Image Representation Learning with Masked Image Modeling Pre-training in Vision Mamba State Space Model

ARDA DUYUM, University of Twente, The Netherlands

Vision Mamba, recognized for its computational and memory efficiency, addresses the need for environmentally sustainable machine learning models. However, it faces challenges in scalability and stability, particularly with large-scale visual tasks such as ImageNet-1k. This paper improves Vision Mamba by integrating Masked Auto-encoders (MAEs) to enhance image representation learning. Specifically, three masking strategies—random, block, and center masking—were implemented and their impact on the model's performance was evaluated. Experiments demonstrate that block masking achieves the highest Structural Similarity Index Measure (SSIM) values, indicating superior image reconstruction quality, while center masking delivers the highest classification accuracy, reaching approximately 0.26 by epoch 20. Conversely, random masking performed the worst in both metrics.

Additional Key Words and Phrases: Bidirectional State Space Model, Masked Image Modeling, Visual Representation Learning, Masked Auto-encoder, Image Classification

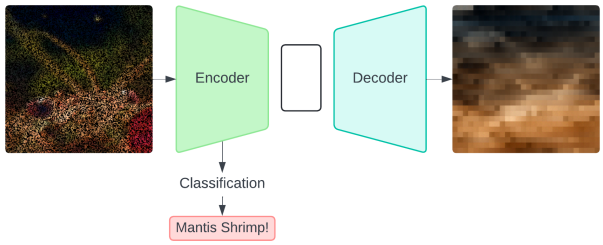

Fig. 1. Architecture of Masked Auto-encoder, including classification logits

1 INTRODUCTION

Convolutional Neural Networks (CNNs) have been the cornerstone of deep learning for vision tasks, known for their ability to hierarchically learn features from images [1]. With advancements in hardware technology, Vision Transformers (ViTs) have gained prominence due to their capability to capture global context through self-attention mechanisms [2]. However, ViTs are computationally expensive and energy-inefficient, posing sustainability challenges [3]. This high computational cost and energy consumption make ViTs less viable for widespread, environmentally sustainable applications.

These problems are addressed in Vision Mamba (Vim) model, which is a promising next-generation backbone for vision foundation models, offering a more efficient alternative. The key innovation of Vision Mamba is the use of bidirectional Mamba blocks, which process images in both forward and backward directions. This bidirectional approach helps to eliminate inductive biases associated with directional processing, thereby enhancing the model's ability to comprehend the global context while maintaining computational and memory efficiency. Vision Mamba leverages bidirectional state space modeling to achieve data-dependent global visual context with significantly lower computational complexity compared to ViTs. Remarkably, Vision Mamba is 2.8 times faster than data-efficient Vision Transformers and saves 86.8% of GPU memory, making it both a high-performance and sustainable solution. [4]

The efficiency comes with a cost for Vision Mamba model, particularly in scalability and stability. One significant challenge is the vanishing/exploding gradients, which hampers the model's performance in large-scale visual tasks such as ImageNet-1k. Additionally, the inherently 1D nature of Mamba's selective scanning technique presents challenges when applied to 2D or higher-dimensional visual data, potentially leading to a loss of critical spatial information. These limitations requires refinement to enhance the model's robustness and applicability.[5]

This paper explores the integration of Masked Autoencoders (MAEs) within the Vision Mamba model. MAEs are particularly useful at managing the spatial redundancy inherent in images. They employ an asymmetric encoder-decoder structure, where the encoder processes only visible patches, significantly reducing computational overhead. Coupling these two designs enables the training of large models efficiently and effectively, accelerating training by 3× or more and improving accuracy. This approach allows for learning high-capacity models that generalize at scale [6].

The aim of this paper is to contribute to the implementation of MAEs within the Vision Mamba model improving the robustness and generalization in order to achieve higher accuracy on the ImageNet-1k dataset [7]. Additionally, this paper investigates the performance of different masking strategies to determine which method yields the best similarity score. The findings indicate that center masking outperforms other strategies in terms of accuracy when pre-trained and fine-tuned on ImageNet-1k.To facilitate achieving these objectives, the following research questions formulated:

(1) *Which masking strategy yields the highest reconstruction quality in the Vision Mamba encoder, as measured by the Structural Similarity Index (SSIM)?*
(2) *What is the impact of Masked Image Modeling (MIM) pretraining on the Vision Mamba model's accuracy when fine-tuned for image classification on the ImageNet-1k dataset?*

The following section provides a review of related work, highlighting improvements made to Vision Mamba. Section 3 describes the methodology, including the mathematical foundation, architecture of the proposed model and the evaluation of masking strategies. Section 4 details the experimental setup, followed by Section 5 which presents the results and discussion of the conducted experiments.

