SeiT: Storage-Efficient Vision Training with Tokens Using 1% of Pixel Storage

Song Park* Sanghyuk Chun* Byeongho Heo Wonjae Kim Sangdoo Yun
* Equal contribution

NAVER AI Lab

Abstract

We need billion-scale images to achieve more generalizable and ground-breaking vision models, as well as massive dataset storage to ship the images (e.g., the LAION-4B dataset needs 240TB storage space). However, it has become challenging to deal with unlimited dataset storage with limited storage infrastructure. A number of storage-efficient training methods have been proposed to tackle the problem, but they are rarely scalable or suffer from severe damage to performance. In this paper, we propose a storage-efficient training strategy for vision classifiers for large-scale datasets (e.g., ImageNet) that only uses 1024 tokens per instance without using the raw level pixels; our token storage only needs <1% of the original JPEG-compressed raw pixels. We also propose token augmentations and a Stem-adaptor module to make our approach able to use the same architecture as pixel-based approaches with only minimal modifications on the stem layer and the carefully tuned optimization settings. Our experimental results on ImageNet-1k show that our method significantly outperforms other storage-efficient training methods with a large gap. We further show the effectiveness of our method in other practical scenarios, storage-efficient pre-training, and continual learning. Code is available at https://github.com/naver-ai/seit

1. Introduction

We need billion-scale data points for more generalizable and ground-breaking vision models, *e.g.*, 400M image-text pairs [45], 1.8B image-text pairs [29], or 3.6B weakly-annotated images [40, 56]. However, designing and operating a high-performance but fault-tolerant generic distributed dataset is a very expensive and challenging problem [71]. This problem has become more challenging for vision datasets compared to language datasets. For example, training GPT-2 with 8M documents only need 40GB of storage [46], while the larger GPT-3 is trained with 410B tokens with 570GB of storage [11]. On the other hand, storing images requires significantly more storage space than storing language. For example, the ImageNet-21k dataset [51] with 11M images requires a 1.4TB storage size, 2.5 times larger than GPT-3 storage despite containing fewer data points. Larger-scale datasets for large-scale pre-training require even more massive storage, *e.g.*, 240TB for 5B images [53]. Consequently, storage remains a major bottleneck in scaling up vision models compared to language models.

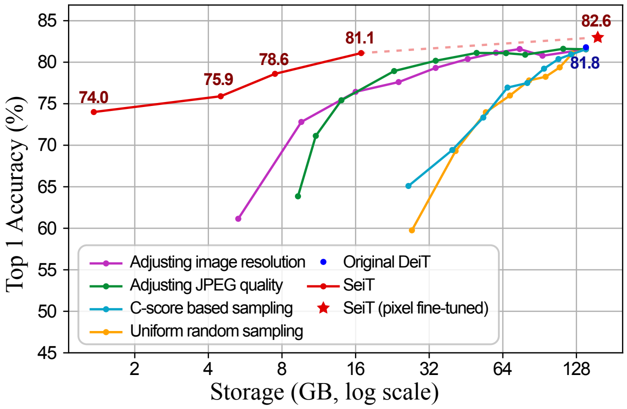


Figure 1. **Training data storage vs. ImageNet 1k Accuracy.** Comparisons on ImageNet-1k [51] using ViT-B/16 backbone [20] are shown. Our SeiT (red lines) significantly outperforms other storage-efficient methods with the same storage size, achieving 74.0% top-1 acc with only 1.36GB and 78.6% with 7.49GB. Note that the original pixel-based image storage requires 140GB of storage to achieve 81.8% top-1 accuracy. Details are in Table B.2.

Why do images require a large storage size than text? This is because while the nature of language is discrete, images are continuous in nature. Additionally, while the quality of a text is independent of document size, the quality of an image directly affects the storage size; better quality images require larger storage sizes. Although a lossy JPEG compression algorithm can reduce the storage size, as witnessed by Rombach *et al.*, still "most bits of a digital image corresponds to imperceptible details" [49]. Such imperceptible details (*e.g.*, fine-grained details or high-frequency information of images) could be unnecessary for our desired vision classifiers. However, deep vision models are vulnerable to imperceptible high-frequency perturbations [23, 39, 17] or unreasonably local areas [22, 6, 55], implying that deep vision models attend too much to imperceptible details instead of the true property of objects. Therefore, we can expect that we can still achieve a high-performing vision model with the reduced image dataset by removing the imperceptible details.

There are two major directions to storage-efficient vision model training. The first direction aims to reduce the total number of data points by discarding less important samples [42, 43, 30] or synthesizing more "condensed" images than natural images [74, 73]. However, this approach shows a significant performance drop compared to the full dataset (the blue and yellow lines in Fig. 1) or cannot be applied to large-scale datasets due to their high complexity. Also, as the sampled or synthesized images are still normal images, these methods still suffer from an inefficient compression ratio to express imperceptible details. Furthermore, these methods need to compute the importance score or the sample-wise gradient of each sample by learning models with the full dataset. It makes these approaches not applicable to unseen datasets or newly upcoming data streams.

The other approach involves reducing the size of each image while keeping the total number of images. One possible direction is to learn a more efficient compression method [7, 8]. However, the neural compression methods have been mostly studied on extremely small-scale datasets (*e.g.*, 24 images [15] or 100 images [4]), and their generalizability to large-scale datasets is still an open problem. Moreover, the goal of neural compression is to compress an image and recover the original image as perfectly as possible, not to extract the most discriminant features for object recognition tasks. In response to these limitations, to the best of our knowledge, no neural compression method has been used to compress large-scale datasets like ImageNet [51] to train deep vision models.

Due to the difficulty of using neural compression methods in practical scenarios, practitioners have attempted to reduce storage usage by controlling the image quality of each image. For example, the LAION dataset [54, 53] stores each image at 256 × 256 resolution, which takes up only 36% of images in ImageNet, which on average use 469 × 387 resolution. Similarly, we can reduce the overall storage by adjusting the JPEG compression quality. As shown in Fig. 1 (green and purple lines), these approaches work well in practice compared to sampling-based methods. However, these methods have a limited compression ratio; if the compression ratio becomes less than 25%, the performances drop significantly. By adjusting the image resolution with a 4% compression ratio and JPEG quality with a 7% compression ratio, we achieve 63.3% and 67.8% top-1 accuracies, respectively. In contrast, our approach achieves 74.0% top-1 accuracy with only a 1% compression ratio.

All shortcomings of the previous methods originate from the fact that too many imperceptible bits are assigned to store a digital image, which is misaligned with our target task. Our approach overcomes this limitation by storing images as tokens rather than pixels, using a pre-trained vision tokenizer, ViT-VQGAN tokenizer [68]. Introducing **S**torage-**e**fficient V**i**sion **T**raining (**SeiT**), we convert each image to 32 × 32 discrete tokens. The number of possible cases each token can have (the codebook) is 391, which takes only 1.15KB to store each token (assuming that the number of 391 cases can be expressed in 9 bits). It costs only less than 1.5GB for storing 140GB pixel-based storage of ImageNet. We train Vision Transformer (ViT) models on our tokenized images with minimum modifications. First, a 1024-length tokenized image is converted to a $32 \times 32 \times 32$ tensor by using pre-trained 32-dimensional codebook vectors from ViT-VQGAN. Next, we apply random resized crop (RRC) to the tensor to get a $32 \times 28 \times 28$ tensor. Then, to convert the tensor into a form that ViT can handle, we introduce *Stem-Adapter* module that converts the RRC-ed tensor into a tensor of size $768 \times 14 \times 14$, the same as the first layer input of ViT after the stem layer. Because the image-based augmentations are not directly applicable to tokens, we propose simple token-specific augmentations, including *Token-EDA* (inspired from easy data augmentation (EDA) [67] for language), *Emb-Noise* and *Token-CutMix* (inspired from CutMix [69]). In our experiment, we achieve 74.0% top-1 accuracy with 1.36GB token storage, where the full image storage requires 140GB to achieve 81.8% [61].

