supervised approaches.

2.1.1 Representation Learning

Representation learning [7] describes the process of transforming raw data into a more meaningful representation. More concretely, the goal is to learn an embedding function $f : x \to z$ that maps data points $x \in \mathbb{R}^D$ to representations $z \in \mathbb{R}^K$. These representations should capture the essential structure or semantics of the data in a compact form and can be used for downstream tasks such as classification.
Traditional approaches such as clustering and dimensionality reduction (e.g., principal component analysis [8]) offer early examples of this principle.
A widely used neural framework for learning such representations is the autoencoder [9], in which f serves as the encoder. A corresponding decoder function $g : z \to x$ reconstructs the input, and the model is trained to minimize a reconstruction loss $L(g(f(x)), x)$.

2.1.2 Principal Component Analysis

Principal component analysis (PCA [8]) is an unsupervised learning technique that performs a linear data transformation to identify the directions of maximum variance in a dataset. These directions, known as principal components, are orthogonal and uncorrelated linear combinations of the input variables.

When applied to N images, each image with c channels and spatial resolution $n \times n$ is first flattened to a vector $x_i' \in \mathbb{R}^D, D = c \cdot n^2$. The data matrix $X \in \mathbb{R}^{N \times D}$ is then constructed by stacking the $x_i', i = 1, ..., N$ as rows and centering. The sample covariance matrix is given by:

$$\Sigma = X^T X.$$

Principal components are obtained via eigenvalue decomposition on the covariance matrix:

$$\Sigma = V \Lambda V^T,$$

where $V = \begin{pmatrix} v_1 & \dots & v_D \end{pmatrix}$ contains the eigenvectors (principal components) as columns, and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_D)$ is a diagonal matrix of the corresponding eigenvalues, sorted in descending order: $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_D$.

Each eigenvalue λ_i is proportional to the variance explained by its corresponding principal component v_i. The principal components are thus ordered by decreasing explained variance. While PCA is commonly used for dimensionality reduction by selecting only the first $L < D$ principal components, we will use a lossless decomposition where all principal components are kept.

A data point x is projected into principal component space via:

$$x_{\text{PC}} = x \cdot V,$$

where x_{PC} contains the component-wise projection coefficients. Reconstruction back into the original space is then given by:

$$x = x_{\text{PC}} \cdot V^T,$$

where any required reshaping operations are omitted.

2.2 Transformers

Transformers [10, 11] are a neural network architecture based on self-attention mechanisms. They will be used in the models introduced and compared later in this work. Compared to earlier sequential models like RNNs and CNNs, Transformers enable greater parallelism during training and have achieved state-of-the-art performance across a wide range of tasks in NLP and computer vision.

2.2.1 The Transformer Architecture

The Transformer [10] is an encoder-decoder architecture introduced for sequence-to-sequence tasks in natural language processing such as machine translation. Unlike prior models based on recurrence or convolution, the Transformer relies solely on attention mechanisms, enabling greater parallelism and efficiency during training. Its ability to model long-range dependencies and learn contextual representations has made it the foundation of modern NLP systems and the basis for models like BERT and GPT.

Before being processed, input text is tokenized and mapped to continuous embeddings. Since the architecture lacks an inherent sense of word order, positional encodings are added to the embeddings to inject sequence information.

The model consists of stacked encoder and decoder layers. Each encoder layer applies multi-head self-attention followed by a position-wise feedforward network, while decoder layers additionally include cross-attention to attend to encoder outputs. The decoder generates the output sequence autoregressively, producing one token at a time based on the previously generated tokens.

In self-attention, each token attends to all others in the same sequence by computing a weighted sum of their representations, where the weights reflect pairwise relevance. This allows the model to capture contextual dependencies regardless of distance. Multiple attention heads operate in parallel, enabling the model to jointly focus on different types of relationships within the sequence. The Transformer's non-recurrent design enables parallel processing of entire sequences, leading to faster training and more stable optimization.

2.2.2 BERT

BERT (Bidirectional Encoder Representations from Transformers [12]) represents a major advance in natural language processing and is specifically designed for tasks that require understanding or classification. It uses only the encoder stack of the original Transformer architecture and consists of multiple Transformer encoder layers (e.g., 12 in BERT-Base, 24 in BERT-Large), where the output of the final encoder layer provides contextualized embeddings for each input token.