TScIT 41, July 5, 2024, Enschede, The Netherlands

Finally, Section 6 concludes the paper and suggests directions for future research.

2 RELATED WORK

The improvement of visual data representation enhances the performance and applicability of models in downstream tasks such as image classification, segmentation, and detection. Effective visual representation is essential for applications ranging from autonomous driving [8] to medical imaging [9] and robotics [10], where the accuracy and efficiency of models directly impact their practical utility. Vision Mamba (Vim) represents a significant advancement in applying state space models to visual representation learning. Vision Mamba utilizes bidirectional state space modeling to achieve data-dependent global visual context with lower computational complexity compared to ViTs. The versatility is demonstrated through its application in medical image classification [11], 3D medical image segmentation [12], PointMamba for point cloud analysis [13], and DeMamba [14] for AI-generated video detection.

To further enhance the capabilities of Vision Mamba, various pretraining methods have been explored, with Masked Auto-encoders (MAEs) emerging as a particularly promising approach. Prior to MAE, BEiT [15] was the first to apply masked language modeling (MLM) from natural language processing to the visual domain. BEiT introduces a visual vocabulary based on the approach used in NLP, making the training process more complex compared to MAE. BEiT uses a block-wise masking strategy, with random block sizes and aspect ratios, masking approximately 40

SimMIM [16], developed concurrently with MAE, simplifies the pretraining process by directly predicting image patches rather than visual tokens. SimMIM employs a linear layer for the decoder, resulting in lower computational complexity. This model uses larger masking blocks instead of increasing the masking ratio, similar to MAE's random masking strategy.

Following MAE, MaskFeat [17] was introduced, focusing on video prediction. MaskFeat distinguishes itself by using the Histogram of Oriented Gradients (HOG) as the prediction target, instead of directly computing pixel values. This method proved effective in video prediction tasks, using a block-wise masking strategy akin to BEiT.

Other pretraining methods have also been explored to enhance visual representation learning. Contrastive learning techniques, such as SimCLR [18] and MoCo [19], focus on learning representations by contrasting positive and negative pairs. These methods rely heavily on data augmentation to create diverse views of the same image, aiding in learning robust features but often involve high computational costs and complexity. Self-supervised learning approaches, such as BYOL [20] and SwAV [21], aim to learn useful representations without requiring labeled data. These methods typically involve creating different augmentations of the same image and learning to predict one view from another, although they can be resource-intensive and complex to implement.

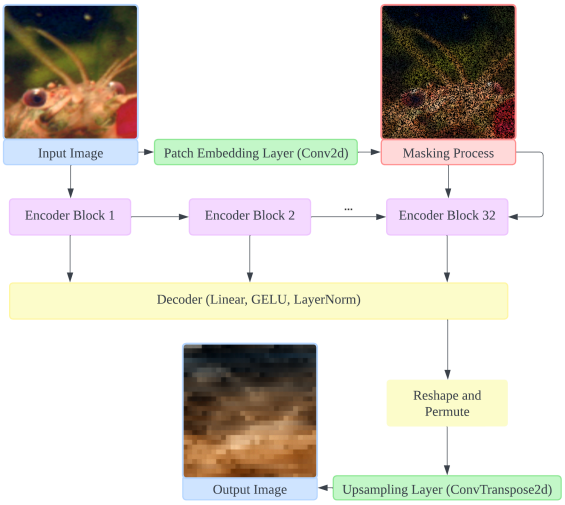

Fig. 2. Architecture of Masked Auto-encoder

3 METHODOLOGY

3.1 Mathematical Foundations

Convolution Operation

The convolution operation [22] in the MAE encoder is used for extracting patches from the input image. The mathematical formulation of the convolution operation is given by:

$$p = W * I + b$$

where:

- p represents the output patches.
- W is the convolution filter (kernel).
- I is the input image.
- b is the bias term.
- $*$ denotes the convolution operation.

Layer Normalization

Layer Normalization (LayerNorm) stabilizes the learning process by normalizing the inputs across the features within each layer [23]. Can be shown as:

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

where:

- x is the input to the layer.
- μ is the mean of the input x.
- σ^2 is the variance of the input x.
- ϵ is a small constant to prevent division by zero.

