MixMAE: Mixed and Masked Autoencoder for Efficient Pretraining of Hierarchical Vision Transformers

Jihao Liu\textsuperscript{1,2} Xin Huang\textsuperscript{2} Jinliang Zheng\textsuperscript{2} Yu Liu\textsuperscript{2} Hongsheng Li\textsuperscript{1,3}
\textsuperscript{1} CUHK MMLab
\textsuperscript{2} SenseTime Research
\textsuperscript{3} CPII under InnoHK

Abstract

In this paper, we propose Mixed and Masked AutoEncoder (MixMAE), a simple but efficient pretraining method that is applicable to various hierarchical Vision Transformers. Existing masked image modeling (MIM) methods for hierarchical Vision Transformers replace a random subset of input tokens with a special \texttt{[MASK]} symbol and aim at reconstructing original image tokens from the corrupted image. However, we find that using the \texttt{[MASK]} symbol greatly slows down the training and causes pretraining-finetuning inconsistency, due to the large masking ratio (e.g., 60% in SimMIM). On the other hand, MAE does not introduce \texttt{[MASK]} tokens at its encoder at all but is not applicable for hierarchical Vision Transformers. To solve the issue and accelerate the pretraining of hierarchical models, we replace the masked tokens of one image with visible tokens of another image, i.e., creating a mixed image. We then conduct dual reconstruction to reconstruct the two original images from the mixed input, which significantly improves efficiency. While MixMAE can be applied to various hierarchical Transformers, this paper explores using Swin Transformer with a large window size and scales up to huge model size (to reach 600M parameters). Empirical results demonstrate that MixMAE can learn high-quality visual representations efficiently. Notably, MixMAE with Swin-B/W14 achieves 85.1% top-1 accuracy on ImageNet-1K by pretraining for 600 epochs. Besides, its transfer performances on the other 6 datasets show that MixMAE has better FLOPs / performance tradeoff than previous popular MIM methods.

1. Introduction

Utilizing unlabeled visual data in self-supervised manners to learn representations is intriguing but challenging. Following BERT \cite{devlin2018bert} in natural language processing, pretraining with masked image modeling (MIM) shows great success in learning visual representations for various downstream vision tasks \cite{deng2009imagenet, he2016deep, xie2020start, zheng2021s, liu2021swin}, including image classification \cite{rasouli2020object}, object detection \cite{he2017mask}, semantic segmentation \cite{zhou2017scene}, video classification \cite{wang2019video}, and motor control \cite{vemulapalli2020hierarchical}. While those state-of-the-art methods \cite{deng2009imagenet, he2016deep} achieved superior performance on vanilla Vision Transformer (ViT) \cite{dosovitskiy2020image, touvron2021training}, it is still an open question how to effectively pretrain hierarchical ViT to purchase further efficiencies \cite{dosovitskiy2020image, touvron2021training, li2021scaling, tian2021efficient, wang2021deit} on broad vision tasks.

In general, existing MIM approaches replace a portion of input tokens with a special \texttt{[MASK]} symbol and aim at recovering the original image patches \cite{deng2009imagenet, he2016deep}. However, using \texttt{[MASK]} symbol leads to two problems. On the one hand, the \texttt{[MASK]} symbol used in pretraining never appears in the finetuning stage, resulting in pretraining-finetuning inconsistency \cite{devlin2018bert}. On the other hand, the pretrained networks waste much computation on processing the less informative \texttt{[MASK]} symbols, making the pretraining process inefficient. Those problems become severer when a large masking ratio is used \cite{deng2009imagenet, he2016deep, zheng2021s, liu2021swin}. For example, in SimMIM \cite{zheng2021s}, a masking ratio of 60% is used during the pretraining, i.e., 60% of the input tokens are replaced with the \texttt{[MASK]} symbols. As a result, SimMIM needs relatively more epochs (i.e., 800) for pretraining. In addition, as the high masking ratio causes much pretraining-finetuning inconsistency, the performances of SimMIM on downstream tasks are limited.

In contrast, MAE \cite{he2016deep} does not suffer from the above problems by discarding the masked tokens in the encoder and uses the \texttt{[MASK]} symbols only in the lightweight decoder. MAE utilizes the vanilla ViT \cite{dosovitskiy2020image} as the encoder, which can process the partial input efficiently with the self-attention operation. However, the design also limits the application of MAE on hierarchical ViTs as the hierarchical ViTs cannot process 1D token sequences with arbitrary lengths \cite{tian2021efficient, li2021scaling}.

