

Peeling the Onion: Hierarchical Reduction of Data Redundancy for Efficient Vision Transformer Training

Zhenglun Kong ^{† 1}, Haoyu Ma ^{† 2}, Geng Yuan ^{† 1}, Mengshu Sun ¹, Yanyue Xie ¹, Peiyan Dong ¹,
 Xin Meng ³, Xuan Shen ¹, Hao Tang ⁴, Minghai Qin ⁵ Tianlong Chen ⁶ Xiaolong Ma ⁷
 Xiaohui Xie ² Zhangyang Wang ⁶ Yanzhi Wang ¹

¹ Northeastern University, ² University of California, Irvine, ³ Peking university, ⁴ ETH Zurich, ⁵ Western Digital Research, ⁶ University of Texas at Austin, ⁷ Clemson University

¹ {kong.zhe,yuan.geng}@northeastern.edu, ² haoyum3@uci.edu,

Abstract

Vision transformers (ViTs) have recently obtained success in many applications, but their intensive computation and heavy memory usage at both training and inference time limit their generalization. Previous compression algorithms usually start from the pre-trained dense models and only focus on efficient inference, while time-consuming training is still unavoidable. In contrast, this paper points out that the million-scale training data is redundant, which is the fundamental reason for the tedious training. To address the issue, this paper aims to introduce sparsity into data and proposes an end-to-end efficient training framework from three sparse perspectives, dubbed *Tri-Level E-ViT*. Specifically, we leverage a hierarchical data redundancy reduction scheme, by exploring the sparsity under three levels: number of training examples in the dataset, number of patches (tokens) in each example, and number of connections between tokens that lie in attention weights. With extensive experiments, we demonstrate that our proposed technique can noticeably accelerate training for various ViT architectures while maintaining accuracy. Remarkably, under certain ratios, we are able to improve the ViT accuracy rather than compromising it. For example, we can achieve 15.2% speedup with 72.6% (+0.4) Top-1 accuracy on Deit-T, and 15.7% speedup with 79.9% (+0.1) Top-1 accuracy on Deit-S. This proves the existence of data redundancy in ViT. Our code is released at <https://github.com/ZLKong/Tri-Level-ViT>

Introduction

After the convolutional neural networks (CNNs) dominated the computer vision field for more than a decade, the recent vision transformer (ViT) (Dosovitskiy et al. 2020) has ushered in a new era in the field of vision (Hudson and Zitnick 2021; Chen et al. 2021c; Kim et al. 2021; Deng et al. 2021; Xue, Wang, and Guo 2021; Zhao et al. 2021; Guo et al. 2021; Srinivas et al. 2021). Existing ViT and variants, despite the impressive empirical performance, suffer in general from large computation effort and heavy run-time memory usages (Liang et al. 2021; Chen, Fan, and Panda 2021; Carion et al. 2020; Dai et al. 2021; Amini, Periyasamy, and Behnke 2021; Misra, Girdhar, and Joulin 2021; Chen et al. 2021d; El-Nouby et al. 2021; Yang et al. 2020; Chen et al. 2021a; Lu et al. 2021). To reduce the computational and memory intensity of the ViT models, many compression algorithms have been

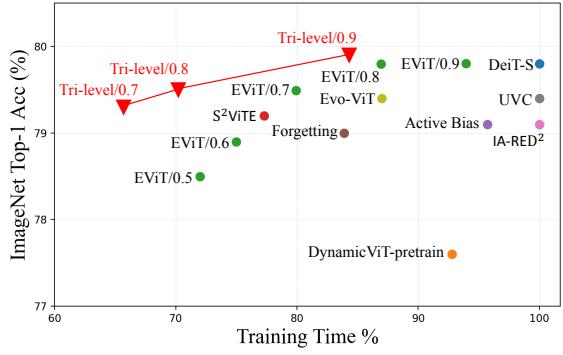


Figure 1: Comparison of different models with various accuracy-training efficiency trade-off. The proposed Tri-level method achieves a better trade-off than the other methods.

proposed (Ryoo et al. 2021; Pan et al. 2021; Liang et al. 2022; Yu et al. 2022b; Kong et al. 2022). These methods mainly target efficient inference, and they are either conducted during the fine-tuning phase with a pre-trained model or require the full over parameterized model to be stored and updated during training (Rao et al. 2021; Kong et al. 2022).

