MIXMIM: MIXED AND MASKED IMAGE MODELING FOR EFFICIENT VISUAL REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In this study, we propose Mixed and Masked Image Modeling (MixMIM), a simple but efficient MIM method that is applicable to various hierarchical Vision Transformers. Existing MIM methods replace a random subset of input tokens with a special [MASK] symbol and aim at reconstructing original image tokens from the corrupted image. However, we find that using the [MASK] symbol greatly slows down the training and causes training-finetuning inconsistency, due to the large masking ratio (e.g., 60% in SimMIM). In contrast, 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 original two images from the mixed input, which significantly improves efficiency. While MixMIM can be applied to various architectures, this paper explores a simpler but stronger hierarchical Transformer, and scales with MixMIM-B, -L, and -H. Empirical results demonstrate that MixMIM can learn high-quality visual representations efficiently. Notably, MixMIM-B with 88M parameters achieves 85.1% top-1 accuracy on ImageNet-1K by pretraining for 600 epochs. Besides, its transferring performances on the other 6 datasets show MixMIM has better FLOPs / performance tradeoff than previous MIM methods.

1 INTRODUCTION

Utilizing unlabeled visual data in self-supervised manners to learn representations is intriguing but challenging. Following BERT (Devlin et al., 2019) in natural language processing, pre-training with masked image modeling (MIM) shows great success on pretraining visual representations for various downstream vision tasks (He et al., 2021; Bao et al., 2021; Xie et al., 2021; Xiao et al., 2022; Tong et al., 2022), including image classification (Deng et al., 2009), object detection (Lin et al., 2014), semantic segmentation (Zhou et al., 2017), video classification (Goyal et al., 2017b), and motor control (Xiao et al., 2022).

Existing MIM approaches generally replace a portion of input tokens with a special symbol [MASK] and aim at recovering the original image patches (Bao et al., 2021; Xie et al., 2021). However, the use of [MASK] symbol leads to two problems. On the one hand, the [MASK] symbol used in pretraining never appears in the finetuning stage, resulting in pretraining-finetuning inconsistency Devlin et al. (2019). On the other hand, the pretrained networks waste much computation on processing the less informative [MASK] symbols, making the pretraining inefficient. Those problems become severer when a large masking ratio is used (He et al., 2021). For example, in BEiT (Bao et al., 2021), a masking ratio of 40% is used in pretraining, i.e., 40% of the input tokens are replaced by the [MASK] symbols. As a result, BEiT needs relatively more epochs (i.e., 800) for pretraining. Besides, as the high masking ratio causes much pretraining-finetuning inconsistency, the performances of BEiT on downstream tasks are limited.

MAE (He et al., 2021) does not suffer from the above problems by discarding the masked tokens and uses the [MASK] symbols only in the lightweight decoder. However, MAE also breaks the 2D structure of the input image and is therefore not applicable to high-performing visual backbones, such as ConvNets (Tan & Le, 2019; Liu et al., 2022) and hierarchical ViT architectures (Liu et al., 2021; Wang et al., 2021). How to efficiently pretrain hierarchical ViT architectures, e.g., Swin Transformer (Liu et al., 2021) and Pyramid Vision Transformer (Wang et al., 2021), with MIM is still an open question.

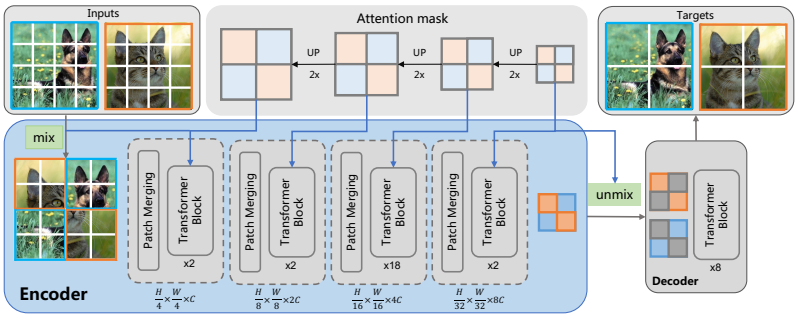

Figure 1: Overview of MixMIM. For pretraining, two images are mixed with a random mixing mask to create a mixed image. MixMIM takes the mixed image as input and reconstructs the two original images. The mixing mask is randomly sampled at the last stage of the encoder and upsampled to match the resolutions of previous stages. Right before decoding, the token embeddings are unmixed and filled with mask tokens for dual reconstruction of the two original images.

Table 1: Key differences between MixMIM and related works.

Approach	Pretraining efficient	Pretraining-finetuning consistent	Applicable to hierarchical ViT
BEiT (Bao et al., 2021)	✗	✗	✓
SimMIM (Xie et al., 2021)	✗	✗	✓
MAE (He et al., 2021)	✓	✓	✗
MixMIM	✓	✓	✓

In this work, we propose MixMIM, a generalized MIM method that takes advantage of both BEiT and MAE while avoiding their limitations. Given two random images from the training set, MixMIM 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. MixMIM adopts an encoder-decoder design. The encoder processes the mixed image to obtain hidden representations of the two partially masked images, while the decoder reconstructs the two original images.

The self-attention operation has a global receptive field and each token can easily interact with others. Fully fusing the two partial images' tokens would also cause pretraining-finetuning discrepancy, as the two groups of tokens might be from images of significantly different appearances at pretraining. To avoid introducing the new discrepancy, MixMIM utilizes a masked attention mechanism to explicitly prevent the interactions of the two groups of tokens. Note that we upsample the mixing mask by nearest interpolation at different stages of the hierarchical encoder to match the resolution of the attention map.

MixMIM can be widely applied to pretrain different hierarchical ViTs, such as Swin Transformer (Liu et al., 2021), PVT (Wang et al., 2021), etc. Our MixMIM also explores a lightweight architecture by modifying the Swin Transformer as the encoder for pretraining and knowledge transfer. Thanks to the hierarchical architecture, MixMIM can naturally be applicable to object detection and semantic segmentation. Empirically, with similar model sizes and FLOPs, MixMIM consistently outperforms BEiT (Bao et al., 2021) and MAE (He et al., 2021) on a wide spectrum of downstream tasks, including image classification on iNaturalist (Horn et al., 2018) and Places (Zhou et al., 2014), object detection and instance segmentation on COCO (Lin et al., 2014), and semantic segmentation on ADE20K (Zhou et al., 2017).