Gaussian Error Linear Unit (GELU)

The Gaussian Error Linear Unit (GELU) [24] is an activation function that combines the properties of the Rectified Linear Unit (ReLU) and the Gaussian distribution. The GELU function is defined as:

$$\text{GELU}(x) = x \cdot \Phi(x)$$

where:

- x is the input.
- $\Phi(x)$ is the cumulative distribution function (CDF) of the standard normal distribution.

Patch Embedding

Patch embedding is a crucial step in converting the input image into a sequence of patches that can be processed by the transformer architecture [25]. This can be described mathematically as follows:

Given an input image I of size $H \times W \times C$, where H is the height, W is the width, and C is the number of channels, the image is divided into patches of size $P \times P$. The number of patches is $(H \cdot W)/P^2$. Each patch is then flattened into a vector and projected into a higher-dimensional space using a linear transformation.

$$\text{PatchEmbedding}(x) = x \cdot W_e + b_e$$

where:

- x is the flattened patch.
- W_e is the learnable weight matrix.
- b_e is the bias term.

Multi-Head Self-Attention

Multi-head self-attention [26] is a key component of transformer architectures, allowing the model to focus on different parts of the input sequence. The self-attention mechanism can be expressed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where:

- Q (queries), K (keys), and V (values) are derived from the input.
- d_k is the dimension of the queries and keys.
- The softmax function ensures that the attention weights sum to one.

In multi-head self-attention, this mechanism is applied h times (with different learned projections) and the results are concatenated:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W_O$$

where:

- $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$
- W_i^Q, W_i^K, W_i^V are the projection matrices for the i-th head.
- W_O is the output projection matrix.

Positional Encoding

Since transformers do not have a built-in notion of sequence order, positional encoding is added to the input embeddings to provide information about the position of each patch in the sequence [26]. The positional encoding can be defined as:

$$\text{PE}(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)$$

$$\text{PE}(\text{pos}, 2i+1) = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)$$

where:

- pos is the position.
- i is the dimension.
- d_{model} is the dimensionality of the input embeddings.

Residual Connections

Residual connections [27] help mitigate the vanishing gradient problem and allow for deeper networks by adding the input of a layer to its output:

$$\text{Output} = \text{Layer}(x) + x$$

This simple yet effective mechanism helps in training deep neural networks.

Feed-Forward Neural Network (FFN)

In transformer architectures, each encoder and decoder layer contains a feed-forward neural network (FFN) [26] applied to each position separately and identically:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

where:

- W_1 and W_2 are weight matrices.
- b_1 and b_2 are bias terms.
- The ReLU activation function $\max(0, x)$ introduces non-linearity.

Adam Optimizer

The Adam optimizer was chosen for the pretraining due to its adaptive learning rate capabilities, which facilitate faster convergence and effective handling of sparse gradients [28]. Adam uses estimates of the first and second moments of the gradients to adapt the learning rate for each parameter. The optimization process involves the following steps:

Initialization:

- Initialize the first moment vector M and the second moment vector V to zero.
- Initialize the time step iT.

Update Rules:

- Compute the biased first moment estimate M using exponential moving averages:

  $$M = \beta_1 M + (1 - \beta_1)\text{gradients}$$

  where β_1 is typically set to 0.9.
- Compute the biased second moment estimate V using exponential moving averages:

  $$V = \beta_2 V + (1 - \beta_2)\text{gradients}^2$$

  where β_2 is typically set to 0.999.

Bias Correction:

- Correct the bias in the first moment estimate:

  $$\hat{M} = \frac{M}{1 - \beta_1^{iT}}$$

- Correct the bias in the second moment estimate:

  $$\hat{V} = \frac{V}{1 - \beta_2^{iT}}$$

Parameter Update:

Update the parameters θ using the corrected first and second moment estimates:

$$\theta = \theta - \alpha \frac{\hat{M}}{\sqrt{\hat{V}} + \epsilon}$$

where α is the learning rate, and ϵ is a small constant (e.g., 10^{-8}) to prevent division by zero.

The complete formula for updating the parameters W and b can be summarized as follows:

$$\begin{aligned}
M &= \beta_1 M + (1 - \beta_1)\text{gradients} \\
V &= \beta_2 V + (1 - \beta_2)\text{gradients}^2 \\
\hat{M} &= \frac{M}{1 - \beta_1^{iT}} \\
\hat{V} &= \frac{V}{1 - \beta_2^{iT}} \\
\alpha_t &= \alpha \frac{\sqrt{1 - \beta_2^{iT}}}{1 - \beta_1^{iT}} \\
\theta &= \theta - \alpha_t \frac{\hat{M}}{\sqrt{\hat{V}} + \epsilon}
\end{aligned}$$

Where:

- θ represents the model parameters (weights W and biases b).
- M and V are the first and second moment vectors.
- β_1 and β_2 are decay rates for the moment estimates.
- iT is the current time step.
- α is the learning rate.
- ϵ is a small constant to prevent division by zero.