Our method has several advantages over previous storage-efficient methods. First, as we use a frozen pre-trained tokenizer (only using ImageNet-1k) that only requires forward operations to extract tokens from images, we do not need an additional optimization for compressing a dataset, such as importance score-based sampling [30], image synthesis methods [74, 73], or neural compression [7, 8]. Hence, SeiT is easily applicable to unseen datasets or newly upcoming data streams directly. Second, unlike previous methods that use pre-trained feature extractors (*e.g.*, HOG [19] or Faster-RCNN features [47, 2]), SeiT can use the same architecture as image-based approaches with only minimal modifications on the stem layer, as well as the carefully tuned optimization settings, such as DeiT [61]. This property becomes a huge advantage when using our method as an efficient pre-training method; we can achieve 82.6% top-1 accuracy by fine-tuning the token pre-trained model with images. Moreover, applying an input augmentation for feature extractor-based approaches is not straightforward, limiting their generalizability. Finally, our method shows a significant compression ratio compared to previous methods, with a 1% compression ratio for ImageNet.

We show the effectiveness of our method on three image classification scenarios: (1) storage-efficient ImageNet-1k benchmark (2) storage-efficient large-scale pre-training, and (3) continual learning. The overview of storage-efficient results is shown in Fig. 1: SeiT outperforms comparison methods with a significant gap with the same storage size. Our method achieves 74.0% top-1 accuracy on ImageNet under 1% of the original storage, where comparison methods need 40% (uniform sampling, C-score sampling [30]), 6% (adjusting image resolution), and 8% (adjusting JPEG quality) of the original storage to achieve the similar performance. We also demonstrate that SeiT can be applied to large-scale pre-training for an image-based approach; we pre-train a ViT-B/16 model on the tokenized ImageNet-21k (occupying only 14.1GB) and fine-tune the ViT model on the full-pixel ImageNet-1k. By using slightly more storage (156GB vs. 140GB), our storage-efficient pre-training strategy shows 82.8% top-1 accuracy, whereas the full-pixel ImageNet-1k training shows 81.8%. Finally, we observe that our token-based approach significantly outperforms the image-based counterpart in the continual learning scenario [48] by storing more data samples in the same size of the memory compared to full-pixel images.

Contributions. (1) We compress an image to 1024 discrete tokens using a pre-trained visual tokenizer. By applying a simple lossless compression for the tokens, we achieve only 0.97% storage size compared to images stored in pixels. (2) We propose Stem-Adapter module and augmentation methods for tokens such as Token-RRC, Token-CutMix, Emb-Noise, and Token-EDA in order to enable ViT training with minimal change to the protocol and hyperparameters of existing ViT training. (3) Our storage-efficient training pipeline named **S**torage-**e**fficient V**i**sion **T**raining (**SeiT**) shows great improvements on the low-storage regime. With only 1% storage size, SeiT achieves 74.0% top-1 ImageNet 1k validation accuracy. (4) We additionally show that SeiT can be applied to a storage-efficient pre-training strategy, and continual learning tasks.

2. Related Works

Importance sampling for efficient training. Sampling-based methods [42, 43, 14, 30] aims to idendity a compact, yet representative subset of the training dataset that satisfies the original objectives for efficient model training. This is usually achieved through exploring the early training stage [43], constructing a proxy model [14], or utilizing consistency score (C-score) [30]. However, the empirical performance gap between sampling-based methods and the baseline approach of random selection is insignificant, particularly in large-scale datasets like ImageNet-1k (See Fig. 1). We believe that preserving the diversity of data points in a dataset is crucial, and therefore we endeavor to maintain the number of data points instead of pruning them.

Dataset distillation. Dataset distillation [66] aims to generate a compact dataset by transferring the knowledge of the training dataset into a smaller dataset. Recent works [74, 73, 35, 52, 50] have shown that the synthesized data can be effective in efficient model training, especially in scenarios such as continual learning [48]. However, due to their high complexity, they have not yet demonstrated successful cases in large-scale datasets such as ImageNet-1k. We recommend the survey paper [36] for curious readers.

Neural compression. Image compression algorithms (also known as learned image compressions) have improved with the use of neural network training to minimize a quality loss on lossy compression. The representative learned image compression methods are based on variational auto-encoder [7, 8]. The compressor encodes an image to discrete latent codes and the codes can be decoded into the image with small losses. Recent studies [12, 31] have utilized the self-attention mechanism [64] with heavy CNN architectures to demonstrate superior compression power compared to conventional methods such as JPEG. However, the learned image compression targets high-quality images with complex and detailed contexts, which are distant from ImageNet samples. Thus, it is challenging to apply these methods to compress ImageNet for ViT training.

Learning with frozen pre-extracted features. Using extracted visual features for a model has been widely used in the computer vision field. It shows reasonable performances with a low computational cost compared to pixel-based visual encoders. For example, the Youtube-8M [1] dataset consists of frame features extracted from Inception [57] instead of raw pixels, allowing efficient video model training [41, 10] with frozen frame features. The pre-extracted features have also been widely used for tasks that need higher knowledge than pixel-level understandings. For example, frozen CNN features [37] or bottom-up and top-down (BUTD) [59, 2] features [32] have been a popular choice for vision-and-language models that aim to understand complex fused knowledge between two modalities, *e.g.*, visual question answering [3, 24]. These approaches show slightly worse performances than the end-to-end training from raw inputs without pre-extracted features [33, 45], but show high training efficiency in terms of computations.

However, these methods need feature-specific modules to handle frozen features and specialized optimization techniques rather than standard optimization methods of pixel-based methods. Furthermore, some fundamental augmentations, such as random resized crop (RRC), are not applicable to the frozen features, resulting in inferior generalizability. SeiT has major advantages over these methods where it is the almost same training method for ViT (*e.g.*, DeiT [61]), and yet it can significantly reduce the storage space.

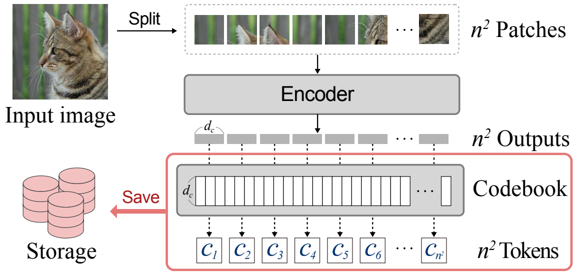


Figure 2. **Tokenization.** The input image is resized to 256 × 256 and then divided into non-overlapping n^2 patches. The patches are fed into the ViT-VQGAN encoder, which produces a sequence of d_c dimensional vectors from the patches. Finally, the tokens are generated by mapping each vector to the nearest code in a pre-trained codebook. We used 32 for both n and d_c in this paper.

3. Token-based Storage-Efficient Training

In this section, we propose **S**torage-**e**fficient **Vi**sion **T**raining (**SeiT**). SeiT aims to learn a high-performing vision classifier at scale (*e.g.*, ImageNet-1k [51]) with a small storage size (*e.g.*, under 1% of the original storage), a minimal change on the training strategy (*e.g.*, highly optimized training strategy such as DeiT [61]), and the minimum sacrifice of accuracies. Our method consists of two parts (1) preparing the compressed token dataset and (2) training a model using the tokens.

3.1. Preparing the token dataset

We extract tokens from the raw images using the ViT-VQGAN tokenizer [68] due to two reasons. First, ViT-VQGAN shows a great reconstruction quality on large-scale datasets, such as ImageNet. Second, we found that the number of valid token indices by ViT-VQGAN for ImageNet is 391, which means each token can be converted into approximately only 9 bits ($2^9 = 512$). We now describe the details of how we efficiently store tokens from raw images.

Fig. 2 shows the overview of the dataset preparation pipeline. We first resize the entire ImageNet dataset to 256 × 256. Then, each resized image is divided into non-overlapping 8 × 8 image patches. Finally, we encode each patch into a 32-dimensional vector and assign a code index by finding the nearest codeword from the pre-trained codebook. Here, we only use 391 codewords from the 8192 original codewords because we found that only 391 codewords are used for the ImageNet training dataset. As a result, each image is converted to 32 × 32 tokens where each token belongs to [0, . . ., 390]. We also store the codebook of ViT-VQGAN (a 32 × 391 vector) to re-use the knowledge of the codebook for better performance.