2 MixMIM for Masked Image Modeling

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

2.1 A REVISIT OF MASKED IMAGE MODELING

Following BERT (Devlin et al., 2019), recent works (Bao et al., 2021; He et al., 2021; Xie et al., 2021) 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 (Dosovitskiy et al., 2021). It then samples a random mask M to mask a portion of the image patches, and fills the masked place with a special symbol $[\mathrm{MASK}]$, $\hat{x}^p = x^p \odot M + [\mathrm{MASK}] \odot (1 - M)$, where $\odot$ denotes element-wise multiplication. The masked image $\hat{x}^p$ 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 (He et al., 2021) or visual tokens (Bao et al., 2021). MIM computes the mean squared error (MSE) between the reconstructed image patches y^p and the original image patches x^p as the reconstruction loss, $\mathcal{L}_{rec} = \|(y^p - x^p) \odot (1 - \mathrm{M})\|_2^2$, which is only calculated on masked patches (He et al., 2021). After pretraining, the decoder is discarded and the encoder is used for further finetuning on downstream visual tasks.

2.2 MIXED AND MASKED IMAGE MODELING (MIXMIM)

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 $[\mathrm{MASK}]$ symbols. Besides, using the $[\mathrm{MASK}]$ 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 MixMIM to reconstruct original images simultaneously. For better transferring the learned multi-scale representations to downstream tasks, we introduce a simple hierarchical vision Transformer as the encoder of the proposed MixMIM. 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:

$$\hat{x}_m^p = x_1^p \odot \mathrm{M} + x_2^p \odot (1 - \mathrm{M}). \tag{1}$$

MixMIM then takes the mixed image as input for reconstruction during pretraining. The mixed image no longer consists of the extra $[\mathrm{MASK}]$ symbol and only actual visual tokens, leading to better performances on downstream tasks. The design shares the same principle of MAE (He et al., 2021), 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 (Wang et al., 2021) and Swin Transformer (Liu et al., 2021). We can conduct better pretraining based on various hierarchical vision architectures. We follow common practices to use random masking (He et al., 2021; Xie et al., 2021).

Hierarchical Vision Transformer. For better encoding multi-scale representations, we build the encoder of MixMIM based on Swin Transformer and introduce minor modifications to make the MixMIM encoder simpler but stronger.

Similar to a regular 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., 14×14), and only conducts global self-attention in stage-3 and -4 [1]. 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 find our simpler architecture has better performance on downstream tasks compared to Swin Transformer.

We scale the encoder of MixMIM with configuration parameters listed bellow:

[1] The feature map resolution is 14×14 / 7×7 for stage-3 / -4. A 14×14 / 7×7 window attention is equivalent to global self-attention.

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

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 14×14 / 7×7 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.

Dual Reconstruction. After encoding the mixed input, we *unmix* the token embeddings into two groups according to the binary mask M. We then add the $[\mathrm{MASK}]$ 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

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

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 pretrain 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 4.

2.3 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 mixing 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 2.2. 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 5.

3 Experimental Setup

We validate our proposed MixMIM by conducting experiments with pretraining-then-finetuning strategy, following previous practices (He et al., 2021; Bao et al., 2021). In particular, we use ImageNet-1K (Deng et al., 2009) as the training set for self-supervised pretraining. We then finetune the encoder of MixMIM to downstreams tasks, including image classification on ImageNet-1K (Deng et al., 2009), iNaturalist (Horn et al., 2018), and Places (Zhou et al., 2014), object detection and instance segmentation on COCO (Lin et al., 2014), and semantic segmentation on ADE20K (Zhou et al., 2017).

Pretraining on ImageNet-1K. We conduct self-supervised pretraining on ImageNet-1K (Deng et al., 2009). By default, we pretrain for 600 epochs with the input size of 224×224. The window size is set as 14×14 for the first 3 stages and 7×7 for stage-4. Note that when using Swin Transformer (Liu et al., 2021) as the encoder, we set the window size following its original design. The patch size of

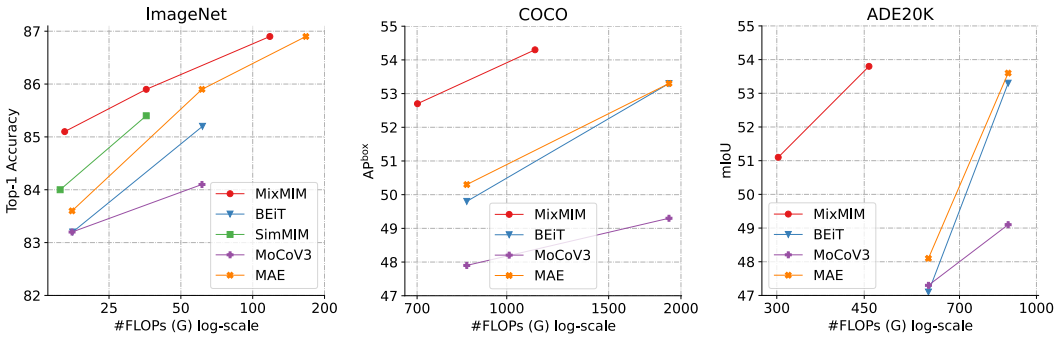

Figure 2: Tradeoffs of FLOPs vs. (left) top-1 accuracy on ImageNet-1K, (middle) AP^{box} on COCO, (right) and mIoU on ADE20K. All results are from various self-supervised pretraining methods followed by supervised finetuning. All entries on COCO (Lin et al., 2014) use Mask RCNN (He et al., 2017) framework. All entries on ADE20K (Zhou et al., 2017) use UperNet (Xiao et al., 2018) framework. Note that this comparison confounds differences in architecture and pretraining strategy.

the mask is set to 32×32 as our hierarchical encoder eventually downsamples the input to $\frac{1}{32}$ of the input resolution. Following MAE, 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 (He et al., 2021) for a fair comparison.

