ISSN 2616-5775 Vol. 7, Issue 5: 101-108, DOI: 10.25236/AJCIS.2024.070513

# The contrastive learning algorithm based on masked image modeling

**Feng Xin[1,2,a], Li Xinwei[1,2,*]**

*[1]School of Electrical Engineering and Automation, Henan Polytechnic University, Jiaozuo, China*
*[2]Henan Key Laboratory of Intelligent Detection and Control of Coal Mine Equipment, Jiaozuo, China*
*[a]630412107@qq.com*
*[*]Corresponding author*

***Abstract:*** *Contrastive learning aims to train an encoder in a self-supervised manner to obtain representations of input images. Traditional contrastive learning methods primarily model relationships between image categories. This approach allows the model to learn representations with strong instance discriminability but overlooks representations of local perceptibility in images. Therefore, to enhance the feature representation capability of contrastive learning, a method called Masked Contrastive Learning (MCLim) is proposed, which is based on masked image modeling. MCLim introduces the idea of masked image modeling in contrastive learning, utilizing a dual-branch contrastive learning structure. The first branch processes the image after being masked and fed into the network, enabling the model to simultaneously perform contrastive learning and restore the original input image. This approach aims to learn representations with both instance discriminability and local perceptibility. MCLim employs a method to restrict the size of the masked area to prevent negative impacts on model performance from feature augmentation. It adapts to various encoding networks by changing the size of the masked block. The projection layer in the first branch serves as both a decoder and reconstructs the input image. Experimental classification using MCLim on the Cifar10 dataset yields a Top1 accuracy of 89.23%, outperforming other algorithms in the same category.*

***Keywords:*** *contrastive learning; mask; feature expression; instance discriminability; local perceptibility*

## 1. Introduction

In recent years, deep learning has achieved rapid development in the field of computer vision. However, deep learning typically relies on a massive amount of annotated data for model training to achieve good performance.When there is a scarcity of annotated data and an abundance of unannotated data, enhancing the feature representation capability of deep learning becomes a critical need. Self-supervised learning is one effective approach to address this issue, enabling the use of large amounts of unannotated data for self-supervised training to improve feature extraction models[1].Contrastive learning, a typical method of self-supervised learning, leverages the ideas of comparing similar dataand instance discrimination tasks. It considers augmented images from the same original as similar samples, and those from different originals as dissimilar. By comparing different views of the same instance, the model learns more discriminative representations, enhancing the neural network's feature extraction capabilities for downstream tasks.

Contrastive learning methods typically use a twin neural network architecture to encode input images and calculate contrastive loss. Factors affecting the quality of representations in contrastive learning include the construction of sample pairs, the choice of encoding networks, and the selection of training tasks. SimCLR[2] constructs positive pairs through rich image augmentations to learn features of similar samples more effectively. MoCo V3[3] introduces a more powerful encoding network, ViT[4], into the contrastive learning framework and stabilizes model training by freezing the first layer, enhancing the encoding capability of contrastive learning. BYOL[5] adds a prediction layer at the end of the first branch to predict the output of the second branch, transforming the contrastive task into a prediction task, thereby improving the model's learning of representations.

Meanwhile, another self-supervised learning method, Masked Autoencoders[6] and SimMIM[7], follows the idea of masked language modeling from natural language processing.These methods randomly mask a large portion of the training images and use neural network models to reconstruct the masked images. By learning the local relationships within input images, these methods can learn internal image representations and achieve excellent results in some computer vision tasks.

Both methods have achieved great success in the field of self-supervised learning, prompting considerations on how to enhance the quality of image representations learned through contrastive learning using masked autoencoder methods. Contrastive Masked Autoencoders[8] incorporate contrastive learning into masked autoencoders, using a twin-network architecture with an online branch and a target branch. The online branch is an asymmetric encoder-decoder, and the target branch is a momentum-updated encoder. During training, the online encoder reconstructs the original image from the representation of the masked image, learning overall features. The target encoder inputs the complete image and enhances the discriminability of the features through contrastive learning with the online encoder, although its main task remains image reconstruction.

Traditional contrastive learning methods focus only on learning discriminative representations of image instances. This paper introduces masked image modeling into contrastive learning, designing a method called MCLim. Unlike existing contrastive learning approaches, MCLim performs masking operations on most areas of the input image in the first branch. The first branch's projection layer output is upsampled to reconstruct the input image, learning both discriminative representations of image instances and local perceptibility representations simultaneously. MCLim uses a novel image augmentation method to prevent negative impacts from image augmentation methods on masked learning and does not introduce additional components, using only upsampling after the projection layer in the first branch to reconstruct the input image.