In this work, we propose MixMAE, a generalized pretraining method that takes advantage of both SimMIM \cite{zheng2021s} and MAE \cite{he2016deep} while avoiding their limitations. In particular, given two random images from the training set, MixMAE creates a mixed image with random mixing masks as input.
and trains a hierarchical ViT to reconstruct the two original images to learn visual representations. From one image’s perspective, instead of replacing the masked tokens of the image with the special [MASK] symbols, the masked tokens are replaced by visible tokens of the other image. MixMAE adopts an encoder-decoder design. The encoder is a hierarchical ViT and processes the mixed image to obtain hidden representations of the two partially masked images. Before the decoding, the hidden representations are unmixed and filled with the [MASK] tokens. Following MAE [16], the decoder is a small ViT to reconstruct the two original images. We illustrate the proposed MixMAE in Figure 1.

MixMAE can be widely applied to pretrain different hierarchical ViTs, such as Swin Transformer [28], Twins [8], PVT [35], etc. Thanks to the utilization of the hierarchical architecture, we can naturally apply the pretrained encoder to object detection and semantic segmentation tasks. Empirically, with similar model sizes and FLOPs, MixMAE consistently outperforms BEiT [4] and MAE [16] on a wide spectrum of downstream tasks, including image classification on iNaturalist [18] and Places [41], object detection and instance segmentation on COCO [25], and semantic segmentation on ADE20K [42]. By abandoning using [MASK] tokens in the encoder, MixMAE shows much better pretraining efficiency than SimMIM [40] on various hierarchical ViTs [8][28][35].

## 2. Related Works

Inspired by BERT [12] for Masked Language Modeling, Masked Image Modeling (MIM) becomes a popular pretext task for visual representation learning [2][4][16]. MIM aims to reconstruct the masked tokens from a corrupted input. Current MIM approaches can be divided into two categories by the reconstruction targets. SimMIM [40] points out that raw pixel values of the randomly masked patches are a good reconstruction target and a lightweight prediction head is sufficient for pretraining. Different from SimMIM, MAE [16] only takes the visible patches as the input of the encoder. Mask tokens are added in the middle of the encoder and the decoder. Such an asymmetric design greatly reduces the computation overhead of the encoder. To further enhance the feature extraction capability of the encoder, CAE [6] separates the encoder and decoder explicitly by adding a feature alignment module in the middle of them. Jean-Baptiste et al. [1] propose to learn representations by reconstructing original videos from synthetically mixed ones.

Instead of building the reconstruction target manually, using a network to generate the reconstruction target has also been widely applied. In such works, an image tokenizer is used to transform an image into visual tokens. BEiT [4] utilizes a pretrained discrete VAE (dVAE) [30][31] as the tokenizer. However, the originally used MSE loss in dVAE is insufficient to force the tokenizer to capture high-level semantics. PeCo [13] proposed to apply perceptual similarity loss on the training of dVAE can drive the tokenizer to generate better semantic visual tokens, which helps pretraining. Moreover, the tokenizer in BEiT [4] needs to be offline pretrained, which limits the model’s adaption ability. To address the problem, iBOT [43] proposed to use an online tokenizer to generate the visual tokens.

There are also concurrent works that explore using MAE on hierarchical Vision Transformers. UM-MAE [22] proposed a new masking strategy for adapting MAE to pretrain pyramid-based ViTs (e.g., PVT [35], Swin [28]). GreenMIM [20] also adapt MAE on hierarchical architectures. It partitions the local windows into several equal-sized groups and proposes an optimal grouping algorithm to find the optimal group size. Instead of designing specific masking or grouping strategies, we focus on rearranging the inputs and targets. Empirical evaluation shows that MixMAE can obtain better performance with various hierarchical architectures.

## 3. Methodology

In this section, we introduce the proposed MixMAE for learning visual representations via Masked Image Modeling (MIM). We start by briefly revisiting MIM, and then introduce how MixMAE creates training inputs and performs image reconstruction, as well as the hierarchical Transformer architecture. Finally, we present how to reduce the difficulty of the pretext task to improve the pretraining efficiency.