In theory, as our token indices belong to [0-390], the optimal bit length to store the tokens is $\log_2 391 = 8.61$[1] by the source coding theorem [16]. Therefore, the optimal storage size of an image will be 1.08 kB[2]. However, in practice, we cannot store tokens in 8.61 bits because the commonly used data types use Byte for the minimal unit, *e.g.*, 1 Byte (`uint8`) or 2 Bytes (`uint16`). To compress the required bits per token to less than 2 Bytes, we propose a simple yet efficient encoding for the tokens. First, we assign each token index following the token popularity, *i.e.*, the most frequent token is assigned to index 0, and the least frequent token is assigned to index 390. Then, we break up token indices larger than 255 into two elements as follows:

$$i = \begin{cases} [i] & \text{if } i < 255 \\ [255, i] & \text{if } i \geq 255 \end{cases} \tag{1}$$

We store multiple tokens in a file to reduce the required storage as small as possible. However, because our encoding process makes the length of each token variable, the naive decoding process for our encoding will need $O(n)$ complexity where n is the number of encoded tokens by Eq. (1). We solve the problem by simply storing the start indices of each image. The index storage only requires 9.8 MB for the entire ImageNet training dataset, but it makes the decoding process becomes $O(1)$ and parallelizable. Pseudo-codes for the proposed encoding-decoding are in Appendix A.1.

Our simple encoding strategy reduces 40% of the overall storage size compared to the naive `uint16` data type as shown in Table 1. Here, as the original baseline storage also employs a compression algorithm, such as JPEG (See the first and the second row of Table 1), we also apply a simple compression algorithm, Huffman coding [27]. After applying Huffman coding to our token storage, we achieve nearly optimal storage size per image (1.11 kB vs. 1.08 kB). We empirically observe that the entire decoding process, including Huffman decoding, is almost neglectable.

Format	Encoding	Storage size	Avg. size per image
Pixels	`uint8` (uncompressed)	1471.2 GB	1.14 MB
Pixels	JPEG (baseline)	140.0 GB	109.3 kB
Tokens	`uint16` (uncompressed)	2.50 GB	2.0 kB
Tokens	Ours (8 bits encoding)	1.54 GB	1.26 kB
Tokens	Ours + Huffman coding	**1.36 GB**	**1.11 kB**
Tokens	*Theoretical optimum*	1.32 GB	1.08 kB

Table 1. **Storage size of the ImageNet-1k training dataset for different formats and encodings.** `uint8` and `uint16` denote uncompressed version of each data format. *Theoretical optimum* is estimated by assuming the token population is uniform.

[1] Following the empirical population of the tokens, the "empirical" optimal bit length is 8.54 by computing $H(p) = -\sum p_i \log p_i$. However, in the rest of the paper, we assume the population is uniform for simplicity.

[2] We have 1.08 kB = bits per token (8.61) × token length (1024) / bits per Byte (8). If we follow the actual distribution, it becomes 1.07 kB.

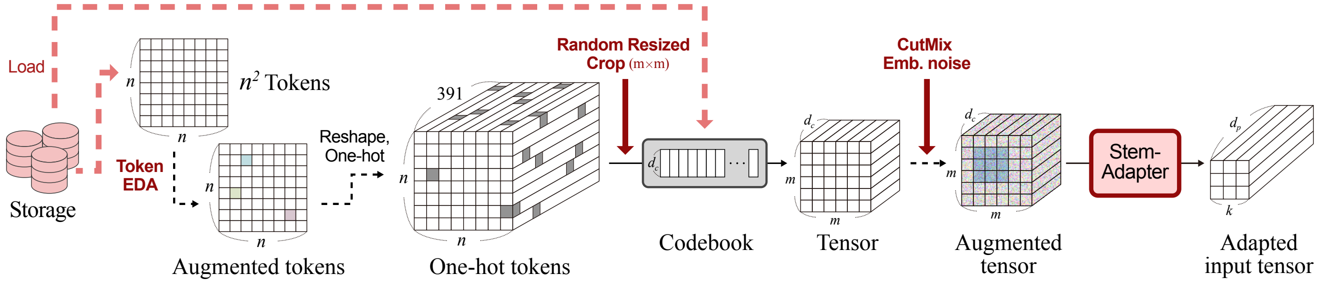


Figure 3. **The data processing pipeline for token data.** For each image, a $n \times n$-shaped token is loaded from the storage. We apply Token-EDA to augment the token and convert it into a one-hot form. Then, we randomly resize crop the one-hot token to $m \times m$ and process it using the pre-trained ViT-VQGAN codebook to transform it into a $d_c \times m \times m$ tensor. We further apply CutMix and Embedding-noise to this tensor and the augmented tensor is then fed into the Stem-Adapter module, which transforms it into a shape of $d_V \times k \times k$, making it suitable for use with ViT models. Our experimental values for the parameters are $n = 32$, $m = 28$, $k = 14$, $d_c = 32$, and $d_V = 768$.

In the remaining part of this paper, we use the compressed version of our token dataset if there is no specification.

3.2. Training classifiers with tokens

Training a classifier with tokenized images is not trivial. For example, an input token has 32 × 32 dimensions, but a conventional image input has 3 × 224 × 224. Furthermore, strong image-level augmentations (*e.g.*, RandAugment [18], Gaussian blur [62]) have become crucial in large-scale vision classifiers, however, these operations cannot be directly applied to the tokens. One possible direction is to decode tokens to pixel-level images during every forward computation. However, this would impose an additional computational load on the network. Instead, we propose simple yet effective token-level augmentations and a simple Stem-Adapter module to train a vision classifier directly on the tokens with minimal modification but small sacrifices.

3.2.1 Token Augmentations

Token-EDA. We utilize the EDA [67], designed for language models, to augment our token data. EDA originally involves four methods: Synonym Replacement (SR), Random Insertion (RI), Random Swap (RS), and Random Deletion (RD). However, we only adopt SR and RS because the others do not maintain the number of tokens, which is not compatible with the ViT training strategy. For SR, we define synonyms of a token as the five tokens that have the closest Euclidean distance in the ViT-VQGAN codebook space. Then, each token is randomly replaced with one of its synonyms with a certain probability p_s during training. For RS, we randomly select two same-sized squares from a 32 × 32 token and swapped the tokens inside them with each other, with a probability p_r. We use 0.25 for p_s and p_r for SeiT.

Token-RRC and Token-CutMix. In addition to EDA, we apply Random Resized Crop (RRC) and CutMix [69] to tokens. For RRC, we adopt a standard ImageNet configuration with a scale (0.08, 1) and an aspect ratio (3/4, 4/3). To enable interpolation, we first convert the original 32 × 32 tokens to one-hot form. Then, apply the random cropping to these one-hot tokens, which are subsequently resized to 28 × 28 using bicubic interpolation. After RRC, the one-hot tokens are converted to a 32 × 28 × 28 tensor using the pre-trained codebook vectors from ViT-VQGAN, where 32 is the size of a pre-trained code vector. Note that tokens that are not in one-hot form due to interpolation are converted to mixed codebooks following their values. CutMix is then applied to these tensors, whereby a patch is randomly selected from one token and replaced with a patch from another token while maintaining the channel dimension.

Adding channel-wise noise. We also developed Emb-Noise, a token augmentation method that mimics color-changing image augmentations, such as color jittering. Inspired by the fact that each channel in an image represents a specific color, we first generate noise of length 32 and add it to each channel of the converted tensor with 32 × 28 × 28 dims, and then apply full-size iid noise, i.e. noise size of 32 × 28 × 28, to the tensor. All of the noise is sampled from a normal distribution. We have empirically demonstrated that this method brings significant performance improvement despite its simplicity. Moreover, we found that adding channel-wise noise to the tokens in ViT-VQGAN, the tokenizer we used, effectively changes the colors of the decoded images, unlike adding Gaussian noise in entire dimensions. Example decoded images by ViT-VQGAN are presented in Appendix A.2.

