ON THE ROLE OF DISCRETE TOKENIZATION IN VISUAL REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In the realm of self-supervised learning (SSL), masked image modeling (MIM) has gained popularity alongside contrastive learning methods. MIM involves reconstructing masked regions of input images using unmasked portions. A notable subset of MIM methodologies employs discrete visual tokens as reconstruction target. This study explores the role of discrete visual tokens in MIM, with the aim of decoding their potential benefits and inherent constraints. Building upon the connection between MIM and contrastive learning, we provide comprehensive explanations on how discrete tokenization affects generalization performance of MIM. Furthermore, we introduce a novel metric designed to quantify the proficiency of discrete visual tokens in the MIM framework. Inspired by this metric, we contribute an accessible tokenizer design and demonstrate its superior performance across various benchmark datasets and ViT backbones.

1 INTRODUCTION

Self-Supervised Learning (SSL) has recently emerged as a promising paradigm for learning data representations without access to labels. Besides the popular contrastive learning methods (He et al., 2020; Chen et al., 2020; Chen & He, 2020; Grill et al., 2020), there is a growing interest in masked image modeling (MIM) techniques. MIM requires masking parts of input images and subsequently attempting their reconstruction using the remaining unmasked parts. Notable examples like MAE (He et al., 2022), BEiT (Bao et al., 2022), PeCo (Dong et al., 2022), MaskFeat (Wei et al., 2022), and MAGE (Li et al., 2022) have demonstrated state-of-the-art performance on various downstream tasks.

Within the MIM domain, various reconstruction targets exist. For instance, MAE (He et al., 2022) employs the raw pixel values of the unmasked parts as its reconstruction target and MaskFeat (Wei et al., 2022) leverages features from other pretrained models as its construction target, whereas some alternative methods adopt visual tokenizers to generate the reconstruction target (Bao et al., 2022; Dong et al., 2022). Such tokenizers convert image patches into predefined discrete tokens. As an illustration, BEiT (Bao et al., 2022) deploys visual tokens generated by a discrete tokenizer known as dVAE (Vahdat et al., 2018). PeCo (Dong et al., 2022) refines the token quality by introducing a tokenizer enriched with perceptual loss. Nevertheless, we notice that different tokenization schemes may bring quite different performance. For example, as Table 1 suggests, PeCo with perceptual tokenizer and MaskFeat with HOG targets outperform MAE, whereas BEiT with dVAE and VQGAN tokenizer underperform MAE using raw pixel. These observations lead us to explore the following natural question:

What is the role of tokenization in MIM? How does it affect downstream performance?

To answer these questions, we leverage the graph perspective of MAE (Zhang et al., 2022) to dissect the influence of different discrete tokenization schemes on downstream generalization. In particular, we find that properly designed discrete visual tokens can improve the masking-induced connectivity between intra-class samples, while misspecified tokenizers may cause confusion between inter-class samples. We provide a comprehensive explanation of this impact on final performance. This insight can also guide the selection of proper tokenizers. Specifically, we design a metric named token-class alignment similarity (TCAS) by measuring the discrepancy in distribution between true patch labels and the discrete token labels assigned by the tokenizer, which can be used to directly compare the quality of different tokenizers without training. Meanwhile, the principle also inspires us to design an easy-to-implement tokenizer and demonstrate its efficacy through enhanced performance on the

Table 1: Linear probing and fine-tuning accuracies (%) of several MIM methods using different tokenizers pretrained on ImageNet100 for 200 epochs. The architecture is chosen as ViT-small following Touvron et al. (2021)). The comparison baseline is the identity function.

Tokenizer	Linear Probing Acc.	Fine-tuning Acc.
Identity function (MAE)	45.94	81.78
dVAE (Vahdat et al., 2018)	41.24 (-4.70)	80.21 (-1.57)
VQGAN (Esser et al., 2021)	44.25 (-1.69)	80.88 (-0.90)
Perceptual tokenizer (Dong et al., 2022)	50.29 ($+5.35$)	83.13 ($+1.45$)
Maskfeat (HOG targets) (Wei et al., 2022)	48.52 ($+2.58$)	82.91 ($+1.13$)

linear probing task across different benchmark datasets (e.g., ImageNet-100, and ImageNet-1K) and different ViT backbones.

We summarize our contributions as follows:

- We notice that discrete visual tokens play an important role in MIM methods, and establish the first theoretical analysis on the influence of different discrete tokenization schemes.
- Based on the mask graph perspective of MIM, we provide a detailed theoretical discussion of how discrete tokenization influences downstream performance, and further verify these understandings on synthetic data.
- Based on the understandings above, we propose 1) a new metric to directly evaluate a given tokenizer, which is shown to align well with its performance; 2) a new discrete tokenizer that is not only much simpler but also enjoys better performance on benchmark datasets.

We hope that this work could inspire more interest in designing better visual tokenization techniques, which also matter for building multi-modal models using both visual and natural languages.

2 Related Works

Masked Image Modeling (MIM). MIM involves learning from a portion of masked input signals by predicting these signals using the unmasked part. Different masked prediction objectives have been proposed for MIM. For instance, MAE (He et al., 2022), and its variants like U-MAE (Zhang et al., 2022) and MixMAE (Liu et al., 2023a), directly predict the raw pixels within the masked part. Alternatively, BEiT (Bao et al., 2022) proposes to use discrete tokens from dVAE (Vahdat et al., 2018) as the reconstruction target. PeCo (Dong et al., 2022) refines the quality of visual tokens by employing an enhanced tokenizer that integrates perceptual loss, yielding improved downstream performance and reconstruction precision. MaskFeat (Wei et al., 2022) explores diverse options for reconstruction targets, such as features from pretrained models and HOG features. DBot (Liu et al., 2023b) can also use representation from another pretrained model as prediction target, but updating the target per certain epochs with current model representation. Notably, the current designs of discrete visual tokens in these methods do not possess theoretical guarantees. In this paper, we provide a theoretical understanding of the working mechanism of discrete visual tokens in MIM, especially regarding their impact on downstream generalization.

Understanding MIM. Despite the remarkable success of MIM methods, their theoretical understandings remain rarely explored. Cao et al. (2022) focus on the attention mechanism of MAE through an integral kernel perspective. Zhang et al. (2022) build a connection between MAE and contrastive learning, highlighting the importance of masking. While these theoretical approaches offer meaningful insights into MAE, they sidestep an examination of the influence of discrete visual tokens on MIM. Therefore, our work intends to explore the specific role and impact of discrete visual tokens within the MIM framework.

3 The Role of Discrete Visual Tokens in MIM

3.1 Preliminary

We begin by detailing the mathematical formulation of MIM. Given a natural image, we first reshape it into n patches, denoted as $\bar{x} \in \mathbb{R}^{n \times s}$ with s representing the patch size. Subsequently, we employ

a random binary mask m drawn from the set $\{0,1\}^n$ to create two complementary views of $\bar{x}$:

$$x_1 = \bar{x}[m] \in \mathbb{R}^{n_1 \times s}, \quad x_2 = \bar{x}[1-m] \in \mathbb{R}^{n_2 \times s}, \tag{1}$$

where n_1 and n_2 are integers satisfying $n = n_1 + n_2$. We denote this random masking process as drawing x_1 and x_2 from the joint distribution $\mathcal{M}(x_1, x_2|\bar{x})$ (respective marginal distributions are represented as $\mathcal{M}(x_1|\bar{x})$ and $\mathcal{M}(x_2|\bar{x})$). Denote the set of all unmasked views as $\mathcal{X}_1 = \{x_1\}$ and the set of all masked views as $\mathcal{X}_2 = \{x_2\}$, where the two sets are assumed to be finite, i.e., $|\mathcal{X}_1| = N_1$ and $|\mathcal{X}_2| = N_2$. The loss function for MIM can be formulated as:

$$\mathcal{L}(h) = \mathbb{E}_{\bar{x}} \mathbb{E}_{x_1, x_2|\bar{x}} L(h(x_1), t(x_2)) \tag{2}$$

Here, the to be learned model network h maps the input x_1 to reconstruct the target $t(x_2)$. The function t transforms the unmasked image patches into tokens, which constitutes the primary research focus of our study. For instance, in MAE (He et al., 2022), t serves as an identity function. While BEiT (Bao et al., 2022) employs a dVAE (Vahdat et al., 2018) tokenizer for t, converting image patches into discrete tokens. The loss function L can adopt a mean square loss form for continuous targets $t(x_2)$ (He et al., 2022; Xie et al., 2022) or a cross-entropy loss form for discrete $t(x_2)$ (Bao et al., 2022; Dong et al., 2022). In this paper, in order to simplify the analysis, we assume that L takes the form of mean square loss and our primary focus revolves around distinguishing different MIM methods based on their prediction target, specifically the selection of t.