### 3.1. A Revisit of Masked Image Modeling

Following BERT [12], recent works [4][16][40] proposed MIM for learning visual representations. Given an input image \( x \), MIM firstly divides the image into non-overlapping image patches \( x_p \), following ViT [14]. It then samples a random mask \( M \) to mask a portion of the image patches, and fills the masked place with a special symbol [MASK], i.e., \( \hat{x} = x_p \odot M + [\text{MASK}] \odot (1 - M) \), where \( \odot \) denotes element-wise multiplication. The masked image \( \hat{x} \) is processed by an image encoder to produce the latent representations, and a lightweight decoder (head) is utilized to reconstruct the original image based on the latent representations. The reconstruction target can be chosen as the normalized raw pixel \( [16][40] \) or visual tokens \( [4][13] \). MIM computes the mean squared error (MSE) between the reconstructed image patches \( \hat{y} \) and the original image patches \( x \) as the
reconstruction loss, \( \mathcal{L}_{\text{rec}} = \| (y^p - x^p) \odot (1 - M) \|^2_2 \), which is only calculated on masked patches \([16]\). After pretraining, the decoder is discarded and the encoder is used for further finetuning on downstream visual tasks.

### 3.2. Mixed and Masked Autoencoder (MixMAE)

While previous MIM works achieved great progress in self-supervised visual representation pretraining, they usually require a large number of epochs for pretraining. One reason is that they waste much computation on processing the less informative [MASK] symbols. Besides, using the [MASK] symbol also causes pretraining-finetuning inconsistency as those symbols never appear during finetuning.

To tackle the issues, we create mixed images as training inputs from pairs of unlabelled training images, which are generated by mixing two groups of visible tokens from two images, for pretraining. The mixed input is processed by MixMAE to reconstruct original images simultaneously. For better transferring the learned multi-scale representations to downstream tasks, we utilize the popular Swin Transformer with larger window size as the encoder of the proposed MixMAE \([27][28]\). Figure 1 illustrates the proposed framework.

#### Mixed Training Inputs

Given two sets of image patches \( \{x_1^p, x_2^p\} \) of two random training images, we create a mixed image by filling each spatial location with the corresponding visual token from either \( x_1^p \) or \( x_2^p \). The mask notation \( M \) is slightly abused and we denote \( M = 1 \) as choosing a token from \( x_1^p \) and vice versa. The mixed training image \( \hat{x}_m^p \) is therefore formulated as:

\[
\hat{x}_m^p = x_1^p \odot M + x_2^p \odot (1 - M).
\]

MixMAE then takes the mixed image as input for reconstruction during pretraining. The mixed image no longer consists of the extra [MASK] symbol and only actual visual tokens, leading to better performances on downstream tasks. The design shares the same principle of MAE \([16]\), but our approach does not disassemble the structure of the 2D image, making it more flexible for adapting to various visual backbones, such as PVT \([35]\) and Swin Transformer \([28]\). We can conduct better pretraining based on various hierarchical vision architectures. We follow common practices to use random masking strategy \([16][40]\).

#### Hierarchical Vision Transformer

For better encoding multi-scale representations, we build the encoder of MixMAE with popular Swin Transformer \([28]\) and use larger window size \([27]\) to encode more context for better reconstruction.

Following Swin Transformer, the input is split into non-overlapping image patches and processed by a linear projection layer. Then the image patches added with positional embeddings are processed by 4 stages of Transformer blocks to produce hierarchical representations, with a downsampling layer between every two successive stages. However, we do not use the complicated shifted window for information propagation across non-overlapping windows. Instead, we use a relatively large window size (i.e., \(14 \times 14\)), and only conduct global self-attention in stage-3 and -4. The larger window size brings negligible computation overhead, but can better integrate the global context. As we usually use a large masking ratio in MIM, the global context is important for better reconstruction.

We scale the encoder of MixMAE following \([27]\) with configuration parameters listed below:

- **Base**: \( C = (128, 256, 512, 1024) \), \( H = (4, 8, 16, 32) \), \( B = (2, 2, 18, 2) \),
- **Large**: \( C = (192, 384, 768, 1536) \), \( H = (6, 12, 24, 48) \), \( B = (2, 2, 18, 2) \),
- **Huge**: \( C = (352, 704, 1408, 2816) \), \( H = (11, 22, 44, 88) \), \( B = (2, 2, 18, 2) \),

where \( C \), \( H \), and \( B \) denote the channel numbers, numbers of the attention heads, and the numbers of blocks for each stage. The window size is set to \(14 \times 14 / 7 \times 7\) for stage-1, -2, and -3 / -4 during pretraining. A linear layer is added between the encoder and the decoder to convert the embedding dimension of the encoder’s output to 512.