## 2. Design Methods

The framework of the MCLim model is illustrated in Figure 1.

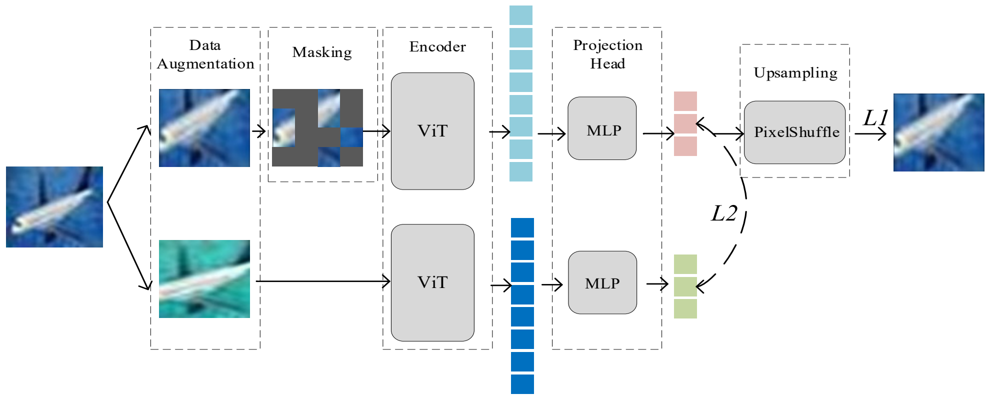

*Figure 1: MCLim framework*

MCLim is based on a twin neural network architecture, where the first branch consists of five modules:Data Augmentation, Masking, Encoding, Projection Head, and Upsampling. The second branch is comprised of three modules: Data Augmentation, Encoding, and Projection Head. The Data Augmentation module transforms the input image into two similar images through different feature enhancement techniques. The Masking module randomly covers parts of the image in the first branch, while the Encoding module encodes the input image into a feature vector using the ViT network. The Projection Head receives high-level feature representations from the encoding network and maps them to a lower-dimensional representation space. The Upsampling module reconstructs the output of the first branch’s Projection Head back to the input image size. MCLim employs a mean squared error loss and a weighted combination of contrastive loss, as shown in the figures. This section will provide a detailed introduction to each part.

### *2.1 Data Augmentation Module*

The goal of the Data Augmentation module in MCLim is to generate similar positive samples through feature enhancement for contrastive learning. Rich image augmentation techniques significantly enhance model performance in contrastive learning methods by enabling the model to learn a more diverse and rich feature expression. This captures the intrinsic structure and patterns of the data, improving the model’s expressive capabilities. Furthermore, feature enhancement increases the model's generalizability, making it more effective in handling new tasks. The model, having learned feature expressions under various transformations, can adapt to different data distributions and changes, thereby generalizing to new tasks. In contrast learning algorithms, both branches typically employ cropping and color distortion strategies. However, when the cropping area is too far apart, the two augmented images lack semantic relevance, which, after masking in the first branch, can amplify these differences and create false positives, thereby hindering contrastive learning. Unlike other contrastive learning methods, literature [8] indicates that feature enhancement techniques such as color distortion can reduce training results in masked image modeling tasks.

Therefore, to balance the characteristics of masked autoencoders and contrastive learning, MCLim employs a new image augmentation method: the input image is randomly cropped and resized to its original size, with the cropping frame limited to 65%-90% of the original image area to prevent negative effects from cropping areas that are too far apart. The first branch’s input image does not undergo feature enhancement transformations to avoid affecting the mask learning performance; the second branch uses feature enhancement methods, such as Color Jitter, Grayscale conversion, and Gaussian blur, to allow the model to learn more image features.

### *2.2 Masking Module*

The purpose of the Masking module in MCLim is to mask the image by a certain proportion, using learnable vectors to replace the masked parts input into the network. In masked image modeling, the masking method generally ensures block alignment. This is because when the size of the mask blocks aligns with the image blocks, edge processing is usually simpler and helps to avoid inconsistencies and artifacts at the edges. MCLim also uses block-aligned masking for the images. MCLim uses a high masking ratio, meaning the masked image blocks make up a significant proportion of the total image blocks and the masking blocks are uniformly distributed. As images are highly spatially redundant natural signals, a high masking ratio largely eliminates the information redundancy between image blocks, making it difficult for the model to easily extrapolate the task from visible neighboring image blocks. Uniform distribution ensures that there are not more masking blocks near the center of the image, avoiding potential center bias.