BERT is pretrained once and can then be finetuned to a various set of tasks with little effort. The pretraining consists of two objectives: masked language modeling and next sequence prediction. For masked language modeling, a small fraction (usually 15%) of tokens from the input sequence is randomly masked out, i.e. replaced with a special `[MASK]` token. The model then has to reconstruct the missing tokens. Since the model uses bidirectional self-attention, it can relate the masked token to previous as well as following tokens in the sequence, learning a contextual understanding across the whole sequence. In next-sequence prediction, BERT is given two sentences and has to predict wether the second sentence

logically follows the first. This task helps BERT understand coherence.

BERT's design and training objectives have made it a foundational model for a wide range of NLP tasks.

2.2.3 Vision Transformers

Following successes of Transformers in NLP, Vision Transformers (ViTs [11]) adapt this architecture to image processing tasks. The ViT consists of a Transformer encoder, composed of alternating layers of multi-headed self-attention and MLP blocks. Layer normalization is applied before each block, and residual connections after each block.

An input image is divided into fixed-size patches, each of which is flattened and linearly projected into a patch embedding. A learnable `[class]` token is prepended to the sequence of embeddings, whose output representation serves as image summary. The sequence is fed as input to the encoder alongside positional embeddings.

Conventionally, the Vision Transformer is trained in a supervised fashion on large labeled image datasets such as ImageNet-21k, where a classification head is attached to the output of the `[class]` token to predict class labels. After pretraining, the model can be finetuned on smaller datasets by replacing or adapting the classification head and continuing supervised training.

However, this approach to pretraining requires large-scale annotated data to achieve strong performance and avoid overfitting. To address this limitation, self-supervised objectives such as masked image modeling (e.g., MAE) have been introduced.

2.3 Masked Image Modeling

2.3.1 Masked Autoencoder

One attempt at solving the problems ViTs face is the Masked Autoencoder (MAE [4]). A self-supervisory task for pretraining a ViT was proposed: Inspired by BERT, the MAE learns to reconstruct masked out patches from an input image.

In pretraining, input images are divided into patches. A high percentage of patches, typically 75%, is masked out. The encoder consists of a ViT which is applied on the unmasked patches only. Since the mask tokens are added later, the encoder operates only on a small subset (i.e. 25%) of the full input, which allows training to scale nicely. The encoded unmasked patches alongside mask tokens with positional embeddings are then fed to a decoder. The decoder consists of

some Transformer blocks but can be flexibly designed, since only the encoder is used for downstream tasks.

This is an efficient and effective way to train the ViT: On top of improved accuracy compared to vanilla supervised training, this approach also made training about three times faster and more scalable.

However, the MAE's performance is sensitive to masking ratio [1]. More precisely, the optimal masking ratio varies between datasets, rendering the standard masking ratio of 75% suboptimal for each dataset individually. Additionally, the high masking ratio may cause too many semantically relevant patches to be masked out which could deteriorate training.

2.3.2 Principal Masked Autoencoder

Recently, a new approach for masked image modeling has been proposed: Instead of masking image patches, the Principal Masked Autoencoder (PMAE [1]) masks out principal components and pretrains by having the model reconstruct the missing data.

Concretely, pretraining is carried out the following way: As a preprocessing step, PCA is performed on the dataset X. Additionally, the explained variance per principal component is calculated.

$$PCA(X) = V\Lambda V^T,$$

Each input image x is projected into principal component space, resulting in a sequence of coefficients.

$$x_{PC} = x \cdot V$$

Then, a masking ratio r is chosen uniformly at random from $[0.1, 0.9]$. Corresponding to explained variance, the coefficients are partitioned randomly into the masked and the unmasked coefficients respectively by a binary mask m:

$$(1 - m) \odot x_{PC}, m \odot x_{PC}$$

Now, the unmasked coefficients are mapped back to image space and fed to the model, which consists of an encoder-decoder architecture analogous to MAE. The output representation is then mapped back to principal component space and the loss is calculated as a mean squared error between masked and reconstructed coefficients:

$$L = ||PCA(g(f(PCA^{-1}(m \odot x_{PC})))), (1 - m) \odot x_{PC}||_2^2 \tag{2.1}$$