\[ \text{The feature map resolution is } 14 \times 14 / 7 \times 7 \text{ for stage-3 / -4. A } 14 \times 14 / 7 \times 7 \text{ window attention is equivalent to global self-attention.} \]
Dual Reconstruction. After encoding the mixed input, we unmix the mask tokens to reconstruct the original two images from the two groups with the decoder, which has 8 Transformer blocks with an embedding dimension of 512. The loss is therefore set as

$$\mathcal{L}_{rec} = \|(y_1^p - x_1^p) \odot (1 - M)\|^2_2 + \|(y_2^p - x_2^p) \odot M\|^2_2,$$

(2)

where $y_1^p$ and $y_2^p$ are the reconstructed images corresponding to $x_1^p$ and $x_2^p$, respectively. The intuition behind is that as the mixed input contains tokens from two images, we can fully utilize them by reconstructing both images to pretrained the neural network. The computation overhead of reconstructing both images is negligible as the decoder is lightweight. Our approach demonstrates much higher efficiency than previous works, as to be introduced in Section 5.

3.3. Reducing the Difficulty of the Pretext Task

Although the dual reconstruction (Eq. (2)) enjoys several benefits, it is a much more challenging optimization problem due to the mixture of image tokens, which causes slow convergence in our preliminary experiments. To reduce the optimization difficulty, we facilitate the dual reconstruction by exploring the following approaches.

• Mix embedding: Besides the positional embeddings, we add two mix embeddings to the visual tokens to implicitly differentiate the two mixing groups. Each mix embedding is a vector and is shared for tokens from the same image. In practice, we use different mix embeddings for the 4 stages of the encoder and add the embedding at the beginning of each stage.

• Masked self-attention: Thanks to the flexibility of the self-attention mechanism, we can also differentiate two mixing images explicitly by masking the self-attention fields. Specifically, for each token, it is only allowed to aggregate information from the tokens belonging to the same image (group). We implement the masked self-attention by reusing the mixing mask $M$ described in Section 3.2. Note that we upsample the mask $M$ by nearest interpolation at different stages to match the resolution of the feature map.

Both approaches do not introduce much computation overhead or extra parameters. The empirical results show that both approaches can help obtain better results. However, the second approach also leads to a faster convergence speed, which is crucial for large-scale pretraining. We use the second approach by default and ablate the design in Section 6.

4. Experimental Setup

We validate our proposed MixMAE by conducting experiments with pretraining-then-finetuning strategy, following previous practices [4, 16]. In particular, we use ImageNet-1K [11] as the training set for self-supervised pretraining. We then finetune the encoder of MixMAE to downstream tasks, including image classification on ImageNet-1K [11], iNaturalist [18], and Places [41], object detection and instance segmentation on ADE20K [42].

Pretraining on ImageNet-1K. We conduct self-supervised pretraining on ImageNet-1K [11]. By default, we pretrain for 600 epochs with the input size of $224 \times 224$. The window size is set as $14 \times 14$ for the first 3 stages and $7 \times 7$ for stage-4. The patch size of the mask is set to $32 \times 32$ as our hierarchical encoder eventually downsamples the input to $\frac{1}{32}$ of the input resolution. Following MAE [16], a masking ratio of 75% is used by default, which is implemented by mixing 4 images. We follow all other pretraining hyperparameters of those in MAE [16] for a fair comparison.

Finetuning for image classification. We conduct supervised finetuning with the pretrained encoder of our MixMAE on image classification tasks, including ImageNet-1K [11], Places [41], and iNaturalist [18]. We follow previous practices [4, 16] and use a layer-wise learning-rate decay strategy [9] for finetuning. We sweep the decay rate in $\{0.7, 0.75, 0.8\}$ and report the best-performing results. We use drop path regularization [19], and set the drop rate to 0.15/0.2/0.3 for Swin-B/L/H, respectively. We finetune Swin-B/L/H for 100/50/50 epochs following MAE [16].

Finetuning on COCO. We perform supervised finetuning on COCO [25] for object detection and instance segmentation using the Mask RCNN framework [17] with our pretrained encoder as the backbone. We reuse the training setup in MAE [16] for a fair comparison. We change the window size to $16 \times 16$ for being divisible by the input $1024 \times 1024$ resolution. Besides, we change the window sizes of the 6th-, 12th-, and 18th-block in stage-3 to $32 \times 32$ for cross-window interactions following the previous practice [23].

Finetuning on ADE20K. We perform supervised finetuning on ADE20K [42] for semantic segmentation. We use the UperNet [38] framework with our pretrained encoder as its backbone. We also change the window size as mentioned above. We reuse the training setup in BEiT [4] for a fair comparison.

We include more details about pretraining and finetuning in the Appendix.