3.2 Formalizing the Influence of Tokenization

Building upon the graph-based perspective of MIM in Zhang et al. (2022), we delve into the effect of discrete tokenization on downstream generalization. The foundational premise of Zhang et al. (2022) was to interlink $\mathcal{X}_1$ and $\mathcal{X}_2$ through a mask graph. This framework enabled them to utilize the 2-hop connectivity within the mask graph, thereby formulating an augmentation graph to examine the relationships between element pairs in the input space $\mathcal{X}_1$. Subsequently, they derived a bound for the downstream classification error, contingent upon the characteristics of the augmentation graph. Our objective is to trace this analytical trajectory, specifically focusing on discerning the effects of discrete tokenization on each facet of this graph-based methodology.

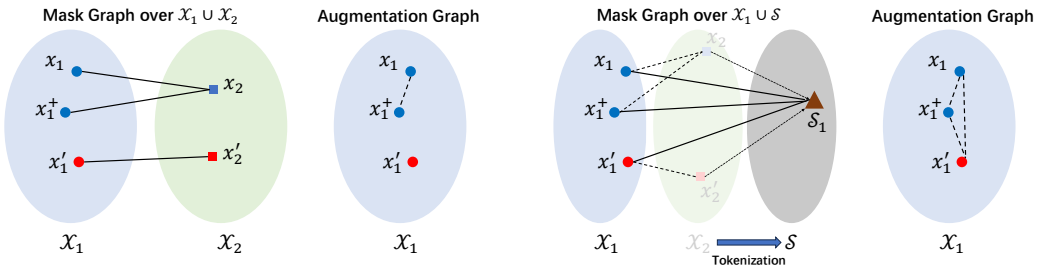
Tokenization Induces Equivalence Class in Target Space. In Equation 2, the function t represents tokenization, which, given its intricate nature, can be analytically challenging. Therefore, it is crucial to identify the core essence of tokenization to simplify the analysis. Notably, the loss function $\mathcal{L}(h)$ aims to align x_1 and its complementary tokenized view $t(x_2)$ through the reconstruction task. Considering pairs (x_1, x_2) sampled from $\mathcal{M}(\cdot, \cdot|\bar{x})$ and (x_1^+, x_2^+) from $\mathcal{M}(\cdot, \cdot|\bar{x}^+)$, an intriguing observation emerges: even if $x_2 \neq x_2^+$, x_1 and x_1^+ can still share the same reconstruction target as long as $t(x_2) = t(x_2^+)$. This observation leads us to focus on the equivalence relation denoted by $\sim$ within $\mathcal{X}_2$. We define $x_2 \sim x_2^+$ if and only if $t(x_2) = t(x_2^+)$. Accordingly, we denote $\mathcal{S} = \mathcal{X}_2/\sim = \{\mathcal{S}_1, \dots \mathcal{S}_l\}$, where each $\mathcal{S}_i$ represents an equivalence class consisting of multiple x_2 elements sharing the same discrete tokens. By doing so, we can treat the target view space as $\mathcal{S}$ because there will be no distinction between x_2 and x_2^+ that belong to the same $\mathcal{S}_i$ due to tokenization. This formalizes discrete tokenization using equivalence classes and obviates the need to delve into the specific properties of t. Therefore, we can equate discrete tokenization with a specific $\mathcal{S}$. Building on this, the joint probability of x_1 and $\mathcal{S}_i$ can be conceptualized as the summation of all joint probabilities of x_1 and each x_2 contained in $\mathcal{S}_i$:

$$\mathcal{M}(x_1, \mathcal{S}_i|\bar{x}) = \sum_{x_2 \in \mathcal{S}_i} \mathcal{M}(x_1, x_2|\bar{x}), \tag{3}$$

where the validity of this definition is further elaborated in Appendix A. It is worth noting that MAE refers to an extreme case where each $\mathcal{S}_l$ becomes a single-element set.

Mask Graph with Tokenized Targets. The original mask graph proposed by Zhang et al. (2022) is defined over the joint set of $\mathcal{X}_1 \cup \mathcal{X}_2$ to describe the joint probability of pairs from $\mathcal{X}_1 \times \mathcal{X}_2$. We now consider the mask graph with tokenized targets $\mathcal{G}_M$, which is defined over the joint set $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{S}$:

- Node: Each view $x_1 \in \mathcal{X}_1$ and each set of views $\mathcal{S}_i \in \mathcal{S}$
- Edge: Edges solely exist between $x_1 \in \mathcal{X}_1$ and $\mathcal{S}_i \in \mathcal{S}$, which essentially results in a bipartite graph structure. The edge weight between x_1 and $\mathcal{S}_i$ is defined as their joint probability $\mathcal{M}(x_1, \mathcal{S}_i) = \mathbb{E}_{\bar{x}} \mathcal{M}(x_1, \mathcal{S}_i|\bar{x})$. Therefore, an edge with non-zero weight connects x_1 and $\mathcal{S}_i$ if and only if there exists $x_2 \in \mathcal{S}_i$ such that x_1 and x_2 are complementary views generated by masking.



(a) Graphs of MAE

(b) Graphs of MIM with discrete tokenization

Figure 1: An illustration of how the discrete tokenization affects the mask graph and the corresponding augmentation graph. x_2 and x_2' share the same discrete token, enabling a connection between x_1 and x_1' through x_2 and x_2', whereas such a connection is not possible in MAE.

Figure 1 visually demonstrates the construction of the mask graph with discrete tokenization, where we can observe three pairs of complementary views: (x_1, x_2), (x_1^+, x_2), and (x_1', x_2'). Notably, since both x_2 and x_2' belong to $\mathcal{S}_1$, this results in the formation of an edge originating from either x_1, x_2, or x_1' connecting to $\mathcal{S}_1$. Intuitively, tokenization aggregates all the weights between x_1 and $x_2 \in \mathcal{S}_i$ into the weight between x_1 and $\mathcal{S}_i$. This aggregation subsequently influences the construction of the augmentation graph which will be illustrated in the following part.

Augmentation Graph Influenced by Tokenization. According to Zhang et al. (2022), MAE generates implicit connections among different input samples through 2-hop connectivity in the mask graph, as it enforces a pair of 2-hop neighbor x_1 and x_1^+ to reconstruct the same output x_2. This treatment transforms the 2-hop input neighbors into positive pairs implicitly aligned as in contrastive learning, thereby forming the concept of an augmentation graph to model relationships among all input samples in $\mathcal{X}_1$. However, with discrete tokenization, this mechanism is modified, now requiring a 2-hop neighbor pair x_1 and x_1^+ to match the same tokenized target $\mathcal{S}_i$. In Figure 1, there is an illustration of how the discrete tokenization affects the mask graph $\mathcal{G}_M$ and the corresponding augmentation graph $\mathcal{G}_A$. Consequently, the augmentation graph influenced by discrete tokenization can be formulated as:

- Node: Each view $x_1 \in \mathcal{X}_1$ is represented as a node in $\mathcal{G}_A$.
- Edge: For any two views $x_1, x_1^+ \in \mathcal{X}_1$, we define their edge weight as the probability of having the same target view. Formally, this edge weight is computed as $\mathcal{A}(x_1, x_1^+) = \mathbb{E}_{\mathcal{S}_i} \mathcal{M}(x_1|\mathcal{S}_i)\mathcal{M}(x_1^+|\mathcal{S}_i)$.