In contrast to the patches which are masked out in MAE, each component masked out in PMAE captures unique global information. Thus, this loss should

allow for learning more meaningful representations. Additionally, masking according to explained variance defines the masking ratio clearly in terms of information masked out whereas the information content masked out per patch in MAE is unclear.
PMAE shows improvements on top-1 accuracy for linear- and MLP probing as well as finetuning. This implies masking in the principal component space indeed yields more meaningful representations than in pixel space.
However, PMAE has longer training times than MAE: training for 800 epochs on CIFAR-10 took 300 minutes with PMAE and only about 260 minutes for MAE depending on masking ratio. This is caused by two disadvantages compared to MAE: Firstly, the input to the MAE model is smaller because only the unmasked patches are passed whereas in PMAE, the input is always of size of the whole image. Secondly, mapping each sample to principal component space for masking, back to image space as input to the model and then finally back to principal component space for loss computation is computationally expensive.

Chapter 3

Experimental Setup

3.1 Motivation

Masked modeling in the principal component space has shown promise as a pre-training strategy for image representation learning. However, the approach used in PMAE requires converting samples between pixel and principal component space several times for creating the masked inputs and computing the reconstruction loss in PC space. This repeated projection introduces computational overhead. Additionally, it may be more effective to train in PC space directly.
To address this, we propose adapting PMAE to operate entirely in PC space. By projecting each image into the PC basis only once and performing both masking and reconstruction directly on the resulting coefficients, we remove the need for repeated pixel-to-PC mappings. This should lead to a more efficient training pipeline while retaining the benefits of PC-space masking.

3.2 Model Overview

The proposed model follows a masked-autoencoding paradigm similar to PMAE, but operates directly in the principal component space. The pipeline consists of four main stages:

1. **PCA projection:** Images are projected into the pre-computed PCA basis.
2. **Masking:** A subset of coefficients is randomly masked based on the masking ratio defined in terms of cumulative explained variance.
3. **Encoding:** The masked sequence of coefficients is passed through a BERT-style encoder, which learns to contextualize the unmasked coefficients and infer representations of the missing ones.
4. **Decoding and reconstruction:** The encoder output is fed to a lightweight decoder that reconstructs the masked coefficients, and training is guided by a mean squared error loss computed on these masked positions.

This design allows the model to perform masked coefficient reconstruction with a single projection to PC space. Furthermore, by processing PCA coefficients as sequences rather than images, the architecture is tailored to the sequential structure of the PC basis.

3.3 Input Preparation and Masking

Principal component analysis is performed in advance on the dataset, and two training configurations are evaluated: one where the model is trained on the raw PC coefficients X_{PC}, and another where the coefficients are normalized. We will refer to the model trained on normalized data as PCA-BERT$^{\text{norm}}$, the model trained on unnormalized data as PCA-BERT$^{\text{unnorm}}$, and use PCA-BERT to collectively denote both variants.

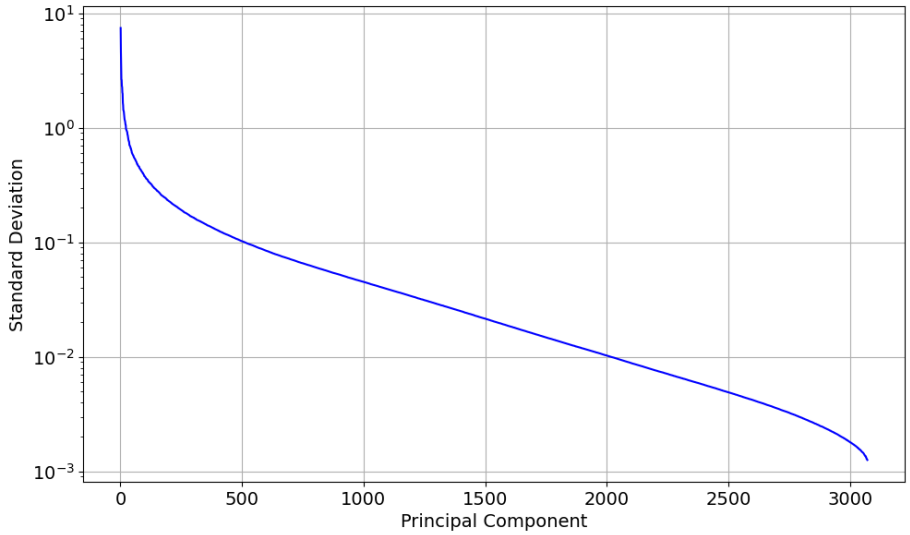

Figure 3.1: Standard deviation explained per principal component on CIFAR-10. 90% of the total standard deviation is explained by only the first 38% of components.