5. Main Results

In this section, we compare our MixMAE to prior arts on various visual benchmarks. We present the results on ImageNet-1K [11] in Section 5.1 and then show the results on the other 6 benchmarks in Section 5.2. Note that all the results of MixMAE are obtained by conducting supervised finetuning of the encoder with self-supervised pretraining, without extra intermediate finetuning [4].
Figure 2. Tradeoffs of FLOPs vs. (left) top-1 accuracy on ImageNet-1K, (middle) APbbox on COCO, (right) and mIoU on ADE20K. All results are from various self-supervised pretraining methods followed by supervised finetuning. All entries on COCO [25] use Mask RCNN [17] framework. All entries on ADE20K [42] use UperNet [38] framework. Note that this comparison confounds differences in architecture and pretraining strategy.

Table 2. Comparison with state-of-the-art MIM methods. All entries are results of base-level models and have comparable model sizes. We report the finetuning accuracy on ImageNet-1K. The FLOPs and Params. are calculated for the encoders. † denotes the number of epochs is based on JFT [32]. ‡ results are from [37]. ○ denotes our implementation with the official code. FT and LIN denote top-1 accuracy on ImageNet-1K after finetuning and linear probing respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>FLOPs (G)</th>
<th>Param. (M)</th>
<th>Supervision</th>
<th>Pretrain Epochs</th>
<th>FT</th>
<th>LIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT [14]</td>
<td>ViT-B</td>
<td>17.5</td>
<td>86</td>
<td>RGB</td>
<td>14  †</td>
<td>79.9</td>
<td>-</td>
</tr>
<tr>
<td>BEiT [4]</td>
<td>ViT-B</td>
<td>17.6</td>
<td>87</td>
<td>DALL-E</td>
<td>800</td>
<td>83.2</td>
<td>37.6</td>
</tr>
<tr>
<td>CAE [6]</td>
<td>ViT-B</td>
<td>17.5</td>
<td>86</td>
<td>DALL-E</td>
<td>800</td>
<td>83.6</td>
<td>68.6</td>
</tr>
<tr>
<td>MaskFeat [36]</td>
<td>ViT-B</td>
<td>17.5</td>
<td>86</td>
<td>HOG</td>
<td>300</td>
<td>83.6</td>
<td>-</td>
</tr>
<tr>
<td>data2vec [3]</td>
<td>ViT-B</td>
<td>17.5</td>
<td>86</td>
<td>Feature</td>
<td>800</td>
<td>84.2</td>
<td>-</td>
</tr>
<tr>
<td>iBOT [43]</td>
<td>ViT-B</td>
<td>17.5</td>
<td>86</td>
<td>Momentum</td>
<td>1600</td>
<td>84.0</td>
<td>79.5</td>
</tr>
<tr>
<td>PeCo [13]</td>
<td>ViT-B</td>
<td>17.5</td>
<td>86</td>
<td>MoCo v3</td>
<td>800</td>
<td>84.5</td>
<td>-</td>
</tr>
<tr>
<td>MAE [16]</td>
<td>ViT-B</td>
<td>17.5</td>
<td>86</td>
<td>RGB</td>
<td>1600</td>
<td>83.6</td>
<td>67.8</td>
</tr>
<tr>
<td>MAE* [16]</td>
<td>Swin-B/W14</td>
<td>16.3</td>
<td>88</td>
<td>RGB</td>
<td>600</td>
<td>84.4</td>
<td>61.0</td>
</tr>
<tr>
<td>EsViT† [21]</td>
<td>Swin-B/W14</td>
<td>16.3</td>
<td>87</td>
<td>Momentum</td>
<td>300</td>
<td>83.9</td>
<td><strong>81.3</strong></td>
</tr>
<tr>
<td>SimMIM [40]</td>
<td>ViT-B</td>
<td>17.5</td>
<td>86</td>
<td>RGB</td>
<td>800</td>
<td>83.8</td>
<td>56.7</td>
</tr>
<tr>
<td>SimMIM [40]</td>
<td>Swin-B</td>
<td>15.6</td>
<td>88</td>
<td>RGB</td>
<td>800</td>
<td>84.0</td>
<td>-</td>
</tr>
<tr>
<td>SimMIM* [40]</td>
<td>Swin-B/W14</td>
<td>16.3</td>
<td>88</td>
<td>RGB</td>
<td>300</td>
<td>84.1</td>
<td>20.2</td>
</tr>
<tr>
<td>GreenMIM [20]</td>
<td>Swin-B</td>
<td>15.6</td>
<td>88</td>
<td>RGB</td>
<td>800</td>
<td>83.8</td>
<td>-</td>
</tr>
<tr>
<td>GreenMIM* [20]</td>
<td>Swin-B/W14</td>
<td>16.3</td>
<td>88</td>
<td>RGB</td>
<td>800</td>
<td>84.1</td>
<td>-</td>
</tr>
<tr>
<td>MixMAE</td>
<td>Swin-B</td>
<td>15.6</td>
<td>88</td>
<td>RGB</td>
<td>600</td>
<td>84.6</td>
<td>61.2</td>
</tr>
<tr>
<td>MixMAE</td>
<td>Swin-B/W14</td>
<td>16.3</td>
<td>88</td>
<td>RGB</td>
<td>300</td>
<td>84.8</td>
<td>63.8</td>
</tr>
<tr>
<td>MixMAE</td>
<td>Swin-B/W14</td>
<td>16.3</td>
<td>88</td>
<td>RGB</td>
<td>600</td>
<td><strong>85.1</strong></td>
<td>71.0</td>
</tr>
</tbody>
</table>