The Adam optimizer is particularly effective for training deep neural networks due to its ability to adaptively adjust the learning rate for each parameter based on estimates of the first and second moments.

Mean Square Error (MSE)

In order to calculate the training and validation loss of the pretraining, MSE is used. MSE quantifies the difference between the predicted values produced by a model and the actual observed values in the data [29], and is defined as follows:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where:

- n is the number of observations.
- y_i represents the actual value of the i-th observation.
- $\hat{y}_i$ represents the predicted value of the i-th observation.
- $(y_i - \hat{y}_i)^2$ is the squared difference between the actual and predicted values for the i-th observation.

The MSE provides a single number that reflects the average squared difference between the predicted and actual values. A lower MSE value indicates a better fit of the model to the data, as it means that the predicted values are closer to the actual values. Conversely, a higher MSE value indicates a poorer fit, as the differences between the predicted and actual values are larger.

Structural Similarity Index (SSIM)

The SSIM [30] is a measure of the similarity between two images, which was used to compare the difference between the original image and the reconstructed image after the masked autoencoder pretraining. calculated as follows:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where x and y are the original and reconstructed images, μ_x and μ_y are the mean intensities, σ_x and σ_y are the standard deviations, σ_{xy} is the covariance, and C_1 and C_2 are constants to stabilize the division by a small denominator.

Cross Entropy Loss

CrossEntropyLoss [31] measures the difference between the true label distribution and the predicted probability distribution produced by the model. The goal is to minimize this difference, thereby improving the model's accuracy.

CrossEntropyLoss is defined as:

$$\text{CrossEntropyLoss} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

where:

- C is the number of classes.
- y_i is the true label for the i-th class (1 if the class is the correct label, 0 otherwise).
- $\hat{y}_i$ is the predicted probability for the i-th class.

For a batch of N examples, the loss is averaged over all examples in the batch:

$$\text{CrossEntropyLoss} = -\frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{C} y_{ji} \log(\hat{y}_{ji})$$

where:

- N is the number of examples in the batch.
- y_{ji} is the true label for the i-th class of the j-th example.
- $\hat{y}_{ji}$ is the predicted probability for the i-th class of the j-th example.

In essence, Cross Entropy Loss evaluates how well the predicted probability distribution matches the true distribution. It penalizes incorrect predictions more heavily when the predicted probability is far from the actual class label. This configuration is selected to balance the learning rate for effective optimization.

Accuracy

The evaluation of the fine-tuned model's performance is conducted using the accuracy metric, calculated as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

The accuracy metric provides a straightforward measure of the model's classification performance, allowing for a clear comparison of different masking strategies.

3.2 Model Architecture

The proposed model is based on the Vision Mamba architecture and incorporates a Masked Auto-encoder for pretraining. The MAE aims to predict masked portions of the input images, enabling the model to learn robust visual features in an unsupervised manner. The overview of the architecture can be found in Figure 2

MAE Encoder

The encoder of the Masked Auto-encoder Mamba architecture starts by converting the input image into a sequence of patches using a convolution operation (`nn.Conv2d`). As previously described in the mathematical foundations, this is achieved by applying a convolution filter W over the input image I, expressed as:

$$p = W * I + b$$

where p represents the output patches, $*$ denotes the convolution operation, and b is a bias term. After patch extraction, these patches undergo a random masking process determined by the `mask_ratio`. This involves randomly replacing a set percentage of these patches with a mask token, introducing an element of data sparsity that forces the encoder to learn robust and generalized representations. The masked patches are then processed through a sequence of transformer-like encoder blocks. Each block typically performs a transformation akin to:

$$\text{Block}(x) = \text{LayerNorm}(x + \text{Mixer}(x))$$

where `Mixer` abstracts a mixing mechanism (similar to multi-head attention [26]) that updates each patch representation based on its neighbors, and `LayerNorm` is a normalization step that stabilizes learning by normalizing data within each layer, as described in the Layer Normalization section.