3.2.2 Stem-Adapter module

As the tokens have a smaller size than images, they cannot be directly used for input of networks. We introduce a Stem-Adapter that converts the augmented tensor into ViT/16 to make minimal modifications on the network. Specifically, the Stem-Adapter module converts the 32 × 28 × 28 preprocessed tokens into 768 × 14 × 14, the same as the input of transformer blocks of ViT after the stem layer. We im-

Method	Reduction factor	Dataset storage size	# of images	Avg. size per image	Top1 Acc.
Full-pixels	100%	140.0 GB	1,281 k	109 kB	81.8
Uniform random sampling	70%	95.7 GB	897 k	107 kB	78.2
	40%	54.6 GB	512 k	107 kB	74.0
	20%	27.2 GB	256 k	107 kB	59.8
C-score [30] based sampling	60%	80.6 GB	769 k	105 kB	77.5
	40%	53.3 GB	512 k	104 kB	73.3
	20%	26.3 GB	256 k	103 kB	65.1
Adjusting image reolution	30%	16.0 GB	1,281 k	13 kB	78.6
	20%	9.6 GB	1,281 k	8 kB	75.2
	10%	5.3 GB	1,281 k	4 kB	63.3
Adjusting JPEG Qualities	10	14.0 GB	1,281 k	11 kB	78.1
	5	11.0 GB	1,281 k	9 kB	74.6
	1	9.3 GB	1,281 k	7 kB	67.8
SeiT (ImageNet-1k-5M, ours) [70]	-	7.5 GB	5,830 k	1 kB	78.6
SeiT (ImageNet-1k-5M, ours) [70]	60%	4.5 GB	3,498 k	1 kB	75.9
SeiT (ImageNet-1k, ours)	-	1.4 GB	1,281 k	1 kB	74.0

Table 2. **Main results.** ImageNet-1k results using various data storage reduction methods are shown. We compare SeiT against reduction factors that achieve comparable performance and storage size. Note that the numbers for all reduction factors are included in Appendix B.2.

plement the Stem-Adapter module as a convolutional layer with a kernel size of 4 and a stride of 2. This allows the module to capture the spatial relationships of adjacent tokens and produce a tensor that can be used as input to ViT. The comparison among the different Stem-Adapter architectures is included in Section Section 4.3.

4. Experiments

In this section, we conduct various experiments to demonstrate the effectiveness of token-based training. First, we compare SeiT with four image compression methods on ImageNet-1k [51]. Next, we explore the potential of SeiT as a large-scale pre-training dataset by employing the ImageNet-21k dataset. We also provide ablation studies on the proposed token augmentation methods and Stem-Adapter module to determine the effectiveness of each proposed element. Lastly, we evaluate a continual learning scenario on the ImageNet-100 [60] dataset to demonstrate the benefits of tokens in a limited memory environment.

4.1. ImageNet-1k classification

Our study on ImageNet-1k classification performance is summarized in Table 2 and Fig. 1. The blue and yellow lines in Fig. 1 depict the results of random sampling and sampling based on c-score [30], respectively. Random sampling of the dataset (yellow) had the most significant negative impact on performance as storage capacity decreased. On the other hand, sampling based on C-score [30] (blue) also resulted in a noticeable performance drop, but it performed better than random sampling when storage capacity reduced to 10% of the original. Although both sampling-based methods led to a considerable performance drop even with a small decrease in storage, JPEG-based compression methods (green) maintained their performance until storage reached 50% of the original. The green line in Fig. 1 illustrates the performance at different JPEG qualities. When the quality was set above 50, the performance remained nearly the same as the original, even with 24.3% of the original storage usage. However, when the quality was set to 1, with only 9.3GB storage, the performance dropped dramatically to only 67.8%. Adjusting the resolution (purple) showed a slightly different trend than adjusting the quality. Reducing the resolution achieved better results than reducing the quality as storage became smaller while reducing the quality performed better than reducing the resolution with relatively large storage. Despite the overall decline in the performance of image-based methods in low-storage environments, SeiT achieved 74.0% accuracy while using only 1% of the original storage. Furthermore, by employing ImageNet-1k-5M proposed in [70], we were able to access more storage on tokens and achieve 78.6% accuracy at 5% of the storage size, where JPEG-based methods demonstrated performances lower than 75%. These results highlight the effectiveness of our proposed method in improving performance in low-storage scenarios compared to traditional image-based methods.

We also evaluate our token-trained model and the image-trained model on robustness datasets, such as adding Gaussian noise, adding Gaussian blur, ImageNet-R [26], and ad-

Pre-training IN-21k	Fine-tuning IN-1k	Storage Size	 Ratio	Acc.
-	Pixels	140 GB	100.0%	81.8†
Tokens	Tokens	16 GB	11.1%	81.1
Tokens	Pixels	154 GB	110.0%	82.6
Tokens	Tokens → Pixels	156 GB	111.4%	**82.8**

Table 3. **Impact of storage-efficient pre-training (PT) and fine-tuning (FT).** We show the scenario of storage-efficient PT; we pre-train a model with a tokenized ImageNet-21k with more data points and fine-tune the model on the pixel or the token ImageNet-1k dataset. † is from the original paper. "Tokens → Pixels" denotes three-staged FT, Token 21k PT, Token 1k PT and Pixels FT.

versarial attacks [39, 17] in Appendix B.5. In summary, we observe that without strong pixel-level augmentations, SeiT shows lower performance drops compared to the pixel-trained counterparts on corruptions and distribution shifts. SeiT shows a significant gradient-based attack robustness compared to the pixel-training.

4.2. Storage-efficient token pre-training

We extract tokens from ImageNet-21k dataset and pre-trained a ViT-B/16 model on the tokenized ImageNet-21k to determine the effectiveness of tokens as a large-scale pre-training. We then fine-tuned the pre-trained model with both tokenized ImageNet-1k and full-pixel ImageNet-1k, respectively (details are in Appendix B.1). Additionally, we extend our storage-efficient pre-training in three stages, namely, 21k token pre-training → 1k token pre-training → 1k image fine-tuning, following BeiT v2 [44] (details are in Appendix B.3). The results are shown in Table 3.

The use of large-scale tokens for pre-training improved not only the performance of ImageNet-1k benchmarks using tokens but also the performance of full-pixel images. Pre-training with ImageNet-21k tokens led to a 2.5% performance gain compared to using ImageNet-1k-5M tokens, using only 8GB more storage. Furthermore, our pre-training strategy improved full-pixel ImageNet-1k performance by 1.0% using only 11.4% more storage compared to the original full-pixel ImageNet-1k training.

4.3. Ablation study

We present an analysis of the proposed augmentation methods, Stem-Adapter architectures, and results on convolutional networks. Table 4 reports the impact of the proposed augmentations for tokens. We found that employing Token-CutMix not only stabilized the overall training procedures but also resulted in the largest performance gain (8.1%) compared to excluding it. The newly proposed methods for tokens, Embedding-Noise and Token-EDA, also showed performance improvements of 0.3% and 1.4%, respectively. Interestingly, these methods not only work effectively when used individually but also achieve higher performance when used in combination (74.0%).

Token-CutMix	Token-EDA	Emb-Noise	Acc. (ViT-B)
✗	✗	✗	63.8
✓	✗	✗	71.9
✓	✓	✗	72.2
✓	✗	✓	73.3
✓	✓	✓	**74.0**

Table 4. **Impact of the proposed augmentations.** ImageNet-1k validation accuracies for the combination of the proposed augmentations for tokens are shown.

	Linear	Convolution 2 × 2	 4 × 4
Accuracy	58.6	73.1	**74.0**

Table 5. **Stem-Adapter architectures.** We compare three Stem-Adapter architectures for ViT-B/16 on ImageNet-1k. Note that stride of Convolution layers set to 2.

We also assessed the impact of the Stem-Adapter architecture on performance in Table 5. We compared two different Stem-Adapter architectures with our design choice. Note that, we used a smaller learning rate of 0.0005 for the linear Step-Adapter because of its unstable convergence using a larger learning rate and an input size of 14 × 14 to match the number of input patches with the convolutional Stem-Adapters. The results validate that our decision to use Conv 4×4 as Stem-Adapter for ViT models yields the highest performance among the considered candidates.