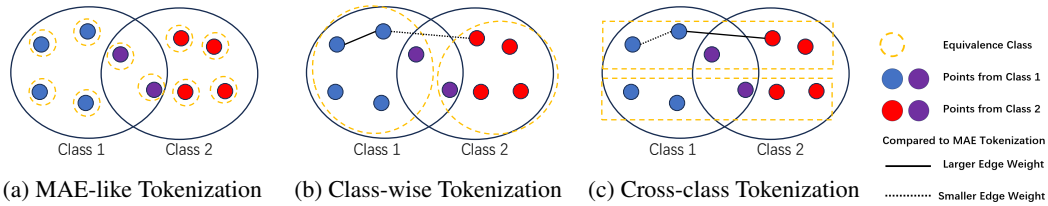
Influence on Downstream Generalization. Denote the adjacent matrix of this graph as A and its normalized version as $\bar{A}$, where $\bar{A} = D^{-1/2}AD^{-1/2}$ and D is a diagonal matrix with $D(x_i, x_i) = \sum_{x_i^+} A(x_i, x_i^+)$. Denote the mask-induced error as $\alpha = \mathbb{E}_{x_1, x_1^+} \mathbb{1}[y(x_1) \neq y(x_1^+)]$ where y is the label function. Zhang et al. (2022) provide a downstream classification error bound as $c_1 \sum_i \lambda_i^2 + c_2\alpha$, where λ_i is the eigenvalue of $\bar{A}$. Therefore, the matrix A plays an important role in our exploration of the downstream classification error bound. Initially, we will investigate how discrete tokenization impacts the values of $A(x_1, x_1^+)$ and subsequently discuss how this influence affects the downstream classification error bound.

3.3 Verification on Synthetic Data

In this part, we further design a toy model to rigorously justify the analysis above. In particular, this model enables us to directly compute the adjacency matrix A and assess the downstream classification error bound across different tokenization scenarios. By doing so, we can develop an intuitive comprehension of how tokenization impacts these numerical values.

Setup. Here is the setup of our toy model:

- **Data Space.** We have two classes, each containing n points representing image patches. These point sets are denoted as P_1 and P_2. There are m overlapping points between the



(a) MAE-like Tokenization (b) Class-wise Tokenization (c) Cross-class Tokenization

Figure 2: Visual illustration of the three tokenization approaches in the toy model. Each orange bounding box represents an equivalence class, whose elements share the same discrete token. **Class-wise tokenization** exhibits a higher intra-class connectivity and lower inter-class connectivity compared to MAE-like tokenization. Consequently, it boasts a lower downstream error bound. In contrast, **cross-class tokenization** leads to lower intra-class connectivity and higher inter-class connectivity, resulting in a significantly larger downstream error bound.

two classes, meaning that $|P_1 \cap P_2| = m$. Therefore, there are a total of $2n - m$ points ($|P_1 \cup P_2| = 2n - m$). Assuming $t = m/n \ll 1$, we define the data distribution such that, when drawing a datum, we randomly select one class, denoted as P_i, and uniformly sample an ordered pair (x_1, x_2) from $P_i \times P_i$.

- **MIM Task and Discrete Tokenization.** In this simulated MIM task, we set x_1 as the unmasked view and x_2 as the masked view. Suppose that the equivalence class induced by the discrete tokenization is $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_l\}$. For the i-th equivalence class $\mathcal{S}_i$, it comprises $n_{i,1}$ elements from P_1/P_2, $n_{i,2}$ elements from P_2/P_1 and $n_{i,3}$ elements from $P_1 \cap P_2$. Therefore, we have the following conditions:

$$\sum_{i=1}^{c} n_{i,1} = \sum_{i=1}^{c} n_{i,2} = n - m, \ \sum_{i=1}^{c} n_{i,3} = m. \tag{4}$$

Tokenizers. We study on three kinds of tokenization: MAE-like tokenization $\mathcal{S}^{\text{MAE}}$ (which essentially implies no tokenization), class-wise tokenization $\mathcal{S}^{\text{class}}$ and cross-class tokenization $\mathcal{S}^{\text{cross}}$. See Figure 2 for visual illustrations of the three tokenization approaches. By calculating the weight edge and downstream error bound in each scenario, we compare $\mathcal{S}^{\text{class}}$ and $\mathcal{S}^{\text{cross}}$ with the baseline $\mathcal{S}^{\text{MAE}}$. By doing this, we are able to explore how tokenization influences the augmentation graph and the downstream error bound as follows (refer to Appendix B for detailed calculations):

- **MAE-like $\mathcal{S}^{\text{MAE}}$.** In this scenario, $\mathcal{S}^{\text{MAE}} = \{\mathcal{S}_1, \ldots, \mathcal{S}_{2n-m}\}$, where each $\mathcal{S}_i$ is a single-element set similar to MAE. In this case, the edge weight between intra-class pairs $w_{\text{intra}}^{\text{MAE}}$, the edge weight between inter-class pairs $w_{\text{inter}}^{\text{MAE}}$, the downstream error bound B^{MAE} should be

$$w_{\text{intra}}^{\text{MAE}} = \frac{2n - m}{4n^3}, \ w_{\text{inter}}^{\text{MAE}} = \frac{m}{4n^3}, \ B^{\text{MAE}} = c(2 - \frac{15t}{4} + O(t^2)). \tag{5}$$

  These numerical results serve as the baselines for the other two tokenization methods.

- **Class-wise $\mathcal{S}^{\text{class}}$.** In this scenario, $\mathcal{S}^{\text{class}} = \{\mathcal{S}_1, \mathcal{S}_2\}$. The two equivalence classes divide the entire point space evenly by class, with $n_{1,1} = n_{2,2} = n - m$, $n_{1,2} = n_{2,2} = n - m$, $n_{1,3} = n_{2,3} = m/2$. In this case, the edge weight between intra-class pairs $w_{\text{intra}}^{\text{class}}$, the edge weight between inter-class pairs $w_{\text{inter}}^{\text{class}}$, the downstream error bound B^{class} should be

$$w_{\text{intra}}^{\text{class}} = \frac{(n - \frac{m}{2})^2 + (\frac{m}{2})^2}{2n^4}, \ w_{\text{inter}}^{\text{class}} = \frac{m(n - \frac{m}{2})}{4n^4}, \ B^{\text{class}} = c(2 - \frac{9t}{2} + O(t^2)). \tag{6}$$

  In comparison to MAE-like tokenization, class-wise tokenization enhances intra-class edge weights while diminishing inter-class edge weights. As suggested by HaoChen et al. (2021), this variation makes the feature space more distinct and separable by class, ultimately leading to improved downstream performance. This assertion aligns with the results of the downstream error bound calculations, where class-wise tokenization exhibits a lower downstream error bound compared to MAE-like tokenization.

- **Cross-class $\mathcal{S}^{\text{cross}}$.** In this scenario, $n_{i,1} = n_{i,2} = (n - m)/l$ and $n_{i,3} = m/l$ which means the l equivalence classes split the three sets of points equally. In this case, the edge

weight between intra-class pairs $w_{\text{intra}}^{\text{cross}}$, the edge weight between inter-class pairs $w_{\text{inter}}^{\text{cross}}$, the downstream error bound B^{cross} should be

$$w_{\text{intra}}^{\text{cross}} = \frac{1}{4n^2},\ w_{\text{inter}}^{\text{cross}} = \frac{1}{4n^2},\ B^{\text{cross}} = c(\frac{7}{2} - \frac{27t}{4} + O(t^2)). \tag{7}$$

In contrast to class-wise tokenization, cross-class tokenization diminishes intra-class edge weights and elevates inter-class edge weights, which, in turn, has a detrimental effect on downstream performance. This observation is further evidenced by the significantly larger downstream error bound B^{cross} compared to that of MAE-like tokenization B^{MAE}.

Summary. Our findings reveal that appropriately designed discrete visual tokens have the potential to enhance the connectivity among intra-class samples induced by masking, leading to a better downstream performance. However, poorly chosen tokenization methods can lead to confusion among inter-class samples, resulting in a considerably larger downstream error bound that may have adverse effects on overall downstream performance.

Moreover, we seek to uncover the characteristics of well-designed visual tokens. Given the contrast properties exhibited by $\mathcal{S}^{\text{class}}$ and $\mathcal{S}^{\text{cross}}$, we delve deeper into their specific compositions. A notable distinction emerges: each equivalence class in $\mathcal{S}^{\text{class}}$ consists of points from the same true class, while each equivalence class in $\mathcal{S}^{\text{cross}}$ comprises an equal distribution of points from both classes. This observation raises a fundamental question: Does having tokenization classes that align with the true classes lead to better downstream performance? The answer is Yes, as demonstrated by the following theorem:

Theorem 1. *Assuming that $\mathcal{M}(x_1|x_2) > 0$ occurs only if $y(x_1) = y(x_2)$, and let $\sim_y$ denote the equivalence relation on $\mathcal{X}_2$ where $x_2 \sim_y x_2^+$ if and only if $y(x_2) = y(x_2^+)$. Then $\mathcal{S}^y = \mathcal{X}_2/\sim_y = \{\mathcal{S}_1^y, \ldots, \mathcal{S}_c^y\}$ minimizes $c_1 \sum_{i=1}^{N_1} \lambda_i^2 + c_2\alpha$.*

Theorem 1 suggests that, under the assumption that two views of an image come from the same class, the optimal discrete tokenizer is simply the label function. This insight serves as valuable inspiration for the design and selection of improved discrete tokenizers, which will be discussed in the next section.