The self-attention mechanism in ViTs abandons the inductive bias that is inherent in CNNs, making the training of ViTs extremely data-hungry. Specifically, the early ViT work (Dosovitskiy et al. 2020) requires to be trained on a much larger dataset (i.e., JFT-300M with 300 million images) to claim its superiority compared to CNNs that are directly trained using the ImageNet dataset. This issue is partially mitigated by the following works, which incorporate knowledge distillation techniques to transfer inductive bias from a convolutional teacher model during training (Touvron et al. 2021) or introduce the design of CNNs into transformer blocks (Liu et al. 2021; Heo et al. 2021). However, the time-consuming training process still remains computational and memory intensive, and even more costly than before, becoming a huge obstacle to a vast number of ViT-based research and applications. Therefore, an efficient training paradigm for ViT models is in demand more than ever. Motivated by such needs, we raise a question: *Can we incorporate data sparsity into the training process to amortize the high training costs caused by the data-hungry nature of the ViTs?*

[†]These authors contributed equally

Due to the quadratic image patch-based computation complexity in the self-attention mechanism of ViT models (Dosovitskiy et al. 2020), it is desirable and reasonable to explore the sparsity by leveraging the redundancy that inherent in natural image data. Towards this end, we propose an efficient ViT training framework, named Tri-Level E-ViT, which hierarchically eliminates the data redundancy in different levels of the ViT training, *just like peeling the onion*. Specifically, to reduce the training costs and accelerate the training process, we explore a tri-level data sparsity, which includes *example-level sparsity*, *token-level sparsity*, and *attention-level sparsity*.

Prior works have only focused on removing redundant patches (tokens) instead of removing the whole image (training example) (Rao et al. 2021; Liang et al. 2022; Kong et al. 2022; Xu et al. 2021). This is due to the fact that the tendency to train a ViT with a large amount of data has become “deeply rooted” and led people to overlook the existence of redundancy in the example level. Motivated by this, we design a ViT-specific online example filtering method, to assess the importance of each example and to remove less important ones during training. More specifically, we use both the output classification logit and the variance of the attention map in the [CLS] token (Dosovitskiy et al. 2020) to evaluate the importance of each example. We train the model with a subset of the full training examples (e.g., 80%) and employ a remove-and-restore algorithm to continuously update the training subset, ensuring both the example-level sparsity and optimization throughout the entire ViT training process.

The ViT decomposes an image into several non-overlapping patches. The patch-based representation (Trockman and Kolter 2022) preserves locality and detailed information of an image. Thus, the patch can be considered a type of fine-grained data. The ViT conducts the dense self-attention among all patches, which leads to quadratic complexity with respect to the number of patches. Recent works suggest that not all patches are informative (Rao et al. 2021) and not all attention connections among patches are necessary (Liu et al. 2021; Heo et al. 2021). Thus, we further explore two levels of data sparsity from the perspective of the patch: One is token-level sparsity, which aims to reduce the number of patches (tokens), and the other is attention-level sparsity, which aims to remove redundant attention connections. We propose the Token&Attention selector (TA-selector) to simultaneously achieve both levels of sparsity during training. In detail, the TA-selector directly selects informative tokens and critical attention connections based on the attention maps of early layers. It is a data-dependent scheme and does not introduce any new weights. Thus, TA-selector is flexible to be installed into many ViT variants.