Learnable vectors are vectors trained as parameters during the model training process, and their values are usually learned automatically during the training process. Because learnable vectors offer greater flexibility, the model can adaptively learn richer and more discriminative representations based on the task and data. In MCLim, learnable vectors replace the masked image blocks, and their dimensions are set to match those of the visible image blocks. Since using a mask affects the semantic information of the image, the second branch's image is input into the encoding network without masking to preserve the semantic integrity and discriminative nature of the image.

### *2.3 Encoding Module*

The purpose of the Encoding module in MCLim is to encode input samples into feature vectors and map them to feature space. Compared to traditional convolutional neural network models, the ViT model uses a self-attention mechanism that captures global information in images more effectively, making it more effective at solving complex problems. Literature [5] introduces the ViT model into contrast learning algorithms and notes that training instability issues in the ViT model within contrast learning algorithms can be resolved by freezing the patch embedding layer parameter updates, enhancing the feature encoding capability of contrast learning.

MCLim's encoding network uses ViT and freezes the patch embedding layer parameter updates. Since the first branch simultaneously undergoes masking and contrast learning, after training is completed, the first branch’s encoding network is retained for downstream task training.

Assuming that the input sample for the first branch is $\left\{x_i^s\right\}_{i=1}^{N}$, the image is partitioned into blocks , where N is the number of image blocks.The visible image blocks are denoted as $\left\{x_i^v\right\}_{i=1}$, and the invisible image blocks are replaced with a learnable vector $\left\{p_i^u\right\}_{i=1}^{u\neq v}$ . After replacement, all image blocks are denoted as $\left\{x_i^v\right\}_{i=1}+\left\{p_i^u\right\}_{i=1}^{u\neq v}$ , which are then input into the first branch encoder to obtain their feature representation $\left\{z_i^s\right\}_{i=1}^N$.

$$\left\{z_i^s\right\}_{i=1}^N = F_1\left(\left\{x_i^v\right\}_{i=1} + \left\{p_i^u\right\}_{i=1}^{u\neq v}\right) \tag{1}$$

Assuming the input sample for the second branch is , the divided image blocks are , which are then input into the second branch encoder to obtain their feature representation $\left\{z_i^s\right\}_{i=2}^N$.

$$\left\{z_i^s\right\}_{i=2}^N = F_2\left(\left\{x_i^s\right\}_{i=2}^N\right) \tag{2}$$

### *2.4 Projection Layer and Upsampling Module*

In MCLim, the projection layer module aims to map the high-dimensional representation of an image to a low-dimensional space, while the purpose of the upsampling module is to rearrange the output of the projection layer through channel shuffling and element reordering, resulting in a tensor that matches the dimensions of the input image. In contrastive learning methods, the projection layer maps the original data to a low-dimensional space and computes contrastive loss in this space, typically consisting of a single Multilayer Perceptronlayer. In masked image modeling methods, the decoder decodes the feature vectors output by the encoder into a reconstructed image and computes the reconstruction loss with the original image, typically using ViT encoding blocks.

In MCLim, the projection layer uses a two-layer MLP, and experiments analyze the impact of different projection layer structures on model performance. Since Pixel Shuffle can reconstruct the details of high-resolution images without adding any additional parameters through feature mappings learned by the network, it is used to reconstruct the output of the first branch.

The specific process involves: projecting the input $\left\{z_i^s\right\}_{i=1}^N$ through the first branching layer $\mathrm{T}_1$ to obtain $\mathrm{z}^1$:

$$\mathrm{z}^1 = \mathrm{T}_1(\left\{z_i^s\right\}_{i=1}^N) \tag{3}$$

The upsampling transformation of $\left\{h_i^s\right\}_{i=1}^N$ results in the reconstructed image block $\left\{\mathrm{y}_i^s\right\}_{i=1}^N$:

$$\left\{\mathrm{y}_i^s\right\}_{i=1}^N = \mathrm{PixelShuffle}(\mathrm{z}^1) \tag{4}$$

The input $\left\{z_i^s\right\}_{i=2}^N$ to the second branching projection layer $\mathrm{T}_2$ to obtain $\mathrm{z}^2$:.