MAE Decoder

The decoder in the architecture is tasked with reconstructing the original image from the encoded and masked patch representations. It uses a series of linear transformations and non-linear activation functions structured as:

$$\text{Decoder}(x) = \text{LayerNorm}(\text{GELU}(W_2(\text{GELU}(W_1 x + b_1)) + b_2))$$

Here, W_1 and W_2 are the weights of linear layers, b_1 and b_2 are biases, and `GELU` is the Gaussian Error Linear Unit, an activation function used for introducing non-linearity, as previously detailed in the GELU section. After processing through these layers, the decoder reshapes the output back into the dimensions corresponding to the original image patches. This reshaped data is then spatially expanded to match the original image's dimensions through an up-sampling step using a transposed convolution (`nn.ConvTranspose2d`), mathematically described as:

$$\text{Upsampled}(x) = W' * x + b'$$

where W' is the transposed convolution filter and b' is the bias. This reconstructed output aims to approximate the original image, effectively learning to fill in the details for the patches that were masked during the encoding phase.

3.3 Masking Strategies

In this paper, three different masking strategies were tested to evaluate the robustness and reconstruction capabilities of our model. Below is a brief description of each strategy:

- **Center Masking**: In this strategy, the central 75% of the image is masked. This involves placing a square mask in the center of the image, leaving only the border areas visible. The purpose of this strategy is to test the model's ability to reconstruct the most critical part of the image, which is often the central region where the main subject is located (Figure 3).

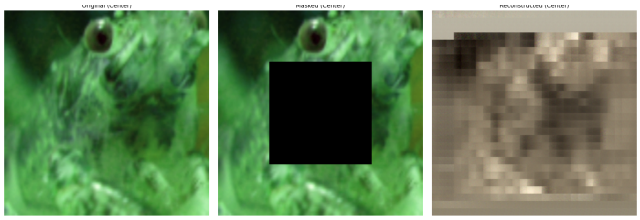

Fig. 3. Center Masking

- **Block Masking**: Here, 75% of the image pixels are randomly masked in blocks. Each block is a single pixel, and the locations of these masked pixels are randomly chosen. This strategy aims to evaluate the model's performance in filling in randomly missing parts of the image, simulating a scenario where random noise or corruption affects various parts of the image (Figure 4).

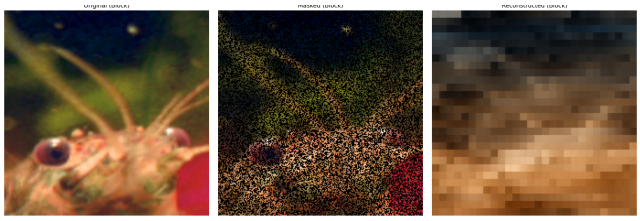

Fig. 4. Block Masking

- **Random Masking**: In this approach, each pixel in the image has a 75% chance of being masked independently of other pixels. This results in a random and dispersed masking pattern. The goal of this strategy is to challenge the model to infer and reconstruct the image with a high degree of randomness and irregular missing areas, which is a common scenario in real-world applications (Figure 5).

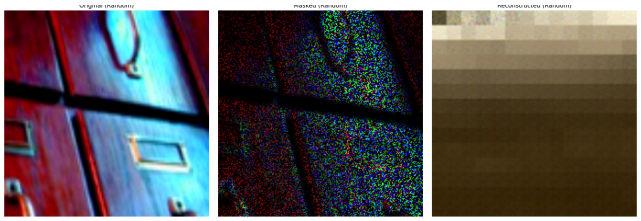

Fig. 5. Random Masking

4 EXPERIMENTAL SETUP

Pretraining Process

The pretraining process involves training the Masked Auto-encoder to reconstruct masked portions of the input images using a subset of the ImageNet-1k dataset [7]. This subset contains 20 classes and is loaded using the Hugging Face datasets library, then split into training and validation sets (80% training, 20% validation). Data augmentation techniques such as resizing, random cropping, horizontal flipping, and color jittering are applied to the images. These techniques are common in contrastive representation learning and help improve the robustness of the model [32].

Training is performed on a single NVIDIA A16 GPU, utilizing the Adam optimizer with a learning rate of 1×10^{-5}. For the loss function, mean squared error (MSE) is employed. The batch size is set to 64 for both training and validation, and the training process runs for up to 20 epochs with early stopping based on validation loss improvement.