By incorporating the tri-level data sparsity, our framework significantly reduces the training costs and accelerates the training process, while maintaining the original accuracy. To the best of our knowledge, we are the first to explore all levels of data sparsity for efficient ViT training. As shown in Figure 1, our method achieves the best accuracy-efficiency trade-offs compared to existing sparsity methods. Most importantly, our Tri-Level E-ViT framework is efficient and practical since we do not require to introduce extra networks during training,

such as using a CNN teacher for distillation (Touvron et al. 2021) or a predictor network for token selection (Rao et al. 2021). In addition to the training acceleration, the data-sparse ViT model obtained by our framework can also be directly used for efficient inference without further fine-tuning. To verify the robustness, we evaluate the training acceleration of our framework on general-purpose GPU and FPGA devices.

Our contributions are summarized as follows:

- We propose an efficient framework that employs tri-level data sparsity to eliminate data redundancy in the ViT training process.
- We propose an attention-aware online example filtering method specifically for ViT to generate example-level sparsity. We use a remove-and-restore approach to ensure data efficiency throughout the entire training process while optimizing the reduced dataset.
- We propose a joint attention-based token&attention pruning strategy to simultaneously achieve both token and attention connection sparsity during training. Our method does not require a complex token selector module or additional training loss or hyper-parameters.
- We conduct extensive experiments on ImageNet with DeiT-T and DeiT-S, and demonstrate that our method can save up to 35.6% training time with comparable accuracy. We also evaluate the training acceleration on both GPU and FPGA devices.

Related Work

Vision Transformers. ViT (Dosovitskiy et al. 2020) is a pioneering work that uses transformer-only structure to solve various vision tasks. Compared to traditional CNN structures, ViT allows all the positions in an image to interact through transformer blocks whereas CNNs operated on a fixed sized window with restricted spatial interactions, which can have trouble capturing relations at the pixel level in both spatial and time domains. Since then, many variants have been proposed. For example, DeiT (Touvron et al. 2021), T2T-ViT (Yuan et al. 2021b), and Mixer (Tolstikhin et al. 2021) tackle the data-inefficiency problem in ViT by training only with ImageNet. PiT (Heo et al. 2021) replaces the uniform structure of transformer with depth-wise convolution pooling layer to reduce spacial dimension and increase channel dimension. SViTE (Chen et al. 2021b) alleviates training memory bottleneck and improve inference efficiency by co-exploring input token and attention head sparsity. Recent works have also attempted to design various self-supervised learning schemes for ViTs to address the data-hungry issue. BEiT (Bao, Dong, and Wei 2021) and PeCo (Dong et al. 2021) pre-train ViT by masked image modeling with an image tokenizer based on dVAE that vectorizes images into discrete visual tokens. MAE (He et al. 2021) eliminates the dVAE pre-training process by reconstructing pixels, in contrast to predicting tokens.

Data Redundancy Reduction. There are many attempts to explore redundancy in data. For example level, (Katharopoulos and Fleuret 2018) proposed an importance sampling scheme based on an upper bound to the gradient norm, which is added to the loss function. (Chang, Learned-Miller, and Mc-