Finetuning for image classification. We conduct supervised finetuning with the pretrained encoder of our MixMIM on image classification tasks, including ImageNet-1K (Deng et al., 2009), Places (Zhou et al., 2014), and iNaturalist (Horn et al., 2018). We follow previous practices (Bao et al., 2021; He et al., 2021) and use a layer-wise learning-rate decay strategy (Clark et al., 2020) 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 (Huang et al., 2016), and set the drop rate to 0.15/0.2/0.3 for MixMIM-B/L/H, respectively. We finetune MixMIM-B/L/H for 100/50/50 epochs following MAE (He et al., 2021).

Finetuning on COCO. We perform supervised finetuning on COCO (Lin et al., 2014) for object detection and instance segmentation using the Mask RCNN framework (He et al., 2017) with our pretrained encoder as the backbone. We reuse the training setup in MAE (He et al., 2021) for a fair comparison. We change the window size to 16×16 for being divisible by the input 1024×1024 resolution. Besides, we change the window sizes of the 6th-, 12th-, and 18th-block in stage-3 to 32×32 for cross-window interactions following the previous practice (Li et al., 2021).

Finetuning on ADE20K. We perform supervised finetuning on ADE20K (Zhou et al., 2017) for semantic segmentation. We use the UperNet (Xiao et al., 2018) framework with our pretrained encoder as its backbone. We also change the window size as mentioned above. We reuse the training setup in BEiT (Bao et al., 2021) for a fair comparison.

We include more details about pretraining and finetuning in the Appendix A.1.

4 Main Results

In this section, we compare our MixMIM to prior arts on various visual benchmarks. We present the results on ImageNet-1K in section 4.1, and then show the results on the other 6 benchmarks in section 4.2. Note that all the results of MixMIM are obtained by conducting supervised finetuning of the encoder with self-supervised pretraining, without extra intermediate finetuning (Bao et al., 2021).

4.1 Results on ImageNet-1K

Comparisons with other MIM approaches. Table 2 presents the comparison between MixMIM and state-of-the-art Masked Image Modeling (MIM) works. Our MixMIM can obtain higher accuracy while requiring fewer epochs for pretraining. In particular, our MixMIM-B achieves 84.8% top-1 accuracy with 300 epochs of pretraining, 1.6% better than BEiT (Bao et al., 2021) with 62.5% fewer epochs for pretraining. Besides, our MixMIM also enjoys longer pretraining as previous methods (He et al., 2021). Specifically, we obtain strong 85.1% top-1 accuracy with only 600 epochs of pretraining.

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 (Sun et al., 2017). ‡ results are from (Wei et al., 2022). ⋄ denotes our implementation with the official code. FT and LIN denote top-1 accuracy on ImageNet-1K after finetuning and linear probing respectively.

Method	Backbone	FLOPs (G)	Param. (M)	Supervision	Pretrain Epochs	FT	LIN
ViT (Dosovitskiy et al., 2021)	ViT-B	17.5	86	RGB	14 †	79.9	-
BEiT (Bao et al., 2021)	ViT-B	17.6	87	DALL-E	800	83.2	37.6
CAE (Chen et al., 2022b)	ViT-B	17.5	86	DALL-E	800	83.6	68.6
MAE (He et al., 2021)	ViT-B	17.5	86	RGB	1600	83.6	67.8
MaskFeat (Wei et al., 2021)	ViT-B	17.5	86	HOG	300	83.6	-
data2vec (Baevski et al., 2022)	ViT-B	17.5	86	Feature	800	84.2	-
iBOT (Zhou et al., 2021b)	ViT-B	17.5	86	Momentum	1600	84.0	79.5
PeCo (Dong et al., 2021)	ViT-B	17.5	86	MoCo v3	800	84.5	-
EsViT‡ (Li et al., 2022)	Swin-B/W14	16.3	87	Momentum	300	83.9	**81.3**
SimMIM (Xie et al., 2021)	ViT-B	17.5	86	RGB	800	83.8	56.7
SimMIM (Xie et al., 2021)	Swin-B	15.6	88	RGB	800	84.0	-
SimMIM (Xie et al., 2021) ⋄	MixMIM-B	16.3	88	RGB	300	84.1	20.2
MixMIM	ViT-B	17.5	86	RGB	600	83.8	67.0
MixMIM	Swin-B	15.6	88	RGB	600	84.6	61.2
MixMIM	MixMIM-B	16.3	88	RGB	300	84.8	63.8
MixMIM	MixMIM-B	16.3	88	RGB	600	**85.1**	71.0

Table 3: Comparison with state-of-the-art MIM works using the same encoder. We report the finetuning accuracy on ImageNet-1K.

Pretrain Method	Backbone	Pretrain Epochs	Finetune Epochs	Top-1 Acc.
Supervised (Touvron et al., 2021)	ViT-B	-	300	81.8
MAE (He et al., 2021)	ViT-B	1600	100	83.6
BEiT (Bao et al., 2021)	ViT-B	800	100	83.2
MixMIM	ViT-B	600	100	**83.8**
Supervised (He et al., 2021)	ViT-L	-	200	82.5
MAE (He et al., 2021)	ViT-L	1600	50	85.9
MixMIM	ViT-L	600	50	**86.0**
Supervised (Liu et al., 2021)	Swin-B	-	300	83.5
SimMIM (Xie et al., 2021)	Swin-B	800	100	84.0
MixMIM	Swin-B	600	100	**84.6**
Supervised (Xie et al., 2021)	Swin-L	-	300	83.5
SimMIM (Xie et al., 2021)	Swin-L	800	100	85.4
MixMIM	Swin-L	600	50	**85.8**
Supervised (Wang et al., 2021)	PVT-L	-	300	81.7
MixMIM	PVT-L	300	50	**83.2**

While previous works design various reconstruction targets to speed up the pretraining process (Bao et al., 2021; Wei et al., 2021), our MixMIM reconstructs simply normalized pixels (He et al., 2021) and demonstrates strong pretraining efficiency. Compared to MaskFeat (Wei et al., 2021), our MixMIM obtains +1.2% better accuracy with the same pretraining epochs. PeCo (Dong et al., 2021) proposed to reconstruct the perceptual codebook from a pretrained MoCo v3 network (Chen et al., 2021) and can achieve better performance to some extent. In comparison, our MixMIM obtains even better performance (+0.6%) than PeCo with less pretraining epochs (-200). The superior performance of MixMIM comes from our mixed pretraining as well as the hierarchical vision transformer. However, SimMIM (Xie et al., 2021) also utilizes a hierarchical Swin Transformer (Liu et al., 2021), but its performance is worse (-1.1%) than MixMIM with more pretraining epochs (+200). We list the training time in the Appendix A.2.4.