Table 2. Comparison with state-of-the-art MIM methods. All entries are results of base-level models and have comparable model sizes. We report the finetuning accuracy on ImageNet-1K. The FLOPs and Params. are calculated for the encoders. † denotes the number of epochs is based on JFT [32]. ‡ results are from [37]. ○ denotes our implementation with the official code. FT and LIN denote top-1 accuracy on ImageNet-1K after finetuning and linear probing respectively.

5.1. Results on ImageNet-1K

Comparisons with other MIM approaches. Table 2 presents the comparison between MixMAE and state-of-the-art Masked Image Modeling (MIM) works. Our MixMAE can obtain higher accuracy while requiring fewer epochs for pretraining. In particular, we achieve 84.8% top-1 accuracy with 300 epochs of pretraining, 1.6% better than BEiT [4] with 62.5% fewer epochs for pretraining. Besides, our MixMAE also enjoys longer pretraining as previous methods [16]. Specifically, we obtain strong 85.1% top-1 accuracy with only 600 epochs of pretraining with Swin-B/W14.

While previous works design various reconstruction targets to speed up the pretraining process [4, 36], our MixMAE reconstructs simply normalized pixels [16] and demonstrates strong pretraining efficiency. Compared to MaskFeat [36], our MixMAE obtains +1.2% better accuracy with the same pretraining epochs. PeCo [13] proposed to recon-
<table>
<thead>
<tr>
<th>Pretrain Method</th>
<th>Backbone</th>
<th>Pretrain Epochs</th>
<th>Finetune Epochs</th>
<th>Top-1 Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised [28]</td>
<td>Swin-B</td>
<td>-</td>
<td>300</td>
<td>83.5</td>
</tr>
<tr>
<td>SimMIM [40]</td>
<td>Swin-B</td>
<td>800</td>
<td>100</td>
<td>84.0</td>
</tr>
<tr>
<td>GreenMIM [20]</td>
<td>Swin-B</td>
<td>800</td>
<td>100</td>
<td>83.8</td>
</tr>
<tr>
<td>MixMAE</td>
<td>Swin-B</td>
<td>600</td>
<td>100</td>
<td><strong>84.6</strong></td>
</tr>
<tr>
<td>Supervised [40]</td>
<td>Swin-L</td>
<td>-</td>
<td>300</td>
<td>83.5</td>
</tr>
<tr>
<td>SimMIM [40]</td>
<td>Swin-L</td>
<td>800</td>
<td>100</td>
<td>85.4</td>
</tr>
<tr>
<td>GreenMIM [20]</td>
<td>Swin-L</td>
<td>800</td>
<td>100</td>
<td>85.1</td>
</tr>
<tr>
<td>MixMAE</td>
<td>Swin-L</td>
<td>600</td>
<td>50</td>
<td><strong>85.9</strong></td>
</tr>
<tr>
<td>Supervised [35]</td>
<td>PVT-L</td>
<td>-</td>
<td>300</td>
<td>81.7</td>
</tr>
<tr>
<td>SimMIM [16]</td>
<td>PVT-L</td>
<td>800</td>
<td>100</td>
<td>82.0</td>
</tr>
<tr>
<td>MixMAE</td>
<td>PVT-L</td>
<td>600</td>
<td>100</td>
<td><strong>83.4</strong></td>
</tr>
<tr>
<td>Supervised [8]</td>
<td>Twins-SVT-L</td>
<td>-</td>
<td>300</td>
<td>83.7</td>
</tr>
<tr>
<td>SimMIM [16]</td>
<td>Twins-SVT-L</td>
<td>800</td>
<td>100</td>
<td>83.3</td>
</tr>
<tr>
<td>GreenMIM [20]</td>
<td>Twins-SVT-L</td>
<td>800</td>
<td>100</td>
<td><strong>83.9</strong></td>
</tr>
<tr>
<td>MixMAE</td>
<td>Twins-SVT-L</td>
<td>600</td>
<td>100</td>
<td><strong>83.9</strong></td>
</tr>
</tbody>
</table>