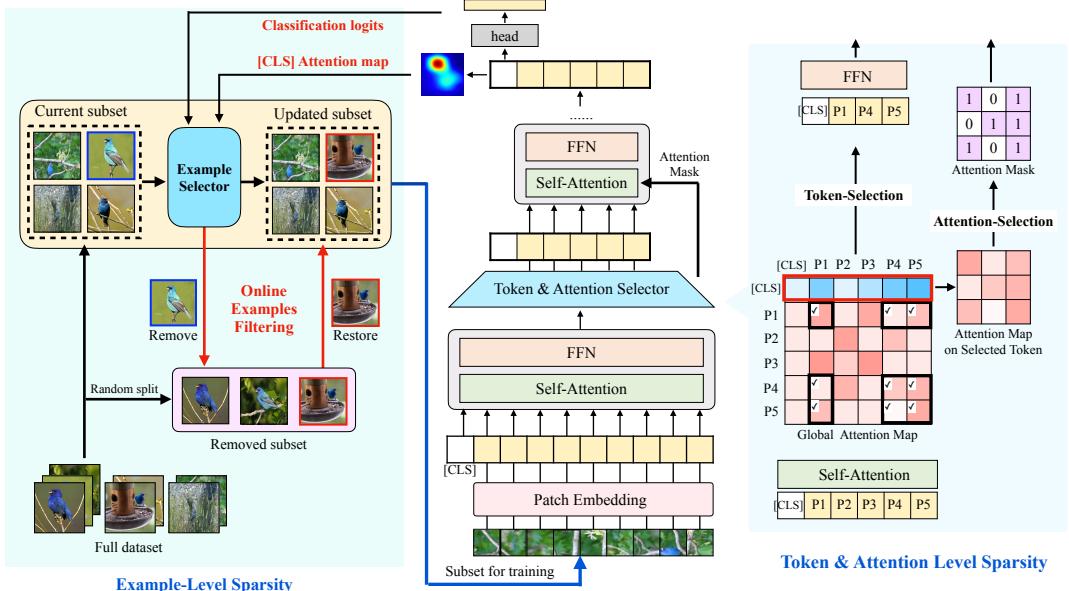


Figure 2: The overall procedure of our proposed Tri-Level E-ViT framework. We explore the training sparsity under three levels: **Left: Example-Level Sparsity**. Randomly remove an amount of examples prior to training. During the training process, the example selector updates the training subset by removing the most unforgettable examples from the training data and restore the same amount of data from the portion removed before training. **Right: Token & Attention level sparsity**. Evaluate token importance by the [CLS] token and prune the less informative tokens. We further use the attention vector of each remaining token to decide critical attention connections.

Callum 2017) emphasizes high variance samples by reweighting all the examples. However, the additional computation overhead makes them less efficient for training. (Toneva et al. 2018) used the number of forgotten counts of training examples as a metric to reflect the complexity (importance) of the examples. However, it requires a two-phase training approach that the partial training examples are only removed in the second phase, which limits the overall acceleration. Moreover, all the existing methods are applied only to CNN, making them less suitable for ViT, because their proposed criteria do not consider the ViT architecture for evaluation. For Patch level (token/attention), DynamicViT (Rao et al. 2021) removes redundant tokens by estimating their importance score with a MLP (Vaswani et al. 2017) based prediction module. Evo-ViT (Xu et al. 2021) develops a slow-fast token evolution method to preserve more image information during pruning. DAT (Xia et al. 2022) selects important key and value pairs in self-attention with a deformable self-attention module. However, all of these methods require additional selectors to guide the removal of redundant data, which is not efficient for training. EViT (Liang et al. 2022) provides a token reorganization method that reduces and reorganizes the tokens. However, it does not take into account example level or attention level redundancy.

Methodology

Overall Framework

To reduce the redundant computing caused by the training data, we propose the Tri-Level E-ViT. The overall framework is shown in Figure 2. Specifically, we introduce the sparsity

relevant to the data from three levels: The training example level, the token level, and the attention connection level. All of these sparse methods are combined together during the training process of ViTs.

Sparsity in Training Examples

Prior works (Toneva et al. 2018; Paul 2021) show that removing some less informative and easy examples from the training dataset will not lessen the performance of the trained CNN model. The amount of information (complexity) of the examples can be identified by recording the number of forgotten events of examples during CNN training. Examples with lower forgetting counts are generally considered easy to be learned and less informative than others, and can therefore be removed in order. More discussion are shown in the Appendix. We intend to incorporate this idea into our example-level sparse ViT training. We face two challenges: 1) *How to design such an example evaluation method for ViT, considering its unique model characteristics?* 2) *How to identify and remove easy examples from the dataset during training to reduce the training time, without prior knowledge of training examples?*