We test with scaling MixMIM up to 600M parameters, as shown in Figure 2 (Left). MixMIM has better FLOPs vs. accuracy tradeoff than other approaches. In particular, MixMIM-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 MixMIM to other backbones. While previous works (He et al., 2021; Wei et al., 2021) may be restricted to a specific architecture, our proposed MixMIM can generalize to various visual backbones, including plain ViT (Dosovitskiy et al., 2021), Swin Transformer (Liu et al., 2021), and PVT (Wang et al., 2021). We conduct a thorough comparison with other MIM approaches with fixed encoders. As shown in Table 3, MixMIM consumes the same or fewer epochs for pretraining but obtains consistently better performance on hierarchical ViTs. In particular, our MixMIM achieves 84.6% top-1 accuracy with Swin-B, +0.6% better than SimMIM (Xie et al., 2021) while requiring

Table 4: Comparison with other self-supervised approaches on COCO and ADK20K. We report AP^{box} and AP^{mask} on COCO, and mIoU on ADE20K. The results of BEiT and MoCo v3 are from MAE (He et al., 2021). The results of EsViT are from (Wei et al., 2022). † denotes using supervised finetuning on ImageNet.

Method	Backbone	Pretrain Epochs	FLOPs (G)	Params. (M)	COCO AP^{box}	COCO AP^{mask}	FLOPs (G)	Params. (M)	ADE20K mIoU
MoCo v3 (Chen et al., 2021)	ViT-B	300	853	116	47.9	42.9	606	164	47.3
BEiT (Bao et al., 2021)	ViT-B	800	853	116	49.8	44.4	606	164	47.1
MAE (He et al., 2021)	ViT-B	1600	853	116	50.3	44.9	606	164	48.1
iBOT (Zhou et al., 2021b)	ViT-B	1600	-	-	51.2	44.2	-	-	50.0
EsViT (Xie et al., 2021)	Swin-B	300	-	-	-	-	-	-	47.3
SimMIM (Xie et al., 2021)	Swin-B	800	-	-	52.3	-	-	-	52.8†
SimMIM (Xie et al., 2021)	MixMIM-B	300	701	110	51.1	45.4	302	122	48.9
MixMIM	MixMIM-B	300	701	110	52.3	46.4	302	122	49.9
MixMIM	MixMIM-B	600	701	110	**52.7**	**47.0**	302	122	**51.1**
MoCo v3 (Chen et al., 2021)	ViT-B	300	1907	339	49.3	43.9	877	392	49.1
BEiT (Bao et al., 2021)	ViT-B	800	1907	339	53.3	47.1	877	392	53.3
MAE (He et al., 2021)	ViT-L	1600	1907	339	53.3	47.2	877	392	53.6
SimMIM (Xie et al., 2021)	Swin-L	800	-	-	53.8	-	-	-	53.5†
MixMIM	MixMIM-L	600	1119	319	**54.3**	**48.2**	460	236	**53.8**

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

Method	Backbone	FLOPs (G)	Params. (M)	INat2018	INat2019	Places205	Places365	Average
DINO (Caron et al., 2021)	ViT-B	17.5	86	72.6	78.2	-	-	-
MAE (He et al., 2021)	ViT-B	17.5	86	75.4	80.5	63.9	57.9	69.4
MixMIM	MixMIM-B	16.3	88	**78.2**	**83.3**	**68.6**	**59.0**	**72.3**
MAE (He et al., 2021)	ViT-L	61.3	304	80.1	83.4	65.8	59.4	72.1
MixMIM	MixMIM-L	35.8	235	**80.6**	**84.4**	**69.3**	**59.6**	**73.5**

200 fewer epochs for pretraining. With Swin-L, our MixMIM obtains 85.8% top-1 accuracy with 600 epochs of pretraining and 50 epochs of finetuning, showing higher efficiency than SimMIM. Besides, our MixMIM achieves 83.2% top-1 accuracy with PVT-L (Wang et al., 2021), improving the supervised baseline by a non-trivial margin.

4.2 Results of Transferring to Downstream Tasks

To further demonstrate the effectiveness of the visual representations learned by MixMIM, we transfer MixMIM to various visual benchmarks with settings described in section 3.

Object detection and instance segmentation. We show the results on COCO (Lin et al., 2014) in Table 4. By pretraining for 600 epochs on ImageNet-1K, our MixMIM-B achieves 52.7 AP^{box} and 47.0 AP^{mask}, surpassing previous self-supervised approaches with fewer FLOPs and parameters. Compared to BEiT (Bao et al., 2021), our MixMIM obtains higher AP^{box} (+2.9) and AP^{mask} (+2.6) with less pretraining epochs (-200). Note that our hierarchical backbone can naturally be transferred to object detection without re-designing (Li et al., 2021) network architectures such as FPN (Lin et al., 2017).

Our MixMIM can also scale up to larger models in object detection task and obtains better performance. As shown in Table 4, our MixMIM-L achieves 54.3 AP^{box} (48.2 AP^{mask}), +1.0 (+1.1) better than BEiT while requiring 200 less epochs for pretraining and 41% less FLOPs for inference. We further evaluate the tradeoff of FLOPs vs. AP^{box} in Figure 2 (Middle). We also found that MixMIM outperforms other approaches by large margins.