$$\mathrm{z}^2 = \mathrm{T}_2(\left\{z_i^s\right\}_{i=2}^N) \tag{5}$$

### *2.5 Loss Function*

The MCLim model performs both the masked image modelling task and the comparison learning task. For the masked image modelling task, the loss between the predicted value and the true value of the masked image block is calculated using the normalized pixels as the computed values and the Mean Squared Errorloss as the loss function, Equation as 6:

$$\mathrm{MSE} = \frac{1}{n}\sum\left(\left\{x_i^s\right\}_{i=1}^N - \left\{\mathrm{y}_i^s\right\}_{i=1}^N\right)^2 \tag{6}$$

n is the number of samples, $\left\{x_i^s\right\}_{i=1}^N$ is the true value of the first image block, $\left\{\mathrm{y}_i^s\right\}_{i=1}^N$ is the predicted value of the first image block.

For the contrast learning task, the projection layer output is used as the computed value and the infoNCE loss is used as the loss function, as in Eq. 7:

$$\mathrm{L}_i^k = -\log \frac{\exp\left(\left\langle z_i^1, z_i^2 \right\rangle / \tau\right)}{\exp\left(\left\langle z_i^1, z_i^2 \right\rangle / \tau\right) + U_{i,k}} \tag{7}$$

where k$\in \{1,2\}$, $\mathrm{U}_{i,k}$ as in Eq.8, $\exp\left(\left\langle z_i^1, z_i^2 \right\rangle / \tau\right)$ is the NPC multiplier, which shrinks the gradient in the loss function and causes the contrast learning model to rely on large training batches;

$$\mathrm{U}_{i,k} = \sum_{l\in\{1,2\}, j\in[\![1,B]\!],\ \mathrm{j}\neq i} \exp\left(\left\langle z_i^k, z_j^l \right\rangle / \tau\right) \tag{8}$$

In order for the model to still achieve good results on small devices, the NPC multiplier is removed to obtain Decoupled contrastive lossas in Eq.9;

$$\mathrm{L}_{DC,i}^k = -\left\langle z_i^1, z_i^2 \right\rangle / \tau + \log U_{i,k} \tag{9}$$

The overall learning objective of MCLim algorithm is a weighted combination of prediction loss and contrast loss, so the overall loss is defined as in Eq.10;

$$\begin{aligned} L &= \lambda L_1 + L_2 \\ &= \lambda\left(\frac{1}{n}\sum\left(\left\{x_i^s\right\}_{i=1}^N - \left\{\mathrm{y}_i^s\right\}_{i=1}^N\right)^2\right) \\ &+ \left(-\left\langle z_i^1, z_i^2 \right\rangle / \tau + log \sum_{l\in\{1,2\}, j\in[\![1,B]\!],\ \mathrm{j}\neq i} \exp\left(\left\langle z_i^k, z_j^l \right\rangle / \tau\right)\right) \end{aligned} \tag{10}$$

$\lambda$ is the prediction loss weight,n is the number of blocked image blocks, $\left\{x_i^s\right\}_{i=1}^N$ is the true value of blocked image blocks, $\left\{\mathrm{y}_i^s\right\}_{i=1}^N$ is the predicted value of blocked image blocks, $z_i^1$ is the output value of the i-th branch network of the first image, $z_i^2$ is the output value of the i-th branch network of the second image, $\tau \in (0,1)$ is the temperature coefficient, $z_j^l$ is the output value of the rest of the images in the same batch after two branches respectively.

## 3. Experiment and Analysis

### *3.1 Experimental setup and parameters*

The experiments in this paper use Python 3.11 as the programming language, Pytorch as the deep learning framework, and CUDA module to accelerate the network training, and the hardware is the 12th Gen Intel Core i7-12700×20K@5.9GHz12 core CPU, 32GB RAM, and the NVIDIA GeForce RTX 3080 12GB graphics memory GPU.

In this paper, we choose to conduct experiments on CIFAR10 dataset.CIFAR10 dataset consists of 10 classes of 32×32 pixel colour images with the following categories: planes, cars, birds, cats, deer, dogs, frogs, horses, boats, and trucks, and there are 6,000 images in each category. The downstream task is selected as the classification task and the evaluation metric is the TOP1 accuracy on CIFAR10.