4 Designing and Choosing Better Discrete Tokenizer

In Section 3, we have explored the role of discrete tokens in MIM and studied their effect on downstream generalization. Therefore, these theoretical studies provide some insights on designing and choosing better discrete tokenizers. In this section, we first design a metric to evaluate the discrete tokenizer in MIM. Then, based on the metric, we design a simple but effective discrete tokenizer utilizing K-means.

4.1 Token-class Alignment Similarity: Mearsuring the Discrepancy between Tokenizer Class and True Class.

Theorem 1 suggests that, under the assumption that two views of an image come from the same class, the optimal discrete tokenizer is simply the label function. Consequently, we can design a metric to evaluate a discrete tokenizer by measuring the discrepancy between the equivalence class induced by this discrete tokenizer and that induced by the true label. In practice, the set of the equivalence class $\mathcal{S}$ could be exponentially large, since there are potentially C^{n_2} distinct discrete representations across all masked views $\mathcal{X}_2$. where C is the size of the codebook. The vast set is intractable for us to manage when designing a practical metric. Therefore, in the following part, we consider a bag-of-word model (Harris, 1954) for patch representations to make the analysis more tractable. Utilizing a bag-of-word model allows us to treat the target view $t(x_2)$ not as a single entity but as a set of individual tokens. Since patch representations are often local, this simplification is meaningful and make the analysis more manageable. With the results in Theorem 1, we anticipate that a discrete tokenizer with a lower value of this metric will yield improved downstream performance. The metric is constructed as detailed below:

1. **Computing Token-class Co-occurrence.** To quantify the discrepancy between the tokenizer class and true class, we should first establish the token-class co-occurrence matrix. Let $R \in \mathbb{R}^{l_1 \times l_2}$ be a matrix, where l_1 represents the size of the codebook of the discrete tokenizer, and l_2 represents the number of true classes. Considering all the patches in the

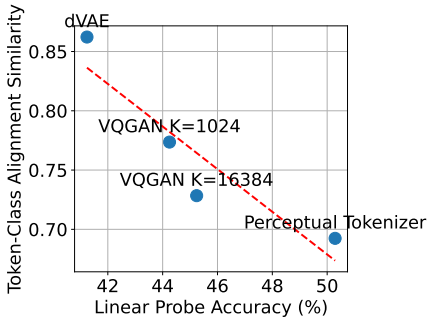


Figure 3: Correlation between the designed metric and the linear probing accuracy.

Table 2: Comparison of token-class alignment similarity (TCAS) between our proposed tokenizer and previous methods.

Method	TCAS (↓)
dVAE (Vahdat et al., 2018)	0.86
VQGAN K=1024 (Esser et al., 2021)	0.77
VQGAN K=16384 (Esser et al., 2021)	0.73
Perceptual Tokenizer (Dong et al., 2022)	0.69
KMIM PIXEL	0.56
KMIM DINO	**0.21**

dataset, we define $R(i,j)$ as the count of patches belonging to the i-th class in the codebook and the j-th true class. The matrix $\bar{R}$, which is the normalized version of R along its rows, is given by $\bar{R}(i,j) = R(i,j)/\sum_{j'} R(i,j')$. Here, the i-th row of the matrix $\bar{R}$, denoted as $\bar{R}(i)$, indicates the distribution of true classes within the patches that belong to the i-th discrete class.

2. **Measuring Tokenization-induced Class Confusion.** Ideally, if we want to minimize the discrepancy between the tokenizer class and the true class, we aim for $\bar{R}(i)$ to be close to a one-hot vector, and $\bar{R}(i)$ and $\bar{R}(i')$ with $i \neq i'$ should be dissimilar. In other words, this pushes $C = \bar{R}\bar{R}^\top$ close to the identity matrix $I_{l_1 \times l_1}$. Accordingly, we formulate the metric as follows:

$$M = \lambda_1 \sum_i (1 - C_{ii})^2 + \lambda_2 \sum_{i \neq i'} C_{i,i'}^2 = \lambda_1 \sum_{i=1}^{l_1} (1 - \|\bar{R}(i)\|_2)^2 + \lambda_2 \sum_{i \neq i'} (\bar{R}(i)\bar{R}(i')^\top)^2, \quad (8)$$

which is similar to the Barlow Twin loss (Zbontar et al., 2021). To ensure that the metric is not influenced by the codebook size l_1, we set $\lambda_1 = 1/l_1$ and $\lambda_2 = 1/l_1^2$. In this metric, as $\bar{R}(i)$ approaches a one-hot vector, the first term in M decreases. Simultaneously, if $\bar{R}(i)$ and $\bar{R}(i')$ with $i \neq i'$ become orthogonal, the second term reaches the minimal. Thus, this metric effectively measures the discrepancy between the tokenizer class and the true class.

We designate this metric as the Token-Class Alignment Similarity (TCAS). To validate the soundness of this metric, we assess the correlation between the metric and linear downstream performance through experiments utilizing various tokenizers, including dVAE (Vahdat et al., 2018), VQGAN(Esser et al., 2021) and Perceptual tokenizer (Dong et al., 2022). The results of these experiments are visually represented in Figure 3, which indicates a noticeable correlation between the metric and the performance, thus affirming the validity of this metric. This validates the utility of this metric in evaluating tokenizers **without the need for pretraining** and provides guidance for the design of improved tokenization methods.

4.2 K-MIM: Leveraging K-means for Generating Discrete Tokens

In the previous sections, we have demonstrated through toy models and theory that an effective tokenizer should exhibit label correlation. However, the practicality of self-supervised learning often lacks access to labeled information. Therefore, we need a feasible alternative. Notably, our previous analysis indicates that the specific labels correlated with discrete classes are less important. The critical factor is that patches with the same labels should be aggregated within a single discrete class. In light of this, we propose a clustering-based approach to form the discrete tokenizer:

- **Step 1: Extracting Patch Features.** Given a dataset, we input each image into a pretrained model to extract features for individual patches within the image.
- **Step 2: Clustering the Features.** We apply a clustering algorithm to the aggregated features, yielding a set of K clustering centers. These centers constitute the codebook for the discrete tokenizer.

Table 3: Linear probing accuracy (LP Acc.) and fine-tuning accuracy (FT Acc.) of pretrained models by various MIM methods with different ViT backbones on ImageNet-100 and ImageNet-1K. ViT-S, ViT-B and ViT-L are abbreviations of ViT-Small, ViT-Base and ViT-Large, repectively. **Bold** indicates the best performance within the same setting.

Dataset	Model	Method	Extra Model	LP Acc.	FT Acc.
ImageNet-100	ViT-S	MAE (He et al., 2022)	No	45.9	81.8
		MaskFeat HOG ver.(Wei et al., 2022)	No	49.1	82.8
		K-MIM PIXEL	No	**52.7**	**86.4**
		BEiT (Bao et al., 2022)	dVAE	43.2	81.0
		PeCo (Dong et al., 2022)	VQVAE	53.2	83.6
		MaskFeat continuous ver. (Wei et al., 2022)	DINO	53.1	84.3
		K-MIM DINO	DINO	**59.7**	84.7
	ViT-B	MAE (He et al., 2022)	No	61.2	86.9
		BEiT (Bao et al., 2022)	dVAE	55.8	86.1
		K-MIM PIXEL	No	**63.1**	**88.8**
	ViT-L	MAE (He et al., 2022)	No	64.4	87.3
		K-MIM PIXEL	No	**67.2**	**88.9**
ImageNet-1K	ViT-B	MAE (He et al., 2022)	No	55.4	82.9
		K-MIM PIXEL	No	**61.2**	**83.4**
		BEiT (Bao et al., 2022)	dVAE	53.2	82.7
		MaskFeat continuous ver. (Wei et al., 2022)	DINO	63.2	83.7
		K-MIM DINO	DINO	**67.4**	**83.8**

- **Step 3: Assigning Nearest Neighbor.** To determine the discrete tokens for an image, we employ the same pretrained model as in step 1 to extract features for each patch in the image. Subsequently, we identify the nearest neighbor among the codebook features and designate it as the discrete token for the respective patch.