Table 3. Comparison with state-of-the-art MIM methods using the same encoder. We report the finetuning accuracy on ImageNet-1K. † denotes our implementation with the official code using input size of 224 × 224.

Figure 3. Efficiency comparison between MixMAE and SimMIM. We report the finetuning accuracy on ImageNet-1K. The encoder is Swin-B/W14 with input size of 224 × 224.

We test with scaling MixMAE up to 600M parameters, as shown in Figure 2 (Left). MixMAE has better FLOPs vs. accuracy tradeoff than other approaches. In particular, MixMAE pretrained Swin-L and -H achieve 85.9% and 86.9% top-1 accuracy, respectively, being comparable with MAE-L (85.9%) and -H (86.9%) but requiring fewer FLOPs for inference (-40% for -L and -30% for -H).

Integrating MixMAE to other backbones. While previous works [16, 36] may be restricted to a specific architecture, our proposed MixMAE can generalize to various visual backbones, including Swin Transformer [28], Twins [8], and PVT [35]. We conduct a thorough comparison with other MIM approaches with fixed encoders. As shown in Table 3, MixMAE consumes the same or fewer epochs for pretraining but obtains consistently better performance on hierarchical ViTs. In particular, our MixMAE achieves 84.6% top-1 accuracy with Swin-B, +0.6% better than SimMIM [40] while requiring 200 fewer epochs for pretraining. With Swin-L, our MixMAE obtains 85.8% top-1 accuracy with 600 epochs of pretraining and 50 epochs of finetuning, showing higher efficiency than SimMIM. Besides, our MixMAE achieves 83.2% top-1 accuracy with PVT-L [35], improving the supervised baseline by a non-trivial margin.

5.2. Results of Transferring to Downstream Tasks

To further demonstrate the effectiveness of the visual representations learned by MixMAE, we transfer MixMAE to various visual benchmarks with settings described in Section 4.

Object detection and instance segmentation. We show the results on COCO [25] in Table 4. By pretraining for 600 epochs on ImageNet-1K, we achieve 52.7 APbox and 47.0 APmask with Swin-B, surpassing previous self-supervised approaches with fewer FLOPs and parameters. Compared to BEiT [4], our MixMAE obtains higher APbox (+2.9) and APmask (+2.6) with less pretraining epochs (-200). Note that our hierarchical backbone can naturally be transferred to object detection without re-designing [23] network architectures such as FPN [24].

Our MixMAE can also scale up to larger models in object detection task and obtains better performance. As shown in Table 4, we achieve 54.3 APbox (48.2 APmask) with Swin-L, +1.0 (+1.1) better than BEiT while requiring 200 fewer epochs for pretraining and 41% fewer FLOPs for inference. We further evaluate the tradeoff of FLOPs vs. APbox in Figure 2 (Middle). We also found that MixMAE outperforms other approaches by large margins.

Semantic segmentation. Table 4 also presents the results of MixMAE on ADE20K [42]. We compare its Mean Intersection over Union (mIoU) on ADE20K with other self-supervised approaches. Our pretrained Swin-B achieves 51.1 mIoU, +4.0 better than BEiT while requiring only half of FLOPs for inference. Besides, we obtain 53.8 mIoU by struct the perceptual codebook from a pretrained MoCo v3 network [7] and can achieve better performance to some extent. In comparison, our MixMAE obtains even better performance (+0.6%) than PeCo with fewer pretraining epochs (-200). The superior performance of MixMAE comes from our mixed pretraining as well as the hierarchical Vision Transformer. However, SimMIM [40] also utilizes a hierarchical Swin Transformer [28], but its performance is worse (-1.1%) than MixMAE with more pretraining epochs (+200). We conduct a thorough comparison with SimMIM by using the same encoder Swin-B/W14 and show the results in Figure 3. Our proposed MixMAE shows much better pretraining efficiency than SimMIM. We list the training time in the Appendix.
Table 4. Comparison with other self-supervised approaches on COCO and ADE20K. We report AP\textsuperscript{box} and AP\textsuperscript{mask} on COCO, and mIoU on ADE20K. The results of BEiT and MoCo v3 are from MAE \cite{16}. The results of EsViT are from \cite{40}. † denotes using supervised finetuning on ImageNet. ⋄ denotes our implementation with the official code.