We also investigated the applicability of SeiT to convolutional networks. The benchmark results on different architectures of ImageNet-1k are presented in Table 6. Note that token-based training only requires 1.4GB storage, which is merely 1% of the storage required for pixel-based training. To match the size of features after the stem layer, we used a deconvolutional Stem-Adapter for ResNet [25] models. Our findings indicate that SeiT can also be used for storage-efficient training of convolutional models.

Finally, we show the impact of the tokenizer in Appendix B.4. In summary, we observe that SeiT works well for various tokenizers, *e.g.*, ViT-VQGAN [68] and VQGAN [21] variants. We chose ViT-VQGAN considering the trade-off between the performance and the storage size, and it is solely trained on ImageNet-1k without external datasets.

4.4. Continual learning

To demonstrate the effectiveness of SeiT in memory-limited settings, we compare SeiT with full-pixel datasets in a continual learning scenario. Specifically, we employed the Reduced ResNet-18 architecture on the ImageNet-100 dataset [60] and evaluated the results following the Experience Relay [48]. We observed that when using the same

Network	Pixel-based training Acc.	Pixel-based training Storage	Token-based training Acc.	Token-based training Storage
ViT-S [20]	79.9	140GB	73.5	1.4GB
ResNet-50 [25]	76.1	140GB	67.7	1.4GB
ResNet-18	69.7	140GB	58.0	1.4GB

Table 6. **Comparisons on various architectures.** We additionally compare the performances of the pixel-training and token-training accuracies of three architectures, including ViT-S, ResNet-50, and ResNet-18, on the ImageNet-1k benchmark.

memory size, SeiT is significantly more memory-efficient than images, with a storage capacity of 147 times that of images. As a result, the total memory required to store the entire dataset in tokens was less than 500MB. Fig. 4 illustrates the comparison results between using a token dataset and a full-pixel dataset in three different settings.

The left figure shows the performances of the token dataset and the full-pixel dataset by increasing memory size while fixing the number of tasks to ten. SeiT outperforms the pixel dataset and shows a neglectable performance drop even when the memory size decreased, as it stored sufficient data even with memory sizes below 100MB.

The center figure presents the results of changing the number of tasks with a fixed memory size of 574MB (≈ 1k images). In this case, both token and full-pixel datasets exhibited decreased performance as the number of tasks increased. However, the performance degradation of the token dataset was less severe than that of the full-pixel dataset.

Finally, with both memory size and the number of tasks fixed, we varied the number of times the dataset was viewed per task (the right figure). When there was only one task, the full-pixel dataset outperformed the token dataset as the epoch increased, consistent with other classification benchmark results. However, when there were ten tasks, the full-pixel dataset had lower performance than the token dataset, even with increased epochs due to insufficient stored data.

4.5. Implementation details

We used a pre-trained ViT-VQGAN Base-Base [68] model for extracting tokens from the images. Extracting tokens of entire ImageNet-21k dataset took 1.1 hours using 64 A100 GPUs with 2048 batch-size. We conducted ImageNet-1k benchmark experiments using the ViT-B/16 model [20, 61] with an input size of 224 x 224. For token ImageNet-1k training, we replaced the patch embedding layer in ViT-B/16 model with the proposed Stem-Adapter module and added a global pooling layer before the final norm layer for tokens. We used a learning rate of 0.0015 with cosine scheduling and a weight decay of 0.1. The model was trained for 300 epochs with a batch size of 1024. We followed the training recipe proposed in DeiT [61] for remaining settings except for the data augmentations. We also followed the training recipe proposed in DeiT for the full-pixel ImageNet-1k training but made a few adjustments to handle the reduced datasets. We used a smaller learning rate of 0.0009 with a batch size of 1024 compared to the original value of 0.001, as we found that the original learning rate did not converge well on smaller datasets. Also, we increased the number of warm-up epochs and total training iterations when the number of data points decreased to ensure a fair comparison. For large-scale token pretraining and token fine-tuning, we adopted simple augmentation strategies as suggested in DeiT-III [62]; we excluded Token-EDA and replaced RRC with a simple random crop. Following the DeiT-III training recipe, we pre-trained the model with tokenized ImageNet-21k dataset for 270 epochs and then we fine-tuned the model for 100 epochs both of token and full-pixel dataset using learning rates of 0.00001 with 4096 batch-size and 0.0005 with 1024 batch-size, respectively. We provide the more detailed hyper-parameter setting of our experiments in Appendix B.1.

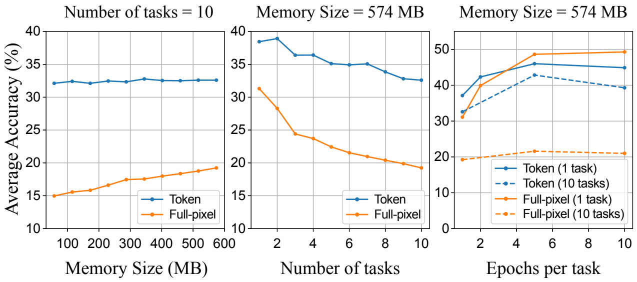


Figure 4. **Comparisons on the continual learning task.** We train two Experience Replay (ER) [48] models on the ImageNet-100 [60] dataset using the pixel dataset and the token dataset. (a) By varying the memory size while the number of tasks is fixed by 10. (b) By varying the number of tasks while fixing the memory size. (c) By increasing the epochs per task. Note that except (c), we set the epochs per task to 1 following the original setting [48].

5. Conclusion

In this paper, we propose **S**torage-**e**fficient **Vi**sion **T**raining (**SeiT**) by storing images into tokens. In practice, we store an image into 1kB as a 32×32 token sequence and propose an efficient and fast encoding and decoding strategy for the token data type. We also propose token augmentations and Stem-Adaptor to train vision transformers with minimal modifications from the highly-optimized pixel-based training. Our experiments show that compared to the other storage-efficient training methods, SeiT shows significantly large gaps; with the same amount of storage size, SeiT shows the best performance among the comparison methods. Our method also shows benefits in other practical scenarios, such as storage-efficient large-scale pretraining and continual learning at scale.

Appendix

In this additional document, we describe more details of SeiT in Appendix A, including the details of token encoding-decoding algorithms (Appendix A.1), the visualization of Emb-noise augmented tokens (Appendix A.2). We also include additional experimental results in Appendix B, including the hyperparameter details (Appendix B.1), the full experimental results of Table 2 (Appendix B.2), the additional results on storage-efficient pretraining (Appendix B.3), exploring other tokenizers (Appendix B.4), and robustness benchmarks (Appendix B.5).

A. More Details for SeiT

A.1. Pseudo-code for Token Encoding-Decoding

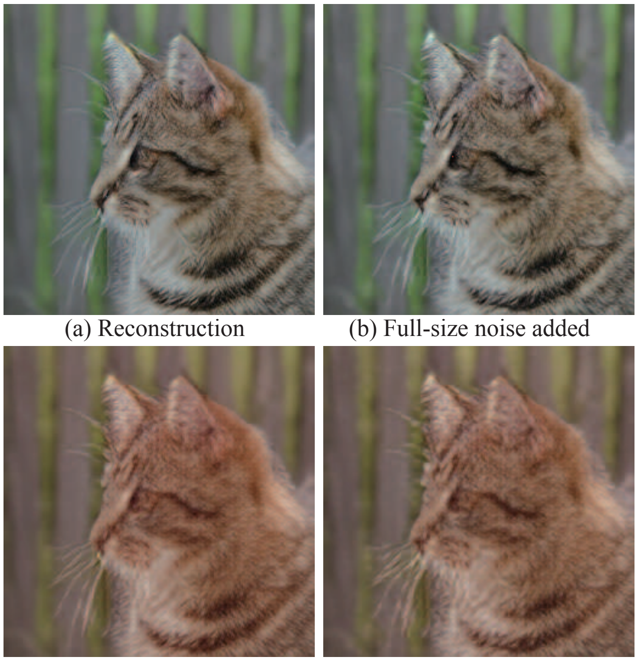
Algorithm 1 and Algorithm 2 describe the psuedo-codes for the proposed token encoding and decoding. Here, we assume $\max t_i < 2^{M+1}$ for the simplicity. For example, in our main experiments, each token belongs to 391 classes and we set $M = 8$, hence, $\max t_i = 391 < 2^{8+1} = 512$. If the number of token classes is larger than 2^{M+1}, then our algorithm can be naturally extended by repeating line 6-7 in Algorithm 1. By this simple algorithm, we achieved a nearly optimal compression ratio (1.11 kB vs. 1.08 kB per image) where almost 0.63 smaller than the 16-bit encoding (2.0 kB per image). Note that, we use the native gzip library to perform Huffman encoding and decoding for simplicity.