In practice, for the pretrained model in step 1, two choices are implementable: 1) utilizing a pretrained unsupervised ViT network (e.g., DINO (Caron et al., 2021)), or 2) employing the identity function, which treats the raw pixel values of the patches as features. For the clustering method in step 2, we opt for the straightforward K-Means clustering. We refer to the MIM method incorporating this designed discrete tokenizer as K-MIM (K-Means Masked Image Modeling). Depending on the choice made in the first step, we have two variants: K-MIM PIXEL, which employs the identity function, and K-MIM DINO, which relies on the DINO pretrained network. As shown in Table 2, our proposed methods indeed have a lower token-class alignment similarity. The experiments presented in Section 5 provide empirical evidence of the effectiveness of our proposed methods.

5 Experiments

In this section, we first present the main empirical results of our proposed K-MIM methods on different real-world datasets with different backbones. Then we conduct a series of ablation experiments to discuss the selection of hyperparameters in our methods.

5.1 Evaluation on Benchmark Datasets

To evaluate the effectiveness of the proposed K-MIM method, extensive experiments are conducted on ImageNet-100 (Deng et al., 2009) and ImageNet-1K (Deng et al., 2009).

Setup. We mainly follow the basic setup of MAE (He et al., 2022): for the encoder, we adopt several variants of ViT (Dosovitskiy et al., 2020), including ViT-Small, ViT-Base, and ViT-Large. For the decoder, we follow the setting of He et al. (2022). The mask ratio is set to 0.75. On both datasets, we pretrain the model for 200 epochs with batch size 4096 and weight decay 0.05. We conduct both linear evaluation and non-linear fine-tuning on the pretrained encoder. For linear evaluation, we train a linear classifier on the frozen pretrained encoder. As for non-linear fine-tuning, we train both the pretrained encoder and the linear classifier with the soft target cross entropy loss (Peterson et al., 2019). For the K-Means algorithm, we use K-Means++ initialization (Arthur & Vassilvitskii, 2007). We train K-Means for 100 epochs on ImageNet-100 and 10 epochs on ImageNet-1K.

Effectiveness of the Proposed K-MIM method. In Table 3, we present a performance comparison of different MIM methods across two benchmark datasets. Notably, our proposed methods exhibit

Table 4: Ablation study on the selection of clustering number K. Linear probing accuracy / Fine-tuning accuracy in each box.

	K-MIM PIXEL	K-MIM DINO
$K = 50$	**52.7/86.4**	58.2/84.3
$K = 100$	50.1/83.7	**59.7**/84.7
$K = 1000$	50.6/84.4	56.9/**85.1**

Table 5: Experiments exploring different K-Means training epochs. Linear probing accuracy / Fine-tuning accuracy in each box.

	K-MIM PIXEL	K-MIM DINO
1 epoch	47.2/82.1	53.5/84.3
10 epochs	52.5/85.9	57.4/84.4
100 epochs	**52.7/86.4**	**59.7/84.7**

a significant performance advantage over all baseline methods. Specifically, on the ImageNet-100 dataset, with minimal extra time constructing discrete tokens through K-Means on the pixel space (5 hours on K-Means and 36 hours on pretraining), our K-MIM PIXEL utilizing the ViT-S backbone outperform MAE by an impressive 6.8% in linear probing and 4.6% in fine-tuning accuracy. The improvement still holds when using larger backbones and the larger ImageNet-1K dataset.

Furthermore, when using the discrete tokens generated by pretrained DINO features as the reconstruction target (K-MIM DINO) instead of the pretrained DINO features themselves (MaskFeat continuous version), we achieve a performance boost of 6.6% in linear probing and 0.4% in fine-tuning accuracy. These improvements provide direct evidence of the effectiveness of our proposed K-means pre-processing method.

5.2 Ablation Study

In this section, we conduct an ablation study to investigate the impact of two key factors in our proposed K-MIM method: the choice of clustering number K and the training epochs of the K-Means algorithm. These experiments are conducted on the ImageNet-100 dataset using the ViT-small architecture.

Clustering Number K. We vary the clustering number K for the K-Means algorithm while fixing the training epoch to be 100. The results of these ablation experiments are presented in Table 4. As observed in the table, different choices of clustering numbers have varying effects on the downstream performance. In the case of K-MIM PIXEL, we achieve the highest linear probing accuracy of 52.7% and fine-tuning accuracy of 86.4% when using $K = 50$. Conversely, for K-MIM DINO, the best performance is achieved when setting $K = 100$, yielding linear probing accuracy of 59.7% and fine-tuning accuracy of 84.7%. Notably, using a very large $K = 1000$ did not necessarily lead to improved performance. Therefore, it is advisable to select a moderate clustering number in practice.

K-Means Training Epochs. We conducted experiments to explore the influence of varying the training epochs for the K-Means algorithm within the K-MIM framework. The results of these experiments are presented in Table 5. Across all scenarios, the highest accuracies are consistently achieved when training K-Means for 100 epochs, demonstrating the importance of a well-trained K-Means model in enhancing K-MIM performance. It is noteworthy that the K-Means algorithm is computationally efficient. For instance, in the case of K-MIM PIXEL trained on ImageNet-100 with 200 epochs, while the pretraining stage takes 36 hours, training 1 epoch of K-Means with 100 cluster centers requires only 0.05 hours. Furthermore, it's worth highlighting that even with just 1 epoch of K-Means pretraining, K-MIM PIXEL outperformed MAE (47.2/82.1 vs. 45.9/81.8), and K-MIM DINO outperformed the continuous version of MaskFeat (53.5/84.3 vs. 53.1/84.3). These results underscore the effectiveness and efficiency of our proposed methods.

6 Conclusion

In this paper, we have drawn attention to the crucial role that discrete visual tokens play in MIM methods and introduced the first comprehensive theoretical analysis of the impact of different discrete tokenization approaches. Specifically, by employing the mask graph perspective of MIM, we have presented an detailed theoretical exploration of how discrete tokenization influences downstream performance. Our analysis has underscored the significance of aligning tokenizers with true labels for improved performance. Building upon these insights, we have introduced a novel metric called Token-Class Alignment Similarity (TCAS) to directly assess the quality of a given tokenizer, which has demonstrated strong alignment with its performance. Leveraging this metric, we have further proposed a straightforward yet effective discrete tokenizer, which exhibits superior performance across various benchmark datasets.

REFERENCES

David Arthur and Sergei Vassilvitskii. K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *ICLR*, 2022.

Shuhao Cao, Peng Xu, and David A Clifton. How to understand masked autoencoders. *arXiv preprint arXiv:2202.03670*, 2022.

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

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *arXiv preprint arXiv:2011.10566*, 2020.

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

Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, Nenghai Yu, and Baining Guo. Peco: Perceptual codebook for bert pre-training of vision transformers, 2022.

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, 2020.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

Jean-Bastien Grill, Florian Strub, Florent Altché, C. Tallec, Pierre H. Richemond, Elena Buchatskaya, C. Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020.

Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for selfsupervised deep learning with spectral contrastive loss. *NeurIPS*, 2021.

Zellig S. Harris. Distributional structure. *WORD*, 10(2-3):146–162, 1954. doi: 10.1080/00437956.1954.11659520.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Tianhong Li, Huiwen Chang, Shlok Kumar Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. Mage: Masked generative encoder to unify representation learning and image synthesis, 2022.

Jihao Liu, Xin Huang, Jinliang Zheng, Yu Liu, and Hongsheng Li. Mixmae: Mixed and masked autoencoder for efficient pretraining of hierarchical vision transformers, 2023a.

Xingbin Liu, Jinghao Zhou, Tao Kong, Xianming Lin, and Rongrong Ji. Exploring target representations for masked autoencoders, 2023b.

Joshua C Peterson, Ruairidh M Battleday, Thomas L Griffiths, and Olga Russakovsky. Human uncertainty makes classification more robust. In *ICCV*, 2019.

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

Arash Vahdat, Evgeny Andriyash, and William G. Macready. Dvae: Discrete variational autoencoders with relaxed boltzmann priors, 2018.

Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *CVPR*, 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*, 2022.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021.

Qi Zhang, Yifei Wang, and Yisen Wang. How mask matters: Towards theoretical understandings of masked autoencoders. In *NeurIPS*, 2022.