Semantic segmentation. Table 4 also presents the results of MixMIM on ADE20K (Zhou et al., 2017). We compare its Mean Intersection over Union (mIoU) on ADE20K with other self-supervised approaches. Our MixMIM-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 scaling up the model to MixMIM-L. Thanks to the hierarchical design, our MixMIM consumes much fewer FLOPs for inference compared to other approaches with plain ViT. In Figure 2 (Right), our MixMIM outperforms other approaches by large margins.

Image classification. We further transfer MixMIM 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 (Zhou et al., 2014) for MAE-B (He et al., 2021). However, our MixMIM

Figure 3: Examples images for different filling contents.

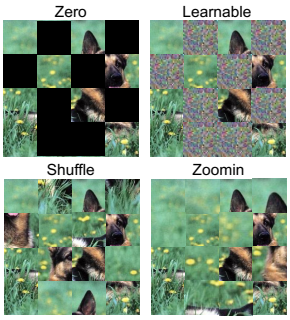

Table 6: Filling content.

Type	Top-1 Acc.	mIoU
Mix	**84.6**	**49.9**
Zero	84.1	48.0
Learnable	84.1	48.9
Shuffle	82.6	43.0
Zoomin	83.5	44.9

Table 7: Pretraining epochs.

# Epochs	Top-1 Acc.	mIoU
300	84.6	49.9
600	**85.1**	50.3
900	**85.1**	**51.0**

Table 8: Number of mixing images.

# Images (ratio)	Top-1 Acc.	mIoU
2 (0.5)	84.6	49.9
2 w/ [M] (0.75)	84.4	49.0
3 (0.67)	84.7	49.9
4 (0.75)	**84.8**	**49.9**
5 (0.8)	84.5	49.5

Table 9: Dual reconstruction.

Dual	Top-1 Acc.	mIoU
✓	**84.6**	**49.9**
✗	84.0	47.3

Table 10: Ablation on reducing the difficulty of the pretext task. We report the top-1 accuracy on ImageNet-1K for each approach with different pretraining epochs. Ours w/o unmixing denotes that we do not reduce the difficulty of the pretext task. Details of the other two approaches are described in section 2.3.

Approach	FLOPs (G)	Params. (M)	300	600	900
Ours w/o unmixing	115	19.0	84.4	84.4	84.4
Ours w/ mix embedding	115	19.1	84.4	84.6	84.8
Ours w/ masked self-attention	115	19.0	84.6	**85.1**	**85.1**

can still outperform previous self-supervised approaches. In particular, our MixMIM-B has an average +2.9% performance gain compared to MAE-B. Besides, our MixMIM-L has an average +1.4% performance gain over MAE-L while requiring only 58% FLOPs for inference.

5 Ablation Studies

In this section, we ablate the key designs of MixMIM and report the transferring results of each ablation.Unless otherwise specified, we pretrain MixMIM-B for 300 epochs with a masking ratio of 50%. By default, we report the top-1 accuracy on ImageNet-1K (Deng et al., 2009) and mIoU on ADE20K (Zhou et al., 2017). We also show the results on COCO Lin et al. (2014) in Appendix A.2.1.

Content to filling. While MixMIM default fills the masked tokens of one image with visible tokens from another image, we also explore more design choices. Specifically, we try to fill the masked tokens with the following contents.

- **Zero:** Filling the masked tokens with zeros. This approach causes serious mismatches between the masked tokens and the visible tokens.
- **Learnable:** Following previous works (Bao et al., 2021; Xie et al., 2021), we fill the masked tokens with a shared learnable token. The difference between the zero approach is that learnable tokens can be adapted to visible tokens to match the distribution of the training set.
- **Shuffle:** We randomly shuffle the masked tokens, and then fill the masked locations with the shuffled tokens.We note that this approach is similar to solving jigsaw puzzles (Noroozi & Favaro, 2016) with the difference that we need to fully regress the pixels.
- **Zoomin:** We zoom in the original image and randomly crop an image patch with the size of the original image. We then fill the masked tokens with tokens from the cropped image. This approach also provides masking tokens that are similar to visible ones but is harder than the shuffle approach.

We visualize the four approaches in Figure 3. 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 the learnable approach has a similar performance on ImageNet-1K but better performance on ADE20K compared to Zero. We hypothesize that the training-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 pretraining loss. However, the learned representation quality is lower.

Dual reconstruction. We ablate the proposed dual reconstruction in Table 9. We find that dual reconstruction greatly boosts the performance on downstream tasks. The performance gap is larger

on ADE20K, where we observe +2.6 mIoU with the dual reconstruction. Note that the computation overhead of dual reconstruction is negligible as the decoder is lightweight.

Masking ratio. Our MixMIM implements different masking ratios by mixing more images at inputs. In addition, we also experiment to add $[\mathrm{MASK}]$ 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 $[\mathrm{MASK}]$ 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 MixMIM can achieve strong performance with few pretraining epochs. We ablate 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 2.3, 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.

6 Related Works

Inspired by BERT (Devlin et al., 2019) for Masked Language Modeling, Masked Image Modeling(MIM) becomes a popular pretext task for visual representation learning (Bao et al., 2021; He et al., 2021; Atito et al., 2021). 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 (Xie et al., 2021) 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 (He et al., 2021) 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 (Chen et al., 2022b) separates the encoder and decoder explicitly by adding a feature alignment module in the middle of them. Jean-Baptiste et al. (Alayrac et al., 2019) proposes to learn representations by reconstructing original videos from synthetically mixed one.

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 (Bao et al., 2021) utilizes a pretrained discrete VAE (dVAE) (Rolfe, 2016; Ramesh et al., 2021) as the tokenizer. However, the originally used MSE loss in dVAE is insufficient to force the tokenizer to capture high-level semantics. PeCo (Dong et al., 2021) finds that applying 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 (Bao et al., 2021) needs to be offline pretrained, which limits the model's adaption. To address the problem, iBOT (Zhou et al., 2021b) proposed to use an online tokenizer to generate the visual tokens.