Algorithm 1 An algorithm for token encoding

```
Require: A sequence of tokens $T = [t_1, \dots, t_N]$, the bits for the storage $M$
 1: $L_T \leftarrow [\phi]$                ▷ Initialize an empty list for tokens
 2: $L_{\text{idx}} \leftarrow [\phi]$      ▷ Initialize an empty list for start indices
 3: $j \leftarrow 0$
 4: while $i \leq$ N do
 5:     if $t_i \geq 2^M$ then
 6:         $L_T$.append($2^M$)
 7:         $L_T$.append($t_i - 2^M$)
 8:         $j \leftarrow j + 2$   ▷ Assume $t_i < 2^{M+1}$ for simplicity
 9:     else
10:         $L_T$.append($t_i$)
11:         $j \leftarrow j + 1$
12:     end if
13:     $L_{\text{idx}}$.append($j$)
14:     $i \leftarrow i + 1$
15: end while
16: Return: Huffman encoding($L_T, L_{\text{idx}}$)
```

Algorithm 2 An algorithm for token decoding

```
Require: A compressed bytestring $L_T\prime$ from Algorithm 1
$L_T = [t_0, \dots, t_N], L_{\text{idx}} \leftarrow$ Huffman deencoding($L'_T, L'_{\text{idx}}$)
$T \leftarrow [\phi]$
$i \leftarrow 0$
while $L_{\text{idx}}$ is not empty do
    $j \leftarrow L_{\text{idx}}$.pop(0)
    $k \leftarrow i$
    while $k \leq j$ do
        if $t_k \geq M$ then
            $T$.append($t_k + t_{k+1}$)
            $k \leftarrow k + 2$
        else
            $T$.append($t_k$)
            $k \leftarrow k + 1$
        end if
    end while
    $i \leftarrow j$
end while
Return: $T$
```



(a) Reconstruction (b) Full-size noise added

(c) Channel-wise noise added (d) Ours

Figure A.1. **Emb-Noise visualization.** "Reconstruction" denotes the reconstructed image by the ViT-VQGAN decoder from the extracted tokens. "Full-size noise" is a random noise whose size is equivalent to the embedding vectors.

A.2. ViT-VQGAN decoded images for Emb-Noise

Fig. A.1 and Fig. A.2 show the visualization examples of the Emb-Noise augmented tokens and the tokens without

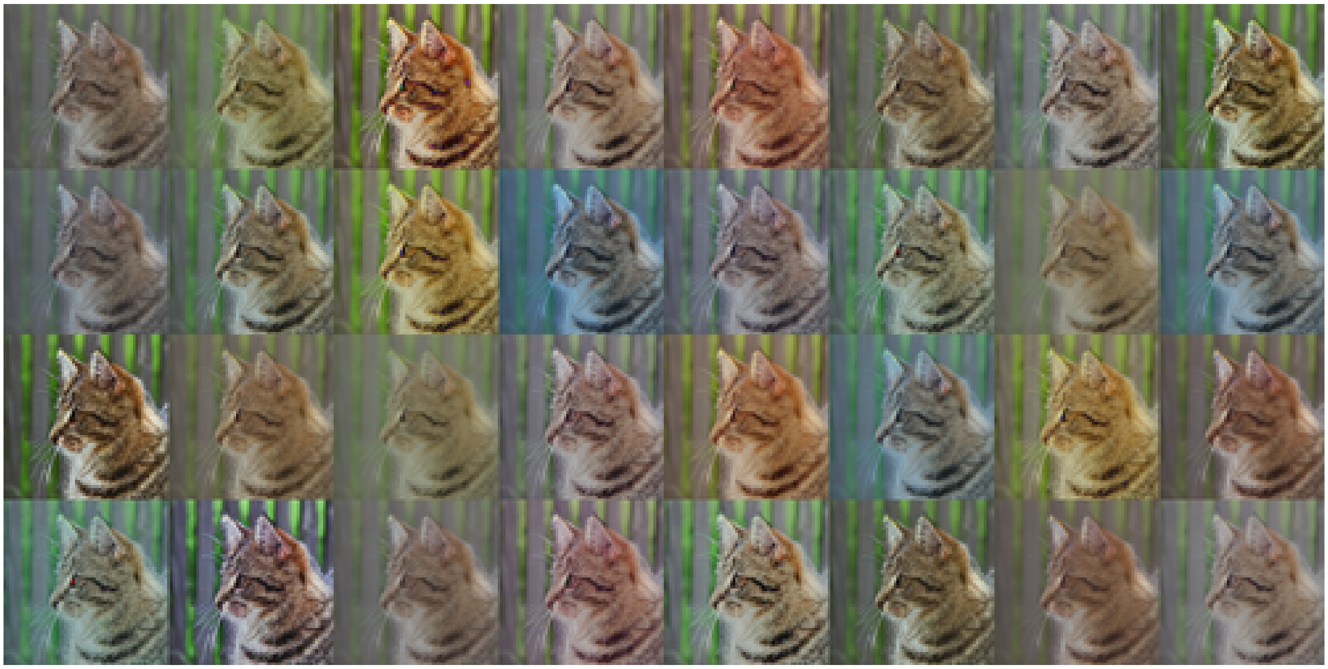


Figure A.2. **Channel-wise modification visualization.** We present ViT-VQGAN decoded images obtained by adding a constant to each of the 32 channels in codebook vectors.

augmentation. We use the ViT-VQGAN decoder for visualization. We observe that our Emb-Noise can make meaningful distortions on the decoded images.

B. Additional Experimental Results

B.1. Hyperparameter details

Table B.1 shows the full list of hyperparameters used in our experiments. All hyperparameters are for the ViT-B backbones. In the table, Token IN-1k corresponds to SeiT (ImageNet-1k) in Table 2, Token IN-21k PT corresponds to token pre-training in Table 3, and Token FT and Image FT correspond to token and image fine-tuning in Table 3, respectively. For other backbones and datasets, we only adjust the learning rate as the maximum learning rate showing a stable convergence (*e.g.*, We use 0.15 for ResNet and ViT-S uses the same learning rate as ViT-B).

B.2. The full experimental results

We report the full experimental results in Table B.2. Details are the same as Table 2.

B.3. Three-stage storage-efficient pre-training

Following BeiT v2 [44], we extend our storage-efficient pre-training in three stages, namely, 21k token pre-training → 1k token pre-training → 1k image fine-tuning. For simplicity, we directly fine-tune the "21k token pre-trained and 1k token fine-tuned model" (*i.e.*, 81.1% model in Table 3) on the image pixels with the same optimization hyperparameter of the image fine-tuned model. As a result, we have 82.8% top-1 accuracy, slightly better than the original two-staged training strategy (+0.2% than 82.6%).

B.4. Exploring other tokenizers

In this subsection, we explore other tokenizers rather than ViT-VQGAN [68], *e.g.*, VQGAN [21]. We employ three VQGAN models from the official repository[3], ImageNet-trained VQGAN with patch size 16 and vocabulary size 1024, ImageNet-trained VQGAN with patch size 16 and vocabulary size 16384, and OpenImages[34]-trained VQGAN with patch size 8 and vocabulary size 8192. Here, the last VQGAN model is trained with the Gumbel softmax [28, 38] quantization, instead of the original vector quantization by VQ-VAE[63]. Here, we slightly change our Stem-Adopter from 4×4 Conv with stride 2 to 2×2 Conv with stride 1 for tokenizers with patch size 16.