Attention Aware Example Selector We learn the conditional probability distribution given a dataset $D = (x_i; y_i)_i$ of observation/label pairs. For example x_i , the predicted label (classification logits) obtained after t steps of AdamW is denoted as $\hat{y}_i^t = \arg \max_k p(y_{ik}|x_i; \theta^t)$. Let $corr_i^t = \mathbb{B}_{\hat{y}_i^t = y_i}$ be a binary variable indicating whether the example is correctly classified at time step. When example i gets misclassified at step $t+1$ after having been correctly classified at

step t , example i undergoes a forgetting event. The binary variable corr_i^t decreases between two consecutive updates: $\text{corr}_i^t > \text{corr}_i^{t+1}$. On the other hand, a learning event occurs if $\text{corr}_i^t < \text{corr}_i^{t+1}$.

After counting the number of forgetting events for each example, we sort the examples based on the number of forgetting events they undergo. However, there exist examples with the same number of forgetting events. Even if the number is the same, the complexity of the images is still different. Therefore, in addition to the classification logits, we employ the unique self-attention mechanism in the transformer to better measure the complexity of images. We calculate the variance of the attention map of the [CLS] token (Dosovitskiy et al. 2020) to obtain the similarity of the image patches. We extract the attention map of the [CLS] token from the self-attention layer of the model and obtain the variance of the attention map, which is exported along with the classification logits to assist in sorting the images, as shown in Figure 2. Note that for ViT variants without [CLS] token (Swin), we replace the variance with the cumulative token of the attention map. After removing the unforgettable examples, the compressed training dataset \hat{D} is described as:

$$\hat{D} = \{x_i | x_i \in D, f(x_i) \geq \text{threshold}\}, \quad (1)$$

where $f()$ stands for the “sort examples by forgetting counts” process. Therefore, $f(x_i)$ is the list of examples x_i sorted by counts. “ $f(x_i) \geq \text{threshold}$ ” means choosing examples whose forgetting count is larger than the threshold. These selected examples will be our training subset \hat{D} .

Online Example Filtering for Efficient Training Existing example sparsity works (Toneva et al. 2018; Yuan et al. 2021a) on CNN split the training process into two phases: 1) Training with the complete dataset to obtain statistics on forgetting events. 2) Training with the remaining dataset by removing the less informative examples based on the learning statistics. In this way, training acceleration can only be achieved in the second phase, which only accounts for 40%~70% of the total training process (Yuan et al. 2021a). This greatly limits the overall acceleration performance.

Unlike their semi-sparse training approach, we use an end-to-end sparse training approach, which keeps the training dataset sparse throughout the training process. This is the first time that the example sparsity is introduced in the ViT training scenario. Our approach is shown in the left part of Figure 2. The main idea is to first remove a random portion of training examples from the dataset, and then continuously update the remaining training subset during the training process by using a remove-and-restore mechanism. After several iterations, only relatively more informative examples remain in the training subset.

Specifically, before the training starts, we first reduce the training subset by randomly removing a given $m\%$ of training examples. We define $r_e = 1 - m\%$ as the keep ratio of training examples. Then, during the training, we periodically remove the $n\%$ least informative examples from the training subset ($n < m$) and randomly select $n\%$ examples from the removed examples to restore to the training subset. And during the training process, we continuously track the number

of forgetting events for each training example and identify the least informative examples. In this way, the data set is optimized after several iterations during training, and $m\%$ the data size remains constant throughout the entire training process, leading to a more consistent training acceleration.

Sparsity in Patches: Token and Attention Level

The dense attention of ViTs leads to quadratic complexity with respect to the number of patches. Thus, introducing sparsity into patches can significantly reduce the computation. We consider both token-level sparsity and attention-level sparsity for patches based on the attention matrix.

Revisiting ViT The ViT decomposes image I into N non-overlapping tokens $\{\mathbf{P}_i\}_{i=1}^N$, and apply a linear projection to obtain the patch embedding $\mathbf{X}_P \in \mathbb{R}^