The training procedure (Figure 2) includes several key steps. First, the input images are passed through a patch embedding layer and masked according to a predefined mask ratio (75%). The masked patches are then processed by the encoder blocks. During the reconstruction phase, the decoder reconstructs the masked portions of the images, and the reconstruction loss is calculated as the MSE between the original and reconstructed images. This loss is then back-propagated, allowing the optimizer to update the model parameters. After each epoch, the model is evaluated on the validation set, with early stopping implemented to halt training if the validation loss does not improve for 10 consecutive epochs (patience). Additionally, the Structural Similarity Index Measure (SSIM) is used to evaluate the reconstruction quality of the images.

Fine-tuning Process

Following pretraining, the Masked Auto-encoder encoder is fine-tuned for the image classification task using a simplified classifier architecture. The fine-tuning involves loading the pretrained MAE encoder weights and replacing the decoder with a classifier head. The same subset of ImageNet-1k used in pretraining is utilized for fine-tuning, but the focus shifts to classification accuracy. The classifier architecture consists of a patch embedding layer followed by multiple blocks for feature extraction and a final classification layer. The training is performed using the Adam optimizer with a learning rate of 1×10^{-4} and the Cross Entropy Loss function. The training process runs for 20 epochs, with early stopping if the validation loss does not improve for 10 consecutive epochs.

5 RESULTS AND DISCUSSION

The SSIM values for different masking methods across epochs are shown in Table 1. The visualisation of the improvements over the epochs can be seen in Appendix (Figure 7,8, 9, 10). SSIM is used to measure the reconstruction quality of the images:

Epoch	Center Masking	Block Masking	Random Masking
5	0.3372	0.2887	0.4151
10	0.1809	0.3674	0.0084
15	0.3488	0.3441	0.4162
20	0.4023	0.5741	0.4686

Table 1. SSIM values for different masking methods across epochs

From Table 1, it can be observed that the SSIM values fluctuate across epochs, showing varying levels of reconstruction quality. Notably, block masking consistently achieves higher SSIM values compared to center masking and random masking. At epoch 20, block masking achieves an SSIM value of 0.5741, the highest among all masking strategies. This suggests that block masking is more effective in maintaining the structural integrity of the images.

The accuracy comparison of different masking strategies over 20 epochs is illustrated in Figure 6

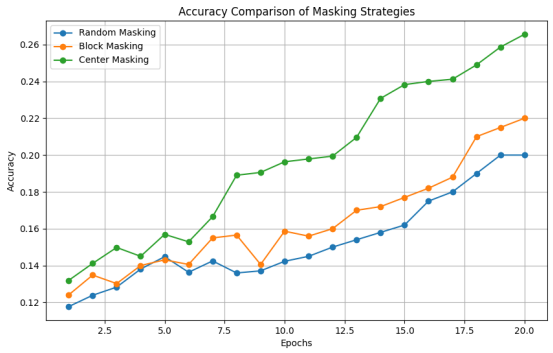

Fig. 6. Architecture of Masked Auto-encoder

From Figure 6, it is evident that center masking consistently outperforms block masking and random masking in terms of accuracy. Center masking shows a steady increase in accuracy, reaching approximately 0.26 by epoch 20. In contrast, block masking and random masking exhibit lower accuracy values, with random masking performing the worst.

The results highlight the effectiveness of different masking strategies in pretraining the Vision Mamba model with MAE. The SSIM values suggest that block masking is more effective in reconstructing images with high structural integrity, as indicated by higher SSIM scores. However, when evaluating the model's accuracy on the ImageNet-1k classification task, center masking proves to be the superior strategy.

The superior performance of center masking in terms of accuracy can be attributed to its ability to focus the model's learning on the most informative parts of the image, typically located in the center. This targeted approach likely helps the model capture more relevant features, leading to better generalization and higher classification accuracy.

On the other hand, the lower accuracy observed with block masking and random masking indicates that these strategies may not be as effective in guiding the model to learn discriminative features essential for classification tasks. Random masking, in particular, shows the poorest performance, suggesting that the dispersed nature of the missing patches may hinder the model's ability to learn coherent and meaningful representations.

6 CONCLUSION

This paper explored integrating Masked Auto-encoders (MAEs) within the Vision Mamba model to enhance image representation learning, evaluating random, block, and center masking strategies on the ImageNet-1k dataset. Block masking achieved the highest SSIM values, performing better compared to center and random masking in image reconstruction. Center masking delivered the best classification accuracy, reaching about 0.26 by epoch 20. Random masking performed the worst in both metrics. These findings highlight the importance of choosing appropriate masking strategies for balancing reconstruction quality and classification performance. Future work should address scalability issues in Vision Mamba and explore other pretraining methods and optimizations for MAEs.