In Table B.3, we report the ViT-S (SeiT) top-1 accuracy on the ImageNet-100 benchmark by varying the choice of tokenizers. We also report the reported ImageNet-1k validation FID score of each tokenizer. In the table, we observe that the top-1 accuracy of SeiT follows the generation quality (FID) if we use the same quantization method (*e.g.*, vector quantization). The ViT-VQGAN shows the best FID (1.28) as well as the best ImageNet performance with SeiT (77.3). While the Gumbel quantized VQGAN achieves the

[3] https://github.com/CompVis/taming-transformers

Methods	DeiT IN-1k [61]	Token IN-1k	Token IN-21k PT	Token IN-1k FT	Image IN-1k FT
Epochs	300	300	270	100	100
Batch size	1024	1024	2048	4096	512
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Learning rate	0.0005 x $\frac{bs}{512}$	0.00075 x $\frac{bs}{512}$	0.0015	0.00001	0.0005
Learning rate decay	cosine	cosine	cosine	✘	cosine
Weight decay	0.05	0.1	0.02	0.1	0.05
Warmup epochs	5	5	5	5	5
Label smoothing	0.1	0.1	0.1	0.1	0.1
Dropout	✘	✘	✘	✘	✘
Stoch. Depth	0.1	0.1	0.1	0.15	0.1
Gradient Clip	✘	✘	✘	✘	✘
Cutmix prob.	1	1	1	1	1
Mixup prob.	0.8	0	0	0	0.8
RandAug	9 / 0.5	-	-	-	9 / 0.5
Repeated Aug	✔	-	-	-	✘
Erasing prob.	0.25	-	-	-	0
EDA prob.	-	0.25 (RS) / 0.25 (SR)	0	0	-
Emb-Noise prob.	-	0.5	0.5	0.5	-

Table B.1. **Hyperparamters for SeiT and DeiT-B.** All hyperparameters are for the ViT-B backbone. DeiT IN-1k is the same as the original DeiT paper (baseline).

Method		Storage size	# of images	Top1 Acc.
Full-pixels	100%	140 GB	1.28 M	81.8
Uniform random sampling	20%	27.2 GB	0.26 M	59.8
	30%	41.0 GB	0.38 M	69.3
	40%	54.6 GB	0.51 M	74.0
	50%	68.4 GB	0.64 M	76.0
	60%	82.0 GB	0.77 M	77.8
	70%	95.7 GB	0.90 M	78.2
	80%	109.3 GB	1.02 M	79.4
	90%	123.1 GB	1.15 M	81.1
C-score based sampling	20%	26.3 GB	0.26 M	65.1
	30%	39.8 GB	0.38 M	69.4
	40%	53.3 GB	0.51 M	73.3
	50%	66.9 GB	0.64 M	76.9
	60%	80.6 GB	0.77 M	77.5
	70%	94.3 GB	0.90 M	79.2
	80%	108.1 GB	1.02 M	80.4
	90%	121.8 GB	1.15 M	80.9
Adjusting image reolution	10%	5.3 GB	1.28 M	63.3
	20%	9.6 GB	1.28 M	75.2
	30%	16 GB	1.28 M	78.6
	40%	24 GB	1.28 M	79.4
	50%	34 GB	1.28 M	80.9
	60%	46 GB	1.28 M	80.8
	70%	60 GB	1.28 M	81.6
	80%	75 GB	1.28 M	81.6
	90%	93 GB	1.28 M	80.8
Adjusting JPEG Qualities	1	9.3 GB	1.28 M	67.8
	5	11 GB	1.28 M	74.6
	10	14 GB	1.28 M	78.1
	25	23 GB	1.28 M	80.7
	50	34 GB	1.28 M	81.1
	75	50 GB	1.28 M	81.5
	85	66 GB	1.28 M	81.3
	90	79 GB	1.28 M	80.9
	95	113 GB	1.28 M	81.6
SeiT (ImageNet-1k, ours)		1.36 GB	1.28 M	74.0
SeiT (ImageNet-1k-5M, ours)		7.49 GB	5.83 M	78.6
SeiT (IN-21k tokens → 1k tokens, ours)		16 GB	12.4 M	81.1
SeiT (IN-21k tokens → 1k pixels, ours)		154 GB	12.4 M	82.6
SeiT (IN-21k tokens → 1k tokens → 1k pixels, ours)		156 GB	12.4 M	82.8

Table B.2. **The full main results.** The full results of Fig. 1, Table 2 and Table 3.

best performance, in practice, we use ViT-VQGAN due to two reasons. First, the storage efficiency: 2886 valid codes need 1.5 times more storage than 391 valid codes. Second, Although the Gumbel quantized VQGAN shows better quality, it needs to be trained on a large-scale external dataset, OpenImages [34]. We did not use the OpenImaged-trained Gumbel quantized VQGAN for a fair comparison with other ImageNet-1k-only training methods.

B.5. Robustness benchmarks

We compare ViT-S models trained on ImageNet-1k with different training strategies using robustness benchmarks. We employ three scenarios: (1) noise and blur scenario (2) domain shift scenario (3) adversarial attack scenario. For the first scenario, we add Gaussian noise and Gaussian blur to the validation images. We use ImageNet-R [26] and Sketch-ImageNet [65] for testing the robustness against domain shifts. Finally, we use a weak version of AutoAttak [17] for measuring adversarial robustness.

As the original DeiT is trained on strong augmentation, such as RandAugment or 3-Augment, we also compare our method with "weak augmented" ViT-S, where it only employs resized random crop (RRC) and CutMix [69]. Our assumption is that because the pixel-trained models are sensitive to imperceptible details, they will be less robust than our approach in noise or adversarial attack scenarios. However, on the other hand, because our method relies on the encoding power of the pre-trained tokenizer, if the employed tokenizer is not a robust feature extractor, our method could be more vulnerable than pixel-trained counterparts.

Table B.4 shows the results of the first and the second scenarios. Here, we observe two important findings. First, when we use the same augmentations with the same strength (ViT-S Weak Aug vs. ViT-S SeiT), SeiT shows smaller performance drops on both noise scenarios and do-

Tokenizer	Training dataset	Quantiztation	Voca size (# of valid voca)	PS	FID	ViT-S (SeiT) Acc
VQGAN	ImageNet	Vector quantization	1024 (454)	16	7.94	75.3
VQGAN	ImageNet	Vector quantization	16384 (971)	16	4.98	76.9
VQGAN	OpenImages	Gumbel quantization	8192 (2886)	8	1.49	**79.1**
ViT-VQGAN	ImageNet	Vector quantization	8192 (391)	8	1.28	77.3

Table B.3. **Exploring other tokenizers.** Various ViT-S (SeiT) results on the ImageNet-100 benchmark are shown. We compare various VQGAN tokenizers with ViT-VQGAN by varying the quantization methods (Gumbel softmax vs. vector quantization) the vocabulary size, the valid vocabulary size (the number of classes actually used for the ImageNet-1k training dataset), and the patch size (PS).

Model	Data format	Clean	Gauss. Noise	Gauss. Blur	ImageNet-R	Sketch
ViT-S (DeiT)	Pixels	79.9	75.1 (6.0%)	73.4 (8.1%)	28.8 (63.9%)	29.9 (62.6%)
ViT-S (Weak Aug)	Pixels	78.0	64.7 (**17.1%**)	66.8 (14.4%)	20.8 (73.4%)	18.1 (76.8%)
ViT-S (SeiT, ours)	Tokens	74.0	60.8 (17.3%)	65.3 (**11.2%**)	26.0 (**64.6%**)	23.0 (**68.7%**)

Table B.4. **Robustness evaluation.** We show the clean and robust accuracies against corruptions and domain shifts of each model trained on ImageNet-1k. The performance drops are put in parentheses (lower is better) for robust accuracies.

main shift scenarios. On the other hand, when we use strong pixel-level augmentations, the pixel-trained counterpart outperforms our approach. It implies that the key to the input pixel robustness depends on the pixel-level augmentations with severe distortions as observed by previous studies [13, 58]. However, because our method uses only tokens, not pixels directly, investigating how to explore pixel-level distortion augmentations on the token level will be an open question and an interesting future research direction.