7 Discussion and Conclusion

This paper proposes Mixed and Masked Image Modeling (MixMIM) for efficient visual representation learning. Our MixMIM uses a mixed input created by mixing two images with random masks, and applies dual reconstruction to recover the original two images from the mixed input. We further explore a simpler but stronger hierarchical Transformer for efficient learning. Empirical results on 7 visual benchmarks demonstrate MixMIM 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.

REFERENCES

Jean-Baptiste Alayrac, Joao Carreira, and Andrew Zisserman. The visual centrifuge: Model-free layered video representations. In *CVPR*, 2019.

Sara Atito, Muhammad Awais, and Josef Kittler. Sit: Self-supervised vision transformer. *arXiv:2104.03602*, 2021.

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. *arXiv:2202.03555*, 2022.

Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. In *ICLR*, 2021.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020.

Tianlong Chen, Zhenyu Zhang, Yu Cheng, Ahmed Awadallah, and Zhangyang Wang. The principle of diversity: Training stronger vision transformers calls for reducing all levels of redundancy. In *CVPR*, 2022a.

Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning. *arXiv:2202.03026*, 2022b.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, and Nenghai Yu. Peco: Perceptual codebook for bert pre-training of vision transformers. *arXiv:2111.12710*, 2021.

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

Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, 2021.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv:1706.02677*, 2017a.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter N. Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017b.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Doll'ar, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2021.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *ICML*, 2020.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *ECCV*, 2020.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, 2019.

Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning. In *ICLR*, 2022.

Yanghao Li, Saining Xie, Xinlei Chen, Piotr Dollar, Kaiming He, and Ross Girshick. Benchmarking detection transfer learning with vision transformers. *arXiv:2111.11429*, 2021.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

Guilin Liu, Kevin J Shih, Ting-Chun Wang, Fitsum A Reda, Karan Sapra, Zhiding Yu, Andrew Tao, and Bryan Catanzaro. Partial convolution based padding. *arXiv:1811.11718*, 2018.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

Zhuang Liu, Hanzi Mao, Chaozheng Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2017.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021.

Jason Tyler Rolfe. Discrete variational autoencoders. In *ICLR*, 2016.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *arXiv:2203.12602*, 2022.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *ICLR*, 2021.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.

Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. *arXiv:2112.09133*, 2021.

Yixuan Wei, Han Hu, Zhenda Xie, Zheng Zhang, Yue Cao, Jianmin Bao, Dong Chen, and Baining Guo. Contrastive learning rivals masked image modeling in fine-tuning via feature distillation. *arXiv:2205.14141*, 2022.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018.

Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv:2111.11429*, 2022.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *CVPR*, 2021.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2017.

Bolei Zhou, Àgata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NeurIPS*, 2014.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.

Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv:2103.11886*, 2021a.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. In *ICLR*, 2021b.

A Appendix

A.1 Hyperparameters of Pretraining and Finetuning

We include details about the hyperparameters for reimplementation.

Pretraining. The default setting is in Table 11. We use xavier_uniform (Glorot & Bengio, 2010) to initialize all Transformer blocks following original ViT (Dosovitskiy et al., 2021). We by default use batch size of 1024 and scale the learning rate with linear rule (Goyal et al., 2017a): lr=base_lr $\times$ batch_size / 256.

Table 11: Pretraining on ImageNet-1K.

config	value
optimizer	AdamW (Loshchilov & Hutter, 2017)
base learning rate	1.5×10^{-4}
weight decay	0.05
optimizer momentum	β_1, β_2=0.9,0.95 (Chen et al., 2020)
learning rate schedule	cosine decay (Loshchilov & Hutter, 2016)
warmup epochs	40
augmentation	RandomResizedCrop

Table 12: Finetuning on ImageNet-1K.

config	value
optimizer	AdamW
base learning rate	5×10^{-4}
layer-wise lr decay (Bao et al., 2021; Clark et al., 2020)	0.7
batch size	1024
weight decay	0.05
optimizer momentum	β_1, β_2=0.9,0.999
learning rate schedule	cosine decay
warmup epochs	5
training epochs	100 (B), 50 (L/H)
augmentation	RandAug(9, 0.5) (Cubuk et al., 2020)
LabelSmooth (Szegedy et al., 2016)	0.1
Mixup (Zhang et al., 2017)	0.8
CutMix (Yun et al., 2019)	1.0
drop path (Huang et al., 2016)	0.15 (B), 0.2 (L), 0.3 (H)

Finetuning on ImageNet-1K. The default setting is in Table 12. We use layer-wise learning rate decay following (Bao et al., 2021; Clark et al., 2020). The decay ratio is swept in $\{0.7, 0.75, 0.8\}$, and we find 0.7 performs best. Following pretraining, the learning rate is scaled with linear rule: lr=base_lr $\times$ batch_size / 256.

Finetuning on other classification datasets. We reuse the setting in Table 12. We adjust the drop path rate for each dataset.

Finetuning on COCO. We use the Mask RCNN (He et al., 2017) framework with the encoder of MixMIM as its backbone. We follow the training setting in (Li et al., 2021; He et al., 2021). In particular, we use large-scale jitter (Ghiasi et al., 2021) augmentation with 1024$\times$1024 resolution and [0.1, 2.0] scale range. We use step learning rate schedule with 0.25 epochs of warmup. We finetune MixMIM-B/-L for 55/80 epochs. We use a layer-wise learning rate and set the decay ratio to 0.85/0.9 for MixMIM-B/-L.

Finetuning on ADE20K. We use the UperNet (Xiao et al., 2018) framework with the encoder of MixMIM as its backbone. We finetune for 16K iterations with a batch size of 16. We use the layer-wise learning rate and set the decay ratio to 0.85/0.9 for MixMIM-B/-L. We adopt others settings from BEiT (Bao et al., 2021).

A.2 Additional Results of Ablation Studies

A.2.1 Ablation results on COCO

We show more results of our ablation studies on COCO benchmark in Table 13 14 15 16. We find that the performance on the COCO is similar to that on ADE20K.

Table 13: Ablation on Filling content.