### *3.2 Experiment*

In this section, the number of training iterations is set to 100, using the AdamW optimizer, as the learning rate typically scales with an increase in batch size. The learning rate is set $l_r \times \mathrm{BatchSize} = 256$ .where is the base learning rate is $l_r$. During the experiments, the mask ratio is set at 0.6.

Using the MCLim method to compare with other methods, the results are shown in Table 1, where the loss function of the MOCO V3 model is replaced with the DCL loss to avoid the batch size negatively affecting the MOVO V3 model.MCLim improves 1.07% over the comparative learning method, MoCo V3, 1.93% over the masked self-encoder method MAE and 1.71%, proving its stronger ability in learning image representations.

The ViT network is replaced with Swin Transforms and HiViT networks, and since MCLim uses a block-aligned masking strategy, the mask block size is used 32×32 in order to fit its dividing block size.The experimental results show that when Swin Transforms and HiViT networks are used instead of the ViT network as an encoder, the accuracy rates are all have been improved, proving that MCLim can be effectively adapted with other network structures.

Due to hardware limitations, the number of samples in the same training batch used in the MCLim experiments is 16, and the performance of the comparison learning algorithm is positively proportional to the number of samples in the same training batch, therefore, the limitations of the hardware conditions reduce the performance of the comparison learning algorithm to a certain extent, and the DCL loss used in MCLim alleviates this problem to a certain extent. Therefore, MCLim can still achieve good performance under limited hardware conditions.

*Table 1: Comparison of the Proposed Framework with Other Algorithms in This Work*

| | Batch Size | Bankbone | depth | Para(M) | CIFAR10 |
|---|---|---|---|---|---|
| MOCO | 128 | ResNet50 | - | 25.56 | 74.1 |
| SimCLR | 128 | ResNet50 | - | 25.56 | 81.3 |
| MoCo v3 | 32 | ViT | 6 | 43.12 | 88.16 |
| SimMIM | 32 | ViT | 6 | 43.12 | 87.04 |
| MAE | 32 | ViT | 6 | 43.12 | 87.3 |
| CMAE | 16 | ViT | 6 | 43.12 | 87.52 |
| MCLim | 16 | ViT | 6 | 43.12 | **89.23** |
| MCLim | 16 | Swin T | [2,2,6,2] | 49.88 | **89.34** |
| MCLim | 16 | HiViT | [2,2,10] | 35.88 | **90.4** |

### *3.2.1 Impact of Feature Enhancement on Model Performance*

Different from the previous comparative learning method, MCLim uses a new image enhancement method to avoid the negative impact of randomly cropping the region too far away on the model performance and the colour distortion to reduce the mask learning ability: the input image is randomly cropped and adjusted to the original size, limiting the cropping frame to 65%-90% of the original image area; the rest of the transformations are not carried out on the input image of the first branch; and feature distortions are used on the image of the second branch, such as the colour distortion. branch image using feature enhancement methods such as colour distortion.

The data in Table 2 shows that adding appropriate feature enhancement in the second branch can effectively improve the model performance. The model performance is best when using the two feature enhancement methods of random colour distortion and random greyscaling, this is because most of the clipping blocks of the same image have similar colour distributions and the colour histogram itself is sufficient to differentiate between the images, the model can take advantage of this shortcut to solve the task. Therefore, the inclusion of colour distortion combinations in the second branch is crucial for model learning performance.

*Table 2: Impact of Different Feature Enhancement Methods on Model*

| Crop (0.65-0.9) | ColorJitter P=0.8 | Grayscale P=0.2 | Gaussian P=0.8 | Acc.Top1 |
|---|---|---|---|---|
| ✓ | | | | 88.84 |
| ✓ | ✓ | | | 89.1 |
| ✓ | ✓ | ✓ | | **89.23** |
| ✓ | ✓ | ✓ | ✓ | 89.14 |

### *3.2.2 Impact of projection layer on model performance*

Contrast learning algorithms aim to train encoders for use in downstream tasks, and in general contrast learning methods, the projection layer serves to map the raw data into a high-dimensional space for better extraction and representation of the data's features. In MCLim the first branch projection layer not only plays the above role but also needs to reconstruct the original image input. This section investigates the effect of different projection layer structures on the model performance and the results are shown in Table 3.