REFERENCES

[1] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594.

[2] Abdelhafid Berroukham, Khalid Housni, and Mohammed Lahraichi. 2023. Vision transformers: a review of architecture, applications, and future directions. *2023 7th IEEE Congress on Information Science and Technology (CiSt)*, 205–210. https://doi.org/10.1109/CiSt56084.2023.10410015.

[3] Shuoxi Zhang, Hanpeng Liu, Stephen Lin, and Kun He. 2024. You only need less attention each stage in vision transformers. *The IEEE/CVF Conference on Computer Vision and Pattern Recognition 2024*, (May 2024). https://www.microsoft.com/en-us/research/publication/you-only-need-less-attention-each-stage-in-vision-transformers/.

[4] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. 2024. Vision mamba: efficient visual representation learning with bidirectional state space model. https://arxiv.org/abs/2401.09417.

[5] Rui Xu, Shu Yang, Yihui Wang, Bo Du, and Hao Chen. 2024. A survey on vision mamba: models, applications and challenges. https://arxiv.org/abs/2404.18861.

[6] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2021. Masked autoencoders are scalable vision learners. https://arxiv.org/abs/2111.06377.

[7] Olga Russakovsky et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115, 3, 211–252. https://doi.org/10.1007/s11263-015-0816-y.

[8] Yuming Li, Jue Wang, Tengfei Xing, Tianlu Liu, Chengjun Li, and Kuifeng Su. 2017. Tad16k: an enhanced benchmark for autonomous driving. *2017 IEEE International Conference on Image Processing (ICIP)*, 2344–2348. https://doi.org/10.1109/ICIP.2017.8296701.

[9] Ruifeng Zheng, Ying Zhong, Senxiang Yan, Hongcheng Sun, Haibin Shen, and Kejie Huang. 2023. Msvrl: self-supervised multiscale visual representation learning via cross-level consistency for medical image segmentation. *IEEE Transactions on Medical Imaging*, 42, 1, 91–102. https://doi.org/10.1109/TMI.2022.3204551.

[10] Kavan Singh Sikand, Sadegh Rabiee, Adam Uccello, Xuesu Xiao, Garrett Warnell, and Joydeep Biswas. 2022. Visual representation learning for preference-aware path planning. *2022 International Conference on Robotics and Automation (ICRA)*, 11303–11309. https://doi.org/10.1109/ICRA46639.2022.9811828.

[11] Yubiao Yue and Zhenzhang Li. 2024. Medmamba: vision mamba for medical image classification. https://arxiv.org/abs/2403.03849.

[12] Zhaohu Xing, Tian Ye, Yijun Yang, Guang Liu, and Lei Zhu. 2024. Segmamba: long-range sequential modeling mamba for 3d medical image segmentation. https://arxiv.org/abs/2401.13560.

[13] Dingkang Liang, Xin Zhou, Wei Xu, Xingkui Zhu, Zhikang Zou, Xiaoqing Ye, Xiao Tan, and Xiang Bai. 2024. Pointmamba: a simple state space model for point cloud analysis. https://arxiv.org/abs/2402.10739.

[14] Haoxing Chen et al. 2024. Demamba: ai-generated video detection on million-scale genvideo benchmark. https://arxiv.org/abs/2405.19707.

[15] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. 2022. Beit: bert pre-training of image transformers. https://arxiv.org/abs/2106.08254.

[16] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. 2022. Simmim: a simple framework for masked image modeling. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9643–9653. https://doi.org/10.1109/CVPR52688.2022.00943.

[17] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. 2022. Masked feature prediction for self-supervised visual pre-training. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 14648–14658. https://doi.org/10.1109/CVPR52688.2022.01426.

[18] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. 2020. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 33, 22243–22255. https://proceedings.neurips.cc/paper_files/paper/2020/file/fcbc95ccdd551da181207c0c1400c655-Paper.pdf.

[19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (June 2020).

[20] Jean-Bastien Grill et al. 2020. Bootstrap your own latent - a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33, 21271–21284. https://proceedings.neurips.cc/paper_files/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf.

[21] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. 2020. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33, 9912–9924. https://proceedings.neurips.cc/paper_files/paper/2020/file/70feb62b69f16e0238f741fab228fec2-Paper.pdf.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 11, 2278–2324. https://doi.org/10.1109/5.726791.

[23] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. https://arxiv.org/abs/1607.06450.