Table 5. Comparison with other self-supervised approaches on classification tasks. We report the top-1 accuracy and average accuracy of all datasets. We also report the average accuracy over the 4 datasets.

scaling up the model to Swin-L. Thanks to the hierarchical design, our MixMAE consumes much fewer FLOPs for inference compared to other approaches with plain ViT. In Figure 2 (Right), our MixMAE outperforms other approaches by large margins.

**Image classification.** We further transfer MixMAE to other 4 classification datasets and show the results in Table 5. These datasets are challenging as the accuracies are relatively low, e.g., 57.9% top-1 accuracy on Places365 \cite{41} for MAE-B \cite{16}. However, our MixMAE can still outperform previous self-supervised approaches. In particular, we achieve an average +2.9% performance gain compared to MAE-B with Swin-B. Besides, our pretrained Swin-L has an average +1.4% performance gain over MAE-L while requiring only 58% FLOPs for inference.

### 6. Ablation Studies

In this section, we ablate the key designs of MixMAE and report the transferring results of each ablation. Unless otherwise specified, we pretrain Swin-B/W14 for 300 epochs with a masking ratio of 50%. By default, we report the top-1 accuracy on ImageNet-1K \cite{11} and mIoU on ADE20K \cite{42}. We also show the results on COCO \cite{25} in Appendix.
We visualize the four approaches in Figure 4. We compare the performances of the four approaches in Table 6. Our default choice Mix performs best in terms of accuracy on ImageNet-1K and mIoU on ADE20K. We find that the approach has a similar performance on ImageNet-1K but better performance on ADE20K compared to Zero. We hypothesize that the finetuning inconsistency has a larger impact on tasks without a lot of labeled images. The shuffle and zoomin approaches perform much worse than other approaches. Those two strategies cause easier pretext task and have lower preprocessing loss. However, the learned representation quality is lower.

### Masking ratio

Our MixMAE implements different masking ratios by mixing more images at inputs. In addition, we also experiment to add [MASK] tokens for a higher masking ratio. We ablate the masking ratios in Table 8. We find that using a masking ratio 75% by mixing 4 images performs best. In contrast, adding [MASK] tokens for a 75% masking ratio has worse performance, demonstrating the effectiveness of the proposed mixing approach.

### Pretraining epochs

Thanks to the dual reconstruction, our MixMAE can achieve strong performance with few pretraining epochs. We ablulate the pretraining epochs in Table 7. We find that the mIoU on ADE20K can be further improved with more pretraining epochs. We achieve 51.0 mIoU with 900 epochs of pretraining. In contrast, the accuracy on ImageNet-1K does not improve after 600 epochs. It might be because the finetuning on ImageNet-1K is more adequate.

#### Reducing the difficulty

As stated in Section 3.3, directly performing reconstruction with the mixed input is a much more challenging optimization problem. Hence, we provide two practical approaches to reduce the difficulty. We ablate the design in Table 10. We note that all the approaches do not bring nonnegligible FLOPs or parameters. We find that the performance of the approach without unmixing is worst even when trained for more epochs. In contrast, using mix embedding alleviates the problem and improves its performance with longer pretraining. However, using masked self-attention in our final solution is much more efficient, and has better performance.

### 7. Discussion and Conclusion

This paper proposes Mixed and Masked AutoEncoder (MixMAE) for efficient visual representation learning. Our MixMAE uses a mixed input created by mixing two (or more) images with random masks, and applies dual reconstruction to recover the original two (or more) images from the mixed hidden representations. We further explore using Swin Transformer with a larger window size for efficient representation learning. Empirical results on 7 visual benchmarks demonstrate MixMAE can learn high-quality visual representations efficiently and has better FLOPs / performance tradeoff than previous MIM works. While this paper focuses on the vision field, we hope our work will inspire future works in other modalities, such as text and audio.

#### Acknowledgement

This project is funded in part by National Key R&D Program of China Project 2022ZD0161100, by the Centre for Perceptual and Interactive Intelligence (CPII) Ltd under the Innovation and Technology Commission (ITC)’s InnoHK, by General Research Fund of Hong Kong RGC Project 14204021. Hongsheng Li is a PI of CPII under the InnoHK.
References


[16] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In CVPR, 2021.