A Elaboration on the Validity of the Definition in Equation 3

It suffices to verify that $\sum_{\mathcal{S}_i \in \mathcal{S}} \mathcal{M}(x_1, \mathcal{S}_i|\bar{x}) = \mathcal{M}(x_1|\bar{x})$. We have

$$\begin{aligned}\sum_{\mathcal{S}_i \in \mathcal{S}} \mathcal{M}(x_1, \mathcal{S}_i|\bar{x}) &= \sum_{\mathcal{S}_i \in \mathcal{S}} \sum_{x_2 \in \mathcal{S}_i} \mathcal{M}(x_1, x_2|\bar{x}) \\ &= \sum_{x_2 \in \mathcal{X}_2} \mathcal{M}(x_1, x_2|\bar{x}) \\ &= \mathcal{M}(x_1|\bar{x}).\end{aligned} \tag{9}$$

B Detailed Calculation in Section 3.3

We denote $w_{x_1,x_2} = \mathcal{M}(x_1, x_2)$ as the joint probability of one complementary pairs and denote $w_{x_1} = \mathcal{M}(x_1)$ as the marginal probability of one view.

Consider the joint probability of one complementary pair x_1, x_2: When the two views both come from the same overlapped region, the joint probability $w_{x_1,x_2} = 1/n^2$. Otherwise, the joint probability is $1/2n^2$.

For the masked view x_2 or unmasked view x_1 in the non-overlapped region, the marginal probability is $1/(2n)$. For masked views and unmasked views in the overlapped region, the marginal probability is $1/n$.

The normalized augmentation graph is

$$\bar{A}_{\text{discrete}}(x_1, x_1^+) = \sum_{x_2, x_2^+} \frac{w_{x_1,x_2} w_{x_1^+,x_2^+} \cdot w_{x_2,x_2^+}}{w_{x_2} w_{x_2^+} \sqrt{w_{x_1} w_{x_1^+}}} = \sum_{\mathcal{D}_i} \sum_{x_2, x_2^+ \in \mathcal{D}_i} \frac{w_{x_1,x_2} w_{x_1^+,x_2^+}}{\sqrt{w_{x_1} w_{x_1^+}} p(\mathcal{D}_i)}. \tag{10}$$

We calculate the joint probability of a positive pair under each situation. Each item denotes where the two unmasked views of the pair come from:

- $(P_1/P_2, P_1/P_2)$

$$\begin{aligned}p_{11} &= 2n \cdot \sum_{i=1}^{l} \frac{2n}{n_{i,1} + n_{i,2} + 2n_{i,3}} \cdot \left((n_{i,1} + n_{i,3})^2 \cdot \frac{1}{2n^2} \cdot \frac{1}{2n^2} \right) \\ &= \sum_{i=1}^{l} \frac{1}{n_{i,1} + n_{i,2} + 2n_{i,3}} \cdot \left((n_{i,1} + n_{i,3})^2 \cdot \frac{1}{n^2} \right)\end{aligned} \tag{11}$$

- $(P_2/P_1, P_2/P_1)$

$$\begin{aligned}p_{22} &= 2n \cdot \sum_{i=1}^{l} \frac{2n}{n_{i,1} + n_{i,2} + 2n_{i,3}} \cdot \left((n_{i,2} + n_{i,3})^2 \cdot \frac{1}{2n^2} \cdot \frac{1}{2n^2} \right) \\ &= \sum_{i=1}^{l} \frac{1}{n_{i,1} + n_{i,2} + 2n_{i,3}} \cdot \left((n_{i,2} + n_{i,3})^2 \cdot \frac{1}{n^2} \right)\end{aligned} \tag{12}$$

- $(P_1/P_2, P_2/P_1)$

$$\begin{aligned}p_{12} &= 2n \cdot \sum_{i=1}^{l} \frac{2n}{n_{i,1} + n_{i,2} + 2n_{i,3}} \cdot \left((n_{i,1} + n_{i,3})(n_{i,2} + n_{i,3}) \cdot \frac{1}{2n^2} \cdot \frac{1}{2n^2} \right) \\ &= \sum_{i=1}^{l} \frac{1}{n_{i,1} + n_{i,2} + 2n_{i,3}} \cdot \left(n_{i,3}^2 \cdot \frac{1}{n^2} \right)\end{aligned} \tag{13}$$

- $(P_1/P_2, P_1 \cap P_2)$

$$p_{13} = \sqrt{2}n \cdot \sum_{i=1}^{l} \frac{2n}{n_{i,1}+n_{i,2}+2n_{i,3}} \cdot \left((n_{i,1}+n_{i,3})n_{i,1} \cdot \frac{1}{2n^2} \cdot \frac{1}{2n^2} + (n_{i,1}+n_{i,3})n_{i,3} \cdot \frac{1}{n^2} \cdot \frac{1}{2n^2} \right)$$
$$= \sum_{i=1}^{l} \frac{1}{n_{i,1}+n_{i,2}+2n_{i,3}} \cdot \left((n_{i,1}+n_{i,3})(n_{i,1}+2n_{i,3}) \cdot \frac{\sqrt{2}}{2n^2} \right) \tag{14}$$

- $(P_2/P_1, P_1 \cap P_2)$

$$p_{23} = \sqrt{2}n \cdot \sum_{i=1}^{l} \frac{2n}{n_{i,1}+n_{i,2}+2n_{i,3}} \cdot \left((n_{i,2}+n_{i,3})n_{i,2} \cdot \frac{1}{2n^2} \cdot \frac{1}{2n^2} + (n_{i,2}+n_{i,3})n_{i,3} \cdot \frac{1}{n^2} \cdot \frac{1}{2n^2} \right)$$
$$= \sum_{i=1}^{l} \frac{1}{n_{i,1}+n_{i,2}+2n_{i,3}} \cdot \left((n_{i,2}+n_{i,3})(n_{i,2}+2n_{i,3}) \cdot \frac{\sqrt{2}}{2n^2} \right) \tag{15}$$

- $(P_1 \cap P_2, P_1 \cap P_2)$

$$p_{33} = n \cdot \sum_{i=1}^{l} \frac{2n}{n_{i,1}+n_{i,2}+2n_{i,3}} \cdot \left((n_{i,1}+n_{1,2}+2n_{i,3})^2 \cdot \frac{1}{2n^2} \cdot \frac{1}{2n^2} \right)$$
$$= \sum_{i=1}^{l} \frac{1}{n_{i,1}+n_{i,2}+2n_{i,3}} \cdot \left((n_{i,1}+n_{1,2}+2n_{i,3})^2 \cdot \frac{1}{2n^2} \right) \tag{16}$$
$$= \frac{1}{n}$$

The normalized adjacent matrix is:

$$A = \begin{bmatrix} p_{11} & p_{11} & p_{11} & \cdots & p_{12} & p_{12} & p_{12} & \cdots & p_{13} \\ p_{11} & p_{11} & p_{11} & \cdots & p_{12} & p_{12} & p_{12} & \cdots & p_{13} \\ p_{11} & p_{11} & p_{11} & \cdots & p_{12} & p_{12} & p_{12} & \cdots & p_{13} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{12} & p_{12} & p_{12} & \cdots & p_{22} & p_{22} & p_{22} & \cdots & p_{23} \\ p_{12} & p_{12} & p_{12} & \cdots & p_{22} & p_{22} & p_{22} & \cdots & p_{23} \\ p_{12} & p_{12} & p_{12} & \cdots & p_{22} & p_{22} & p_{22} & \cdots & p_{23} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{13} & p_{13} & p_{13} & \cdots & p_{23} & p_{23} & p_{23} & \cdots & p_{33} \end{bmatrix} \tag{17}$$

The sum of square of the eigenvalues is equal to the sum of square of the elements in the adjacent matrix:

$$\sum_i \lambda_i^2 = (n-m)^2(p_{11}^2 + p_{22}^2 + 2p_{12}^2) + (n-m)m(p_{13}^2 + p_{23}^2) + m^2 p_{33}^2$$
$$= \frac{(n-m)^2}{n^4} \left((\sum_{i=1}^{l} \frac{(n_{i,1}+n_{i,3})^2}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 + (\sum_{i=1}^{l} \frac{(n_{i,2}+n_{i,3})^2}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 + 2(\sum_{i=1}^{l} \frac{(n_{i,1}+n_{i,3})(n_{i,2}+n_{i,3})}{n_{i,1}+n_{i,2}+2n_{i,3}})^2) \right)$$
$$+ \frac{(n-m)m}{2n^4} \left((\sum_{i=1}^{l} \frac{(n_{i,1}+n_{i,3})(n_{i,1}+2n_{i,3})}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 + (\sum_{i=1}^{l} \frac{(n_{i,2}+n_{i,3})(n_{i,2}+2n_{i,3})}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 \right) + \frac{m^2}{n^2} \tag{18}$$