[24] Dan Hendrycks and Kevin Gimpel. 2023. Gaussian error linear units (gelus). https://arxiv.org/abs/1606.08415.

[25] Alexey Dosovitskiy et al. 2021. An image is worth 16x16 words: transformers for image recognition at scale. https://arxiv.org/abs/2010.11929.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. https://arxiv.org/abs/1706.03762.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90.

[28] Diederik P. Kingma and Jimmy Ba. 2015. Adam: a method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6980.

[29] Aryan Jadon, Avinash Patil, and Shruti Jadon. 2022. A comprehensive survey of regression based loss functions for time series forecasting. https://arxiv.org/abs/2211.02989.

[30] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13, 4, 600–612. https://doi.org/10.1109/TIP.2003.819861.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

[32] Peilin Zhou, You-Liang Huang, Yueqi Xie, Jingqi Gao, Shoujin Wang, Jae Boum Kim, and Sunghun Kim. 2024. Is contrastive learning necessary? a study of data augmentation vs contrastive learning in sequential recommendation. *Proceedings of the ACM on Web Conference 2024*, 3854–3863. https://doi.org/10.1145/3589334.3645661.

7 APPENDIX

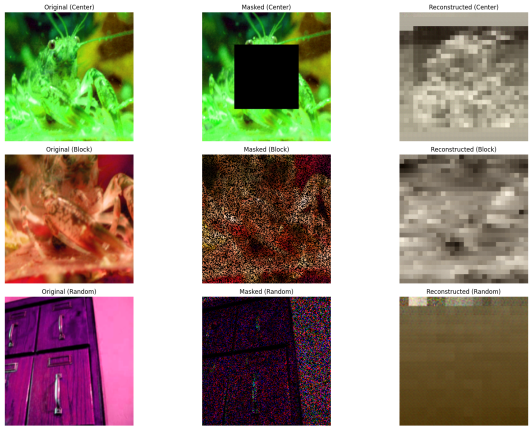

Fig. 7. Epoch 5 of pre-training

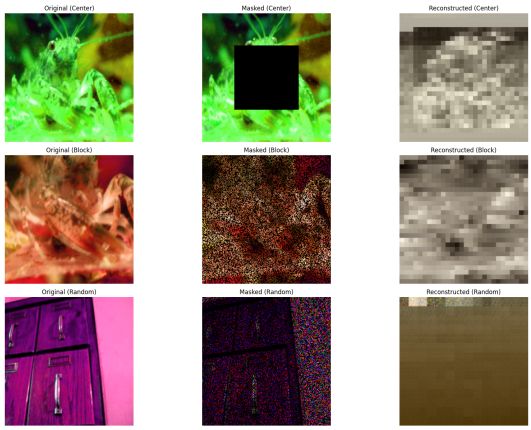

Fig. 8. Epoch 10 of pre-training

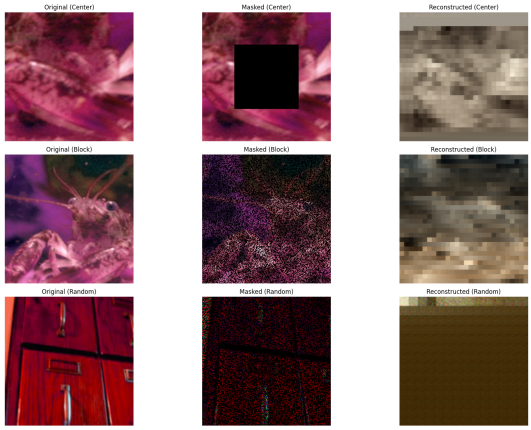

Fig. 9. Epoch 15 of pre-training

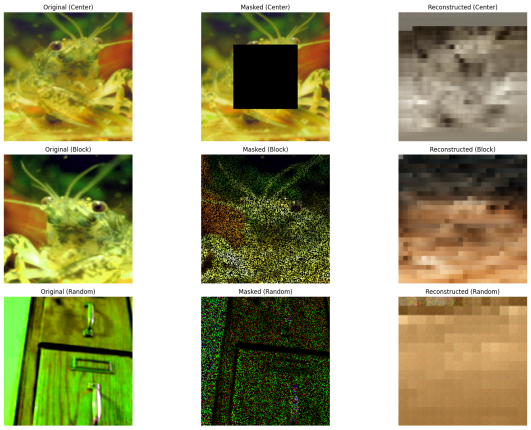

Fig. 10. Epoch 20 of pre-training