Type	AP^{box}	AP^{mask}
Mix	**51.5**	**45.9**
Zero	51.0	45.3
Learnable	50.9	45.1
Shuffle	46.5	41.6
Zoomin	47.9	42.6

Table 14: Ablation on dual reconstruction.

# Images (ratio)	AP^{box}	AP^{mask}
2 (0.5)	51.5	45.9
2 w/ [M] (0.75)	51.2	45.4
3 (0.67)	51.6	45.9
4 (0.75)	**52.3**	**46.4**
5 (0.8)	51.4	45.4

Table 15: Ablation on Filling content.

# Epochs	AP^{box}	AP^{mask}
300	51.5	45.9
600	52.2	46.5
900	**52.4**	**46.7**

Table 16: Ablation on dual reconstruction.

Dual	AP^{box}	AP^{mask}
✓	**51.5**	**45.9**
✗	50.0	44.4

A.2.2 Finetuning with higher resolution

We finetune the pretrained MixMIM-B with higher resolution on ImageNet-1K and show the results in Table 17. Note that we also scale up the window size when finetuning with higher resolution. We achieved 86.3% top-1 accuracy on ImageNet-1K with 384×384 resolution. In comparison, BEiT(Bao et al., 2021) obtained 84.6% top-1 accuracy with the same resolution.

A.2.3 Training MixMIM-B from scratch

We train MixMIM-B on ImageNet-1K from scratch with the training recipe from Swin Transformer Liu et al. (2021). We obtained 83.8% top-1 accuracy on ImageNet-1K (vs. 83.5% of Swin-B).

A.2.4 Pretraining Time Comparison

We compare the wall-clock time of the pretrain in Table 18. The pretrain time is measured on 8 NVIDIA-A100-SXM-80GB GPUs with a total batch size of 1024.

A.3 Comparison between using image mixing and learnable mask tokens

We perform an empirical evaluation of the learned representations with using image mixing (Mix) and [MASK] token (Learnable).

A.3.1 Attention distance

We follow the procedural in ViT Dosovitskiy et al. (2021) to visualize the attention distance. Note that we normalize the attention distance with the window size for better visualization. We show the visualization in Figure 4. We find that the attentive distances of different heads are more diverse in the Mix approach. In other words, the Mix approach tends to use more diverse information for reconstruction. We further visualize the average attention map (averaged across heads) in Figure 5 and find that the Mix approach can attend to more diverse pixels for reconstruction. In contrast, the Learnable approach attends to more short-range pixels.

A.3.2 KL divergence

We compute the Kullback–Leibler (KL) divergence between the attention maps of different heads in each layer and show the visualization in Figure 6. We find that the Mix approach has more diverse attention maps in middle layers (layers 2-20) in different heads. As pointed out by previous works (Chen et al., 2022a; Zhou et al., 2021a), attention diversity is beneficial for improving expression ability and can therefore lead to better performance. Our ablation in Table 6 also verified the observation.

Table 17: Performances with different resolutions. We report the top-1 accuracy on ImageNet-1K.

224×224	256×256	384×384
85.1	85.5	86.3

Table 18: Wall-clock time comparison of MIM methods.

Method	Backbone	Pretrain epochs	Pretrain Time (GPU hours)	Top-1 Acc.
SimMIM Xie et al. (2021)	Swin-B	800	116	84.0
MAE He et al. (2021)	ViT-B	1600	123	83.6
BEiT Bao et al. (2021)	ViT-B	800	151	83.2
MixMIM	ViT-B	600	125	83.8
MixMIM	Swin-B	600	85	84.6
MixMIM	MixMIM-B	300	64	84.8
MixMIM	MixMIM-B	600	127	85.1

A.3.3 Representation Structures via CKA similarity

We calculate the centered kernel alignment (CKA) Kornblith et al. (2019) similarity between features of different stages, and show the visualization in Figure 7. We find that stage 3 of the Mix approach contains more low-level information than the Learnable approach. In addition, stage 4 of the Mix approach contains more low-level but less middle-level information than the Learnable approach.

A.4 Extend to ConvNets

While our MixMIM uses a hierarchical Transformer as the encoder, we also explore popular ConvNets. In particular, we use ResNet50x3 and ResNet101x3 as the encoder and compare the finetuning results on ImageNet-1K with BiT (Kolesnikov et al., 2020). To reduce the difficulty of the pretext task, we extend the idea of partial convoluation (Liu et al., 2018) and propose a *mixed* version, as illustrated in Figure 8.

We compare the results in Table 19. In particular, our MixMIM outperforms BiT-S by a large margin with half the input size. We note that BiT-M achieves better results by pretraining with 10× larger dataset ImageNet-21K. We believe the results of MixMIM can be further improved by using much larger datasets as shown by Bao et al. (2021), and we leave it as future work.

A.5 Limitation and Potential Negative Societal Impact.

While our MixMIM enables efficient visual representation learning, it may be further improved by mixing with priors, such as only mixing the salient regions (Kim et al., 2020). This work does not have a direct negative societal impact. However, we should be aware that our pretrained neural network can be further finetuned on applications that involving discrimination.

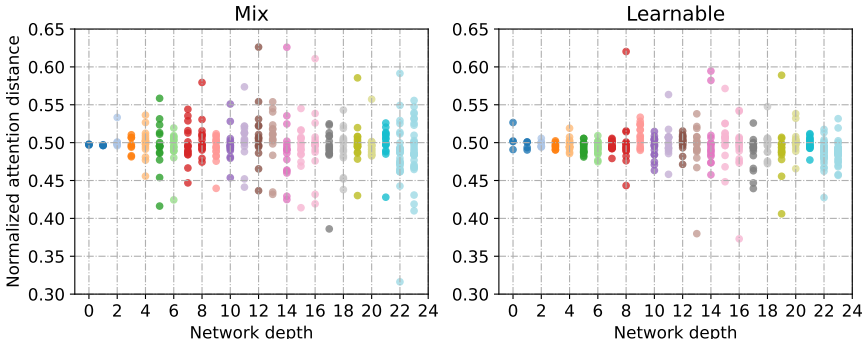

Figure 4: The normalized attention distance in different attention heads.