The labeling error should be

$$\alpha = \mathbb{E}_{x_1, x_1^+} 1_{y(x_1) \neq y(x_1^+)} = 2(n-m)^2 p_{12}/(2n) = \frac{(n-m)^2}{n^3} \sum_{i=1}^{l} \frac{(n_{i,1}+n_{i,3})(n_{i,2}+n_{i,3})}{n_{i,1}+n_{i,2}+2n_{i,3}} \tag{19}$$

We consider the downstream error bound derived in (Zhang et al., 2022), which is

$$
\begin{aligned}
B &= c_1 \sum_i \lambda_i^2 + c_2 \alpha \\
&= c \left(\frac{(n-m)^2}{n^4} \left((\sum_{i=1}^{l} \frac{(n_{i,1}+n_{i,3})^2}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 + (\sum_{i=1}^{l} \frac{(n_{i,2}+n_{i,3})^2}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 + 2(\sum_{i=1}^{l} \frac{(n_{i,1}+n_{i,3})(n_{i,2}+n_{i,3})}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 \right) \right. \\
&\quad \left. + \frac{(n-m)m}{2n^4} \left((\sum_{i=1}^{l} \frac{(n_{i,1}+n_{i,3})(n_{i,1}+2n_{i,3})}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 + (\sum_{i=1}^{l} \frac{(n_{i,2}+n_{i,3})(n_{i,2}+2n_{i,3})}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 \right) + \frac{m^2}{n^2} + c_3 \alpha \right) \\
&= c \left(\frac{(n-m)^2}{n^4} \left((\sum_{i=1}^{l} \frac{(n_{i,1}+n_{i,3})^2}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 + (\sum_{i=1}^{l} \frac{(n_{i,2}+n_{i,3})^2}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 + 2(\sum_{i=1}^{l} \frac{(n_{i,1}+n_{i,3})(n_{i,2}+n_{i,3})}{n_{i,1}+n_{i,2}+2n_{i,3}} + \frac{c_3 n}{4})^2 \right) \right. \\
&\quad \left. + \frac{(n-m)m}{2n^4} \left((\sum_{i=1}^{l} \frac{(n_{i,1}+n_{i,3})(n_{i,1}+2n_{i,3})}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 + (\sum_{i=1}^{l} \frac{(n_{i,2}+n_{i,3})(n_{i,2}+2n_{i,3})}{n_{i,1}+n_{i,2}+2n_{i,3}})^2 \right) + \frac{m^2}{n^2} - C. \right.
\end{aligned} \tag{20}
$$

where $C = c_3^2(n-m)^2/(8n^2)$ and $c_3 = 5/2$. We denote $t = m/n$ and assume $t \ll 1$. Since p_{11}, p_{12} and p_{22} are elements from normalized adjacency matrix, the edge weight of intra-class pairs can be computed as $(p_{11}+p_{22})/4n$ and the edge weight of inter-class pairs can be computed as $p_{12}/4n$.

MAE Tokenization. We have $l = 2n - m$ and only one of $n_{i,1}$, $n_{i,2}$ and $n_{i,3}$ is 1 and the other two are 0 for each $1 \le i \le l$. The bound for MAE tokenization should be

$$
\begin{aligned}
B^{\text{MAE}} &= c(\frac{(n-m)^2}{n^4}(2(n-\frac{m}{2})^2 + 2(\frac{m}{2}+\frac{cn}{4})^2) + \frac{(n-m)m}{2n^4}(n^2+n^2) + \frac{m^2}{n^2} - C) \\
&= c(2 - 5t + 7t^2 - 4t^3 + t^4 + \frac{c}{2}(1-t)^2 t) \\
&= c(2 - \frac{15}{4}t + \frac{9}{2}t^2 - \frac{11}{4}t^3 + t^4)
\end{aligned} \tag{21}
$$

The two types of edge weights are

$$
w_{\text{intra}}^{\text{MAE}} = \frac{2n-m}{4n^3},\ w_{\text{inter}}^{\text{MAE}} = \frac{m}{4n^3} \tag{22}
$$

Class-wise Tokenization. We consider an extreme case where $l = 2$, $n_{1,1} = n - m, n_{1,2} = 0, n_{1,3} = m/2$ and $n_{2,1} = 0, n_{2,2} = n - m, n_{2,3} = m/2$. The bound for this case should be

$$
\begin{aligned}
B^{\text{class}} &= c(2 - 7t + 16t^2 - \frac{39}{2}t^3 + \frac{29}{2}t^4 - \frac{95}{16}t^5 + \frac{15}{16}t^6 + c\frac{(n-m)^2 m(n-\frac{m}{2})}{n^4}) \\
&= c(2 - 7t + 16t^2 - \frac{39}{2}t^3 + \frac{29}{2}t^4 - \frac{95}{16}t^5 + \frac{15}{16}t^6 + \frac{5}{2}(1-t)^2(t-\frac{t^2}{2})) \\
&= c(2 - \frac{9}{2}t + \frac{39}{4}t^2 + O(t^3)).
\end{aligned} \tag{23}
$$

The two types of edge weights are

$$
w_{\text{intra}}^{\text{class}} = \frac{(n-\frac{m}{2})^2 + (\frac{m}{2})^2}{2n^4},\ w_{\text{inter}}^{\text{class}} = \frac{m(n-\frac{m}{2})}{4n^4} \tag{24}
$$

Cross-class Tokenization $n_{i,1} = n_{i,2} = (n-m)/l$ and $n_{i,3} = m/l$. The bound should be

$$
\begin{aligned}
B^{\text{class}} &= c(\frac{(n-m)^2}{n^4}(\frac{n^2}{4} + \frac{n^2}{4} + 2\frac{n^2}{4}) + \frac{(n-m)m}{2n^4}((\frac{n+m}{2})^2 + (\frac{n+m}{2})^2) + \frac{m^2}{)}n^2 + c\frac{(n-m)^2}{2n^2} \\
&= c((1-t)^2 + \frac{1}{4}(1-t)t(t+1)^2 + t^2 + \frac{c}{2}(1-t)^2) \\
&= c(\frac{7}{2} - \frac{27}{4}t + \frac{15}{4}t^2 + O(t^3)).
\end{aligned} \tag{25}
$$

The two types of edge weights are

$$
w_{\text{intra}}^{\text{cross}} = \frac{1}{4n^2},\ w_{\text{inter}}^{\text{cross}} = \frac{1}{4n^2}, \tag{26}
$$

C TOY MODEL WITH MULTIPLE CLASSES

In this section, We extend the discussion in Section 3.3 from two classes to multiple classes. Here are the setting of the toy models.

- **Data Space.** We have s classes, each containing n points representing image patches. These point sets are denoted as P_i, $i = 1, 2, \dots s$. There are m overlapping points between any two classes, meaning that $|P_i \cap P_j| = m$. There does not exist any point belonging to three classes. Therefore, there are a total of $sn - ms(s-1)/2$ points. Assuming $t = m/n \ll 1$, we define the data distribution such that, when drawing a datum, we randomly select one class, denoted as P_i, and uniformly sample an ordered pair (x_1, x_2) from $P_i \times P_i$.
- **MIM Task and Discrete Tokenization.** In this simulated MIM task, we set x_1 as the unmasked view and x_2 as the masked view. Suppose that the equivalence class induced by the discrete tokenization is $\mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_l\}$. For the i-th equivalence class $\mathcal{S}_i$, it comprises $n_{i,a,b}$ elements from $P_a \cap P_b$ where $a \neq b$, $n_{i,a}$ elements from $P_a / \cup_{b \neq a} P_b$. Therefore, we have the following conditions:

$$\sum_{i=1}^{c} n_{i,a} = n - (s-1) * m, \ \sum_{i=1}^{c} n_{i,a,b} = m, \ a \neq b. \tag{27}$$