The normalization is motivated by the distribution shown in Figure 3.1: the standard deviation (and thus explained variance) drops sharply across components. If a coefficient which exhibits a small standard deviation is masked out, the model can achieve a near-minimal reconstruction loss by simply predicting the mean value of this coefficient. To mitigate this, the samples are normalized as

$$\hat{X}_{PC} = \frac{X_{PC} - \mu_{PC}}{\sigma_{PC}} \tag{3.1}$$

where μ_{PC} and σ_{PC} are the coefficient-wise mean and standard deviation respectively.
Masking is applied independently to each sample. First, a masking ratio r is drawn uniformly at random from the interval $[0.1, 0.9]$. Then, the coefficient sequence is randomly shuffled and coefficients are removed one by one until the cumulative explained variance of their associated principal components reaches r. Importantly, masking is always based on the original explained variance; the normalization of coefficients does not affect the process.

3.4 Model Architecture

3.4.1 Encoder

While PMAE operates on input images, our model instead processes sequences of coefficients. Directly adopting a ViT-based architecture as in PMAE presents challenges: unlike image pixels, coefficients are orthogonal and lack local spatial relationships. Consequently, patchifying them (as done by ViTs) would be inappropriate.
To address this, a BERT-style architecture was adopted. Note that our pretraining strategy is similar to the masked language modeling task of standard BERT since given an input sequence, a subset of positions is randomly masked and must be reconstructed. However, our input sequence consists of continuous values rather than words which are tokenized. Masked positions are always replaced with a learnable `[MASK]` token, and learnable positional encodings are added to all positions to retain the decreasing order of explained variance.
The PCA-BERT is a small version of a standard BERT-style model, however since the input coefficients are continuous values, they are linearly projected to the embedding dimension rather than tokenized as in natural language processing. Details on the hyperparameters can be found in Appendix A.2.

3.4.2 Decoder

To obtain the reconstructed coefficients, the resulting representations from the encoder are linearly projected to the coefficient space. Specifically, each position from the last hidden state output (of size 128 in our case) is projected to one coefficient (size 1) again. This lightweight decoder was chosen to keep computational intensity low.

3.5 Training Objective

Given a masked coefficient sequence $m \odot x_{PC}$, encoder $f(\cdot)$, and decoder $g(\cdot)$, the reconstruction loss Equation 2.1 can be simplified to:

$$L = ||g(f(m \odot x_{PC})), (1 - m) \odot x_{PC}||_2^2 \tag{3.2}$$

where m is a binary mask and the loss is computed as a mean squared error over the masked positions.

3.6 Evaluation

We evaluate representation quality using an online linear probe trained jointly with the encoder and decoder. Specifically, the encoder outputs are average-pooled across unmasked positions to form a fixed-length representation of the size of one output position in the sequence, which is then mapped to class logits by a single linear layer. The probe is updated alongside the main model, allowing validation accuracy to be tracked continuously throughout training. For evaluation, no positions are masked so that the probe operates on the full representation, making the task closer to a realistic downstream setting. This validation accuracy serves as a simple and interpretable proxy for downstream performance, reflecting how much task-relevant information is retained in the learned embeddings.

Chapter 4

Results

4.1 PCA-BERT

As shown in Figure 4.1, the training loss plateaus after the first epoch, indicating the model ceases to make meaningful progress beyond the first epoch. A similar trend is observed in the linear probe evaluation: as shown in Figure 4.2, PCA-BERT achieves approximately 10% final online validation accuracy when trained on unnormalized data, and around 12% when trained on normalized data.