The experimental results show that the performance of the first branch projection layer using the two-layer MLP model is higher than that of using Linear or ViT block, which is because the first branch projection layer not only needs to map the coded data into the high-dimensional space in order to extract and represent the features of the data in a better way, but also needs to decode these data representations into reconstructed outputs, i.e., the first branch projection layer needs to be involved not only in the comparative learning task but also in the decoding and reconstruction task. also participate in the decoding reconstruction task, while overly complex decoders will take part of the encoding function, resulting in the encoder performance is not fully exploited, and overly simple decoders are unable to take on the decoding task, so that the encoder is involved, resulting in its encoding capacity being shared.

*Table 3: Impact of Projection Head on Model Performance*

| Projection Head | CIFAR 10 |
|---|---|
| Linear | 89.02 |
| 2-layer MLP | **89.23** |
| 1 ViT blocks | 87.77 |

### *3.2.3 Impact of loss function on model performance*

MCLim defines the overall learning objective as a weighted combination of prediction loss and contrast loss, and this section analyses the impact of different training tasks and contrast loss weights on the model performance and proves it in experiments.

The reconstruction loss and prediction loss are combined with the contrast loss respectively, and the experimental results are shown in Table 4. $L_{1,rec}$ is the reconstruction loss, the first branch outputs all the image blocks with the true value to calculate the MSE loss; For the prediction loss $L_{1,pre}$, the first branch is occluded from the image block output with the true value to calculate the MSE loss; For the DCL loss $L_2$. The experimental results show that the accuracy of the model using predictive loss is significantly higher compared to reconstruction loss, this is because using predictive loss, the model focuses its learning ability on the masked part of the image blocks to better understand their content.

*Table 4: Impact of Pretext Tasks on Model Performance*

| Loss | CIFAR 10 |
|---|---|
| $L_{1,rec}+L_2$ | 88.71 |
| $L_{1,pre}+L_2$ | **89.23** |

In order to analyse the effect of predictive loss weights on model performance, the loss weights $\lambda$ are set to 0, 0.5, 1, 1.5 and the experimental results are shown in Table 5. When the loss weight is 0, the model degenerates to the MOCO V3 algorithm using DCL loss, but the first branch uses a large scale masking operation, which makes the semantic information of the image incomplete, resulting in a degradation of the model performance. When the loss weight is increased from 0, the masked image modelling task is introduced in MCLim and the model performance is improved accordingly.

*Table 5: Impact of Contrastive Loss Weight on Model Performance*

| $\lambda$ | CIFAR 10 |
|---|---|
| 0 | 84.7 |
| 0.5 | 89.2 |
| 1 | 89.23 |
| 1.5 | 89.17 |

## 4. Conclusion

In order to improve the representation quality of contrast learning methods, this paper designs a contrast learning method that uses a masking approach to enhance feature representation, and verifies its effectiveness on the downstream task-image classification on the public dataset Cifar10. The method improves the model performance under small training batches by combining contrast learning and mask self-encoder features, adding mask operations to the first branch input image, and using a projection layer to perform contrast learning and image reconstruction simultaneously, learning the maximum similarity of similar images while learning the internal connections of the images, and using the DCL loss for the contrast loss function.

## Acknowledgements

Thank you for the support of Special Funding for Basic Research Operating Costs of Universities in Henan Province (NSFRF220444), Henan Provincial Science and Technology Tackling Project (232102210040).

## References

*[1] ZHANG Chong-Sheng, CHEN Jie, LI Qilong,et al. Deep Contrastive Learning: A Survey[J]. Acta Automatica Sinica. 2023,(1):15-39.*
*[2] Chen T, Kornblith S, Norouzi M, et al. A simple framework for contrastive learning of visual representations[C]//International conference on machine learning. PMLR,2020: 1597-1607.*
*[3] Chen X, Xie S, He K. An empirical study of training self-supervised vision transformers. In 2021 IEEE[C]//CVF International Conference on Computer Vision (ICCV). 9620-9629.*
*[4] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv preprint arXiv:2010.11929, 2020.*
*[5] Grill J B, Strub F, Altché F, et al. Bootstrap your own latent-a new approach to self-supervised learning[J]. Advances in neural information processing systems, 2020, 33: 21271-21284.*
*[6] He K, Chen X, Xie S, et al. Masked autoencoders are scalable vision learners[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 16000-16009.*
*[7] Xie Z, Zhang Z, Cao Y, et al. Simmim: A simple framework for masked image modeling[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 9653-9663.*
*[8] Huang Z, Jin X, Lu C, et al. Contrastive masked autoencoders are stronger vision learners[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.*