  Tokenizers. We also study on three kinds of tokenization: MAE-like tokenization $\mathcal{S}^{\text{MAE}}$ (which essentially implies no tokenization), class-wise tokenization $\mathcal{S}^{\text{class}}$ and cross-class tokenization $\mathcal{S}^{\text{cross}}$. By calculating the weight edge and downstream error bound in each scenario, we compare $\mathcal{S}^{\text{class}}$ and $\mathcal{S}^{\text{cross}}$ with the baseline $\mathcal{S}^{\text{MAE}}$. By doing this, we are able to explore how tokenization influences the augmentation graph and the downstream error bound as follows:

  - **MAE-like $\mathcal{S}^{\text{MAE}}$.** In this scenario, $\mathcal{S}^{\text{MAE}} = \{\mathcal{S}_1, \dots, \mathcal{S}_{sn-ms(s-1)/2}\}$, where each $\mathcal{S}_i$ is a single-element set similar to MAE. In this case, the edge weight between intra-class pairs $w_{\text{intra}}^{\text{MAE}}$, the edge weight between inter-class pairs $w_{\text{inter}}^{\text{MAE}}$, the downstream error bound B^{MAE} should be

$$\begin{aligned} w_{\text{intra}}^{\text{MAE}} &= \frac{2n-(s-1)m}{2sn^3}, \ w_{\text{inter}}^{\text{MAE}} = \frac{m}{2sn^3}, \\ B^{\text{MAE}} &= c(s - \frac{15(s-1)t}{4} + O(t^2)). \end{aligned} \tag{28}$$

    These numerical results serve as the baselines for the other two tokenization methods.
  - **Class-wise $\mathcal{S}^{\text{class}}$.** In this scenario, $\mathcal{S}^{\text{class}} = \{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_s\}$. The s equivalence classes divide the entire point space evenly by class, with $n_{s,s,b} = m/2$, $n_{s,s} = n - (s-1)m$. In this case, the edge weight between intra-class pairs $w_{\text{intra}}^{\text{class}}$, the edge weight between inter-class pairs $w_{\text{inter}}^{\text{class}}$, the downstream error bound B^{class} should be

$$\begin{aligned} w_{\text{intra}}^{\text{class}} &= \frac{(n-(s-1)\frac{m}{2})^2 + (s-1)(\frac{m}{2})^2}{2sn^4}, \ w_{\text{inter}}^{\text{class}} = \frac{m(n-(s-1)\frac{m}{2})}{2sn^4}, \\ B^{\text{class}} &= c(s - \frac{9(s-1)t}{2} + O(t^2)). \end{aligned} \tag{29}$$

    In comparison to MAE-like tokenization, class-wise tokenization enhances intra-class edge weights while diminishing inter-class edge weights. This leads to lower downstream error bound, which is consistent with the situation where $s = 2$.
  - **Cross-class $\mathcal{S}^{\text{cross}}$.** In this scenario, $n_{i,a} = (n-(s-1)m)/l$ and $n_{i,a,b} = m/l$, , $a \neq b$ which means the l equivalence classes split each set of points equally. In this case, the edge weight between intra-class pairs $w_{\text{intra}}^{\text{cross}}$, the edge weight between inter-class pairs $w_{\text{inter}}^{\text{cross}}$, the downstream error bound B^{cross} should be

$$w_{\text{intra}}^{\text{cross}} = \frac{1}{2sn^2}, \ w_{\text{inter}}^{\text{cross}} = \frac{1}{2sn^2}, \ B^{\text{cross}} = c(\frac{1}{2} + \frac{3s}{2} + O(t)). \tag{30}$$

    In contrast to class-wise tokenization, cross-class tokenization diminishes intra-class edge weights and elevates inter-class edge weights, which, in turn, has a detrimental effect on downstream performance. This observation is further evidenced by the significantly larger downstream error bound B^{cross} compared to that of MAE-like tokenization B^{MAE}.

In conclusion, the conclusion for multiple classes is similar to that for two classes. While class-wise tokenization will obtain larger intra-class connectivity and lower inter-class connectivity, leading to lower downstream error bound and potentially better downstream performance. On the other hand, cross-class tokenization that does not align well with the class, obtain larger inter-class connectivity and lower intra-class connectivity, leading to larger downstream error bound.

D EXPERIMENTS ON LONGER TRAINING

We conduct additional experiments using K-MIM pixel involving 800 epochs of pretraining on ImageNet-1k using ViT-B as the backbone. The comparisons with baseline methods are presented in Table 6. Notably, with only half the number of pretraining epochs, K-MIM pixel reaches the same finetuning accuracy with MAE and ourperforms MAE in the linear probing accuracy. The results indicate that K-MIM pixel outperforms the baseline results in the classification tasks, ensuring the better learning ability of K-MIM.

Table 6: Linear probing accuracy and fine-tuning accuracy of pretrained models with ViT-B on ImageNet-1K. Bold indicates the best performance.

Methods	Epochs	Linear Probing Accuracy	Fine-tuning Accuracy
MAE	1600	68.0	**83.6**
BEiT	800	N/A	83.2
K-MIM pixel	800	**69.4**	**83.6**

E TCAS SCORE OF VARIOUS TOKENIZERS

In this section, we will present and analyze the TCAS scores of various tokenizers and the corresponding performance on ImageNet-100. The results are detailed in Table 7, and we visually organize the outcomes in Figure 4, where we use linear regression to fit the data. The r-value of the fitting is 0.85, demonstrating a strong correlation between TCAS score and linear probing accuracy. Specifically, the tokenizer's linear probing performance tends to improve as the TCAS metric decreases. Therefore, TCAS can serve as a valuable guidance for assessing the quality of a tokenizer.

Table 7: TCAS scores and performances on classification tasks of various discrete tokenizers on ImageNet-100 with 200 epochs pretraining.

K	Method	TCAS	Linear Probing Acc.	Fine-tuning Acc.
10	K-MIM PIXEL	0.69	42.7	81.9
25	K-MIM PIXEL	0.60	48.3	84.6
50	K-MIM PIXEL (KMEANS 1 epoch)	0.66	47.2	82.1
50	K-MIM PIXEL (KMEANS 10 epochs)	0.56	52.5	85.9
50	K-MIM PIXEL (KMEANS 100 epochs)	0.56	52.7	86.4
50	K-MIM DINO	0.21	58.2	84.3
100	K-MIM PIXEL	0.52	50.1	83.7
100	K-MIM MAE	0.34	52.3	83.2
100	K-MIM DINO	0.15	59.7	84.7
100	K-MIM DeiT	0.09	64.8	83.1
200	K-MIM PIXEL	0.49	50.8	84.8
1000	K-MIM PIXEL	0.39	50.6	84.4
1000	K-MIM DINO	0.12	56.9	85.1
1024	VQGAN	0.77	44.3	80.9
8192	dVAE	0.86	41.2	80.2
8192	Perceptual Tokenizer	0.69	50.3	83.1

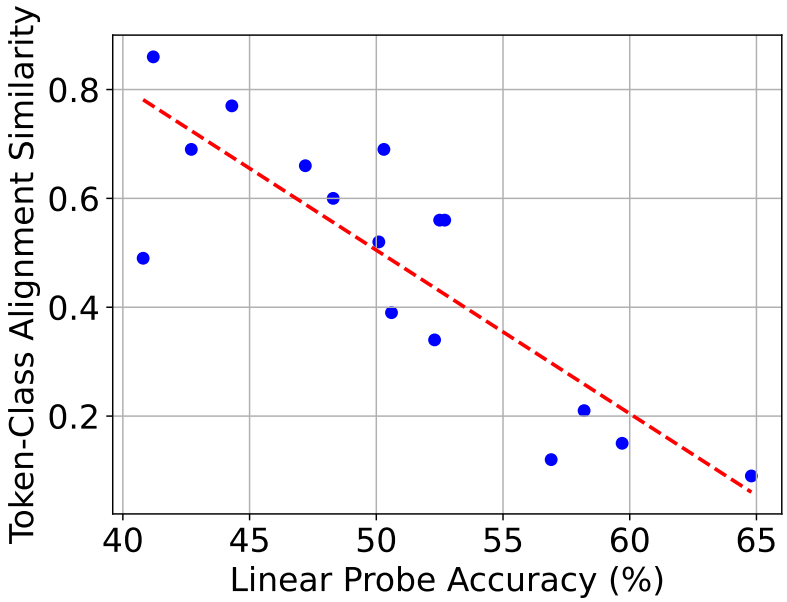


Figure 4: The relation between TCAS score and linear probing accuracy on ImageNet-100.