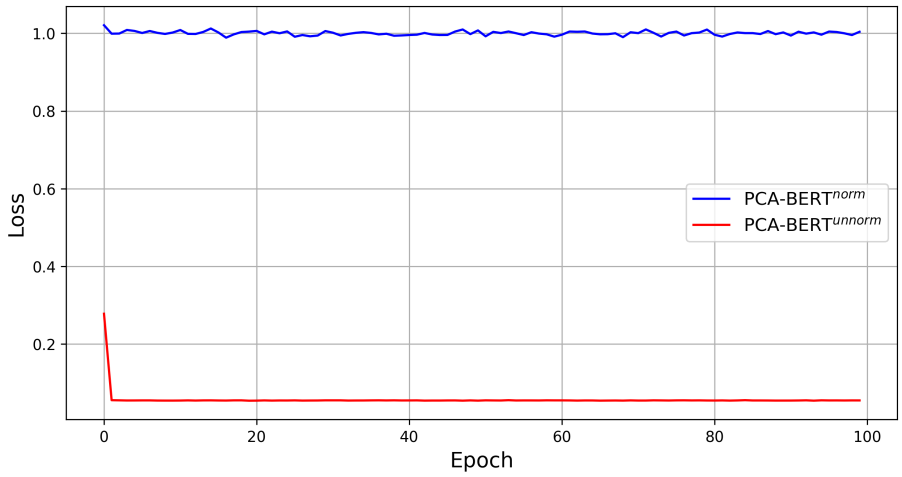

Figure 4.1: Training loss when training on normalized and unnormalized data. While training loss is higher with normalized data, neither shows any improvement beyond the first epoch.

Although normalization seems to provide a slight improvement in validation accuracy, it does not resolve the underlying issue: the training loss remains stagnant, and fluctuations in validation accuracy appear random without any upward trend for both variants. The lack of improvements of PCA-BERT$^{\text{norm}}$'s training loss could mean that the improved accuracy is just a random fluctuation as well.

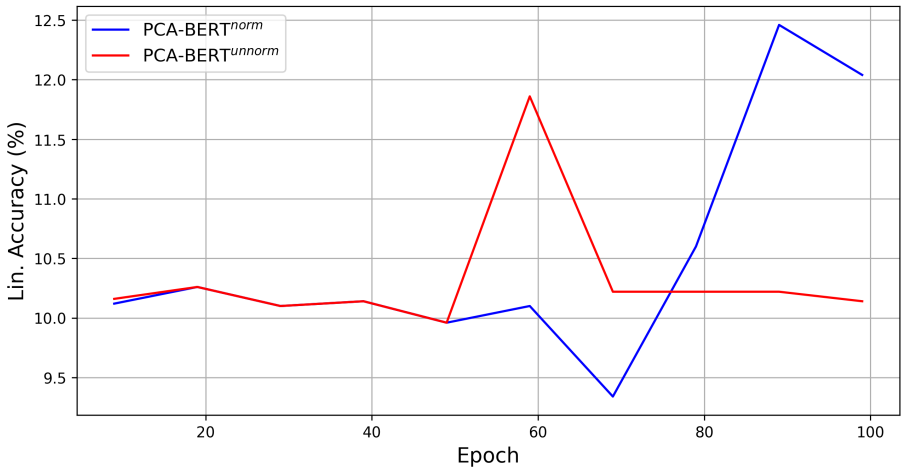

Figure 4.2: Online validation accuracy when training on normalized and unnormalized data. Fluctuations seem random, but PCA-BERT$^{\text{norm}}$ exhibits a higher peak and final accuracy.

In any case, PCA-BERT$^{\text{norm}}$ only marginally surpasses an accuracy of 10%, the accuracy a random classifier would achieve.

4.2 Comparison to PMAE

In contrast to PCA-BERT, PMAE exhibits a steady decrease in training loss throughout its training, as shown in Figure 4.3. Importantly, PMAE's loss already decreases substantially within the first 100 epochs, whereas PCA-BERT shows no comparable reduction over the same period. This suggests that simply extending PCA-BERT's training beyond 100 epochs would likely not resolve its stagnation. Furthermore, when comparing online validation accuracy in Table 4.1, PMAE clearly outperforms both PCA-BERT variants, even within the first 100 epochs.