We also compare the adversarial robustness of DeiT-S and SeiT-S. We employ the APGD (a step size-free version of PGD attack [39]) with cross-entropy loss and DLR loss, following AutoAttack [17]. Because SeiT employs discrete non-differentiable representations in the computational graph, we employ the straight-through estimator (STE) [9] to estimate the non-differentiable gradients, following Athalye *et al*. [5]. We also evaluate the non-quantized version of the quantizer (*i.e*., omitting the vector quantization process, but using the extracted feature by the encoder directly to the ViT input), but we empirically observe that attacking the non-quantized version cannot drop the performance at all. Instead, we use the STE, also used during the training as well as the previous extensive robustness study [5]. We compare the attacked accuracies of DeiT and SeiT by varying ε (a control parameter for the attack intensity) from 0 to 8 in Fig. B.1. We observe that SeiT shows almost neglectable performance drops even under the strongest attack (showing 73.95 for $\varepsilon = 8$ where 73.98 for $\varepsilon = 0$), where DeiT shows 2.8% top-1 accuracy.

However, we should be careful to interpret Fig. B.1; it could be due to a strong obfuscated gradient effect [5] that cannot be detected by a naive straight-through estimator. Moreover, our method could be vulnerable to the codebook attack by changing the token indices directly, not by perturbing the pixels. However, as an efficient and natural adversarial attack on discrete domains is still an open problem [72] (*e.g*., altering indices as imperceptible to humans but sensitive to machines — only a small index change can make a huge semantic gap, such as replacing "huge" in the previous sentence to "neglectable"), we leave the investigation of advanced adversarial attack methods for SeiT beyond straight-through estimator as future work.

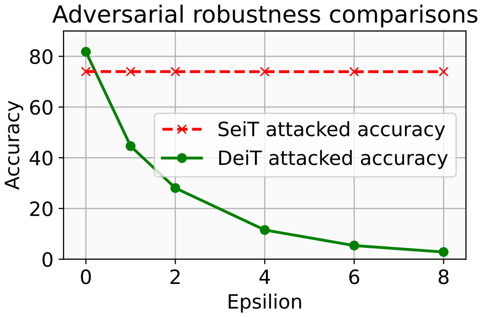


Figure B.1. **Adversarial robustness of DeiT and SeiT by varying ε.** $\varepsilon = 0$ denotes the clean accuracy.

References

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 3

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018. 2, 3

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015. 3

[4] Nicola Asuni and Andrea Giachetti. Testimages: a large-scale archive for testing visual devices and basic image processing algorithms. In *STAG*, pages 63–70, 2014. 2

[5] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018. 12

[6] Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. Learning de-biased representations with biased representations. In *International Conference on Machine Learning (ICML)*, 2020. 2

[7] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016. 2, 3

[8] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018. 2, 3

[9] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 12

[10] Shweta Bhardwaj, Mukundhan Srinivasan, and Mitesh M Khapra. Efficient video classification using fewer frames. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 354–363, 2019. 3

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[12] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7939–7948, 2020. 3

[13] Sanghyuk Chun, Seong Joon Oh, Sangdoo Yun, Dongyoon Han, Junsuk Choe, and Youngjoon Yoo. An empirical evaluation on robustness and uncertainty of regularization methods. *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2019. 12

[14] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019. 3

[15] Eastman Kodak Company. Kodak lossless true color image suite (photocd pcd0992). 2

[16] Thomas M Cover. Elements of information theory. chapter 5. John Wiley & Sons, 1999. 4

[17] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020. 2, 7, 11, 12

[18] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 5

[19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005. 2

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 8

[21] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021. 7, 10

[22] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[23] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 2

[24] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017. 3

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7, 8

[26] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021. 6, 11

[27] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952. 4

[28] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. 10

[29] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021. 1

[30] Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C Mozer. Characterizing structural regularities of labeled data in overparameterized models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5034–5044. PMLR, 18–24 Jul 2021. 2, 3, 6

[31] Jun-Hyuk Kim, Byeongho Heo, and Jong-Seok Lee. Joint global and local hierarchical priors for learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5992–6001, 2022. 3

[32] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. *Advances in neural information processing systems*, 31, 2018. 3

[33] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR, 2021. 3

[34] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International Journal of Computer Vision*, 128(7):1956–1981, 2020. 10, 11

[35] Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. Dataset condensation with contrastive signals. In *International Conference on Machine Learning (ICML)*, 2022. 3

[36] Shiye Lei and Dacheng Tao. A comprehensive survey to dataset distillation. *arXiv preprint arXiv:2301.05603*, 2023. 3

[37] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. *Advances in neural information processing systems*, 29, 2016. 3

[38] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. 10

[39] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 2, 7, 12

[40] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pages 181–196, 2018. 1

[41] Feng Mao, Xiang Wu, Hui Xue, and Rong Zhang. Hierarchical video frame sequence representation with deep convolutional graph network. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018. 3

[42] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6950–6960. PMLR, 13–18 Jul 2020. 2, 3

[43] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34, 2021. 2, 3

[44] Zhiliang Peng, Li Dong, Hangbo Bao, Qixiang Ye, and Furu Wei. Beit v2: Masked image modeling with vector-quantized visual tokenizers. *arXiv preprint arXiv:2208.06366*, 2022. 7, 10

[45] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1, 3

[46] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 2

[48] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 3, 7, 8

[49] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 1

[50] Andrea Rosasco, Antonio Carta, Andrea Cossu, Vincenzo Lomonaco, and Davide Bacciu. Distilled replay: Overcoming forgetting through synthetic samples. In *Continual Semi-Supervised Learning: First International Workshop, CSSL 2021, Virtual Event, August 19–20, 2021, Revised Selected Papers*, pages 104–117. Springer, 2022. 3

[51] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 1, 2, 4, 6

[52] Mattia Sangermano, Antonio Carta, Andrea Cossu, and Davide Bacciu. Sample condensation in online continual learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 01–08. IEEE, 2022. 3

[53] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. 1, 2

[54] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. In *NeurIPS Data-Centric AI Workshop*, 2021. 2

[55] Luca Scimeca, Seong Joon Oh, Sanghyuk Chun, Michael Poli, and Sangdoo Yun. Which shortcut cues will dnns choose? a study from the parameter-space perspective. In *International Conference on Learning Representations (ICLR)*, 2022. 2

[56] Mannat Singh, Laura Gustafson, Aaron Adcock, Vinicius de Freitas Reis, Bugra Gedik, Raj Prateek Kosaraju, Dhruv Mahajan, Ross Girshick, Piotr Dollár, and Laurens Van Der Maaten. Revisiting weakly supervised pre-training of visual perception models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 804–814, 2022. 1

[57] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 3

[58] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020. 12

[59] Damien Teney, Peter Anderson, Xiaodong He, and Anton Van Den Hengel. Tips and tricks for visual question answering: Learnings from the 2017 challenge. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4223–4232, 2018. 3

[60] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer, 2020. 6, 7, 8

[61] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021. 2, 3, 4, 8, 11

[62] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*, pages 516–533. Springer, 2022. 5, 8

[63] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 10

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[65] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019. 11

[66] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018. 3

[67] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019. 2, 5

[68] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*, 2021. 2, 4, 7, 8, 10

[69] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision (ICCV)*, 2019. 2, 5, 11

[70] Sangdoo Yun, Seong Joon Oh, Byeongho Heo, Dongyoon Han, Junsuk Choe, and Sanghyuk Chun. Re-labeling imagenet: from single to multi-labels, from global to localized labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2340–2350, 2021. 6

[71] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, pages 15–28, 2012. 1

[72] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020. 12

[73] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, 2021. 2, 3

[74] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021. 2, 3