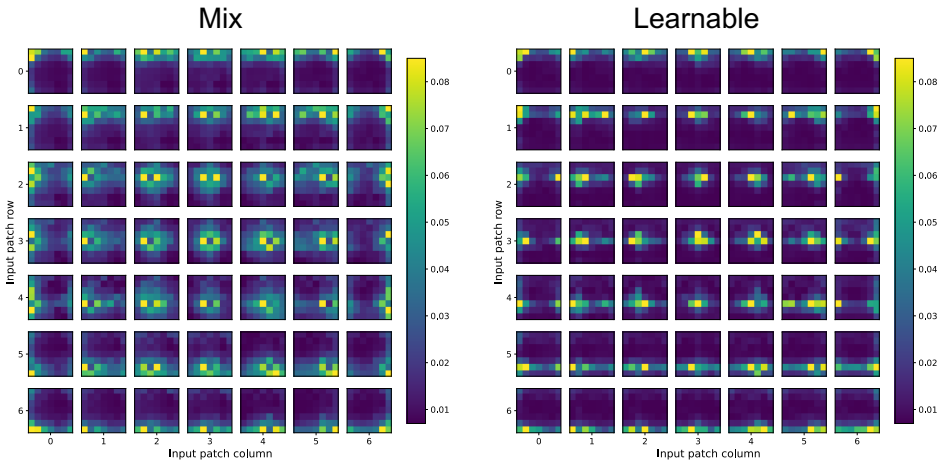

Figure 5: The attention maps of different positions.

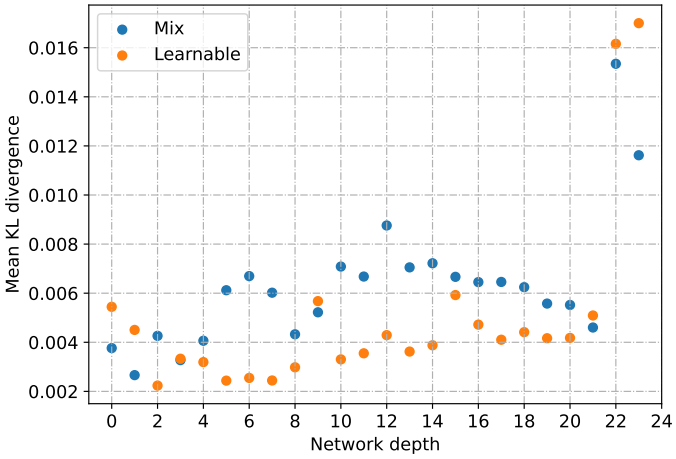

Figure 6: The KL divergence between attention distributions of different heads. The KL divergence is averaged across the attention heads.

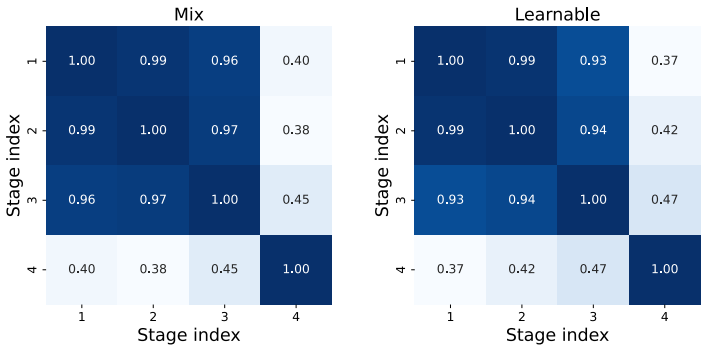

Figure 7: The CKA similarities between feature maps of different stages.

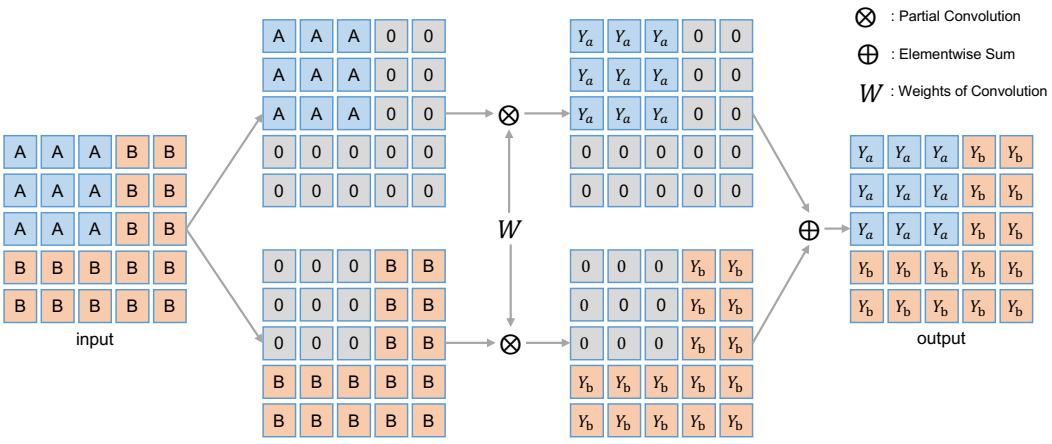

Figure 8: Mixed convolution.

Table 19: **Results on ConvNets.** All results of MixMIM are obtained by pretraining for 300 epochs and finetuning for 100 epochs on ImageNet-1K. We report the top-1 accuracy on ImageNet-1K.

Method	Backbone	Input Size	Pretrain Data	Top-1 Acc.
BiT-S (Kolesnikov et al., 2020)	Res50x3	448×448	ImageNet-1K	80.0
BiT-M (Kolesnikov et al., 2020)	Res50x3	448×448	ImageNet-21K	84.0
MixMIM	Res50x3	224×224	ImageNet-1K (w/o labels)	81.8
BiT-S (Kolesnikov et al., 2020)	Res101x3	448×448	ImageNet-1K	80.3
BiT-M (Kolesnikov et al., 2020)	Res101x3	448×448	ImageNet-21K	84.3
MixMIM	Res101x3	224×224	ImageNet-1K (w/o labels)	82.6