Table 4.1: Online validation accuracy of tested models

Model	**Epoch**	**Lin. Accuracy (%)**
PCA-BERT$^{\text{unnorm}}$	100	10.14
PCA-BERT$^{\text{norm}}$	100	12.04
PMAE	100	22.23
PMAE	800	45.00

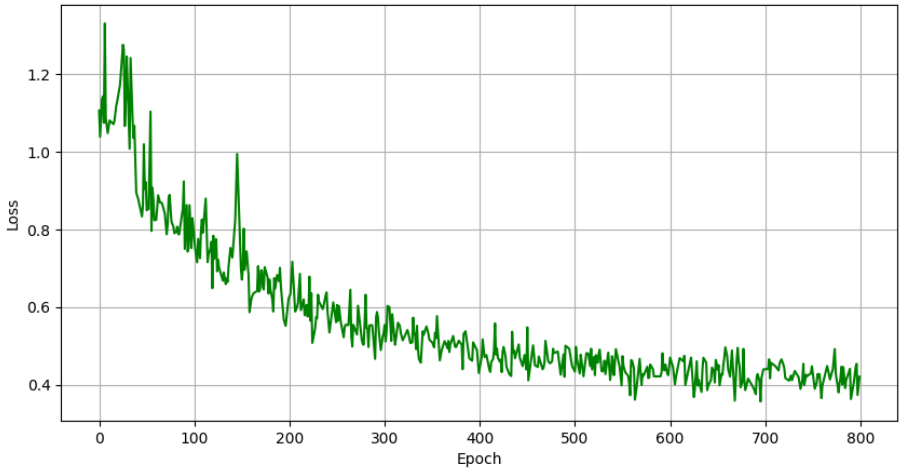

Figure 4.3: Training loss of PMAE. The model steadily improves on the pretraining objective throughout training.

4.3 Representation Quality

When training a linear classifier on top of the raw images projected to PC space, one obtains a final test accuracy of roughly 39%. This should serve as a baseline of representation quality: the goal of representation learning is finding a representation which is more meaningful than the original one. While PMAE is able to surpass this by 6%, PCA-BERT clearly falls short. PCA-BERT's inability to

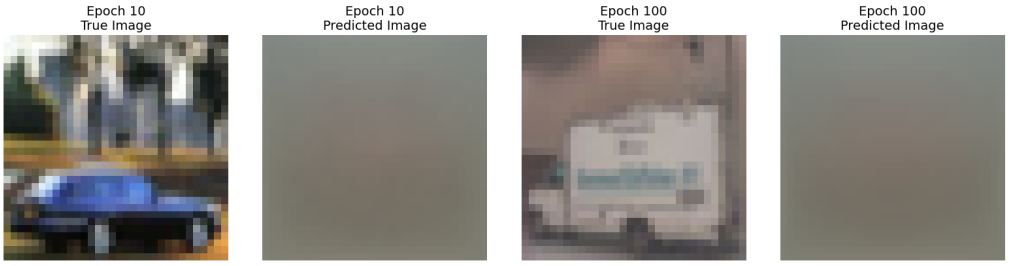

Figure 4.4: Examples of training inputs (before masking) and decoder outputs from PCA-BERT$^{\text{norm}}$, projected back to image space. Even though they are taken from different images and training epochs, the reconstructions are indistinguishable.

learn meaningful representations is further reflected by the reconstructed training images shown in Figure 4.4. The reconstructed images remain unrecognizable throughout training, and their quality does not seem to improve between epochs 10 and 100. Moreover, they are strikingly similar, with a mean squared error of only 0.000018 despite originating from different input samples. This indicates

that PCA-BERT fails to learn effectively reconstructing the missing coefficients, and that the model's outputs are effectively independent of its inputs.

4.4 Other Results

Several additional experiments were conducted during the course of this work. These results are not included here because they were obtained with an earlier version of the code that contained an implementation error in the masking strategy: masked positions were not replaced by a mask token as intended. Since this behavior differs from the final implementation, the results are not directly comparable. These experiments include:

- Stable masking ratios between 20% and 99%.
- Using first token pooling instead of average pooling.
- Weighting the loss by coefficient-wise explained variance and by its inverse.
- Modifying training hyperparameters, including learning rate adjustments and additional training epochs.
- Different hyperparameter configurations for the BERT variant

Under the faulty masking strategy, none of these changes yielded a substantial improvement in final accuracy.

Chapter 5

Conclusion and Future Work

In this work, we introduced a new approach to image representation learning in principal component (PC) space, in which PC coefficients are masked out and the model, comprising a BERT-style encoder and a lightweight decoder, is trained to reconstruct the missing coefficients. Our model operates entirely in PC space, eliminating the need for repeated projections between PC and image space.

Our experiments did not yield a competitive model. Stagnant training loss and lack of improvement in online accuracy throughout training suggest that the model fails to learn the reconstruction task. Visual inspection of decoder reconstructions further supports this conclusion: representation quality does not improve throughout training, and model outputs don't seem to correlate to model inputs. By contrast, PMAE demonstates steady improvements in both training loss and online validation accuracy, indicating that PCA-BERT's failure is not due to the reconstruction task itself but rather to the chosen adaptation of BERT. These findings highlight the need for more fundamental architectural modifications, rather than incremental adjustments, to make a transformer-based model effective in this setting. Possible contributing factors include the direct substitution of BERT's discrete tokenization with a linear projection of the continuous coefficients, which may be too small a modification to fully leverage the architecture. Furthermore, the interpretation of a `[MASK]` token in this continuous setting is unclear. It also remains uncertain whether a hidden implementation error contributed to the results.

Future work could address these limitations by redesigning the input processing to more closely emulate a tokenization step. The simple decoder could be replaced with a more complex one to keep the architecture more similar to the PMAE. Beyond this, architectures better suited to continuous representations than BERT should be explored. Given the relatively strong baseline performance of a linear probe on PC coefficients, simpler encoder architectures may be more effective, especially if they exploit the MAE advantage of shorter input sequences by omitting masked positions.

Bibliography

[1] A. Bizeul, T. Sutter, A. Ryser, B. Schölkopf, J. von Kügelgen, and J. E. Vogt, "From pixels to components: Eigenvector masking for visual representation learning," 2025. [Online]. Available: https://arxiv.org/abs/2502.06314

[2] H. Bao, L. Dong, S. Piao, and F. Wei, "Beit: Bert pre-training of image transformers," 2022. [Online]. Available: https://arxiv.org/abs/2106.08254

[3] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," 2016. [Online]. Available: https://arxiv.org/abs/1604.07379

[4] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," 2021. [Online]. Available: https://arxiv.org/abs/2111.06377

[5] R. Balestriero and Y. LeCun, "Learning by reconstruction produces uninformative features for perception," 2024. [Online]. Available: https://arxiv.org/abs/2402.11337

[6] J. Gui, T. Chen, J. Zhang, Q. Cao, Z. Sun, H. Luo, and D. Tao, "A survey on self-supervised learning: Algorithms, applications, and future trends," 2024. [Online]. Available: https://arxiv.org/abs/2301.05712

[7] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," 2014. [Online]. Available: https://arxiv.org/abs/1206.5538

[8] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[9] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, 2006.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words:

Transformers for image recognition at scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

Appendix A

Experimental Setup

A.1 Dataset

The CIFAR-10 dataset was used throughout this work. It contains 50'000 training and 10'000 validation RGB images of size 32×32 divided into ten classes. No data augmentations were used.

A.2 PCA-BERT

A BERT architecture with the configurations found in A.1 was used for PCA-BERT. The model was trained for 100 epochs using the hyperparameters listed in table A.2. For numerical stability, gradients are clipped at a threshold of 1.

Table A.1: Model hyperparameters.

config	**value**
hidden size	128
number of transformer layers	4
number of attention heads	4
intermediate size	512
normalization	Y/N

Table A.2: Training hyperparameters.

config	**value**
batch size	16
learning rate	0.0001
optimizer	AdamW
betas	(0.9, 0.999)
warmup steps	10% of total steps
weight decay	0.01

A.3 PMAE

The PMAE was trained as specified in [1], using the standard random masking approach.

A.4 Computational Resources

Training of both PCA-BERT variants was conducted on two NVIDIA GeForce RTX 2080 Ti GPUs with training times between 580 and 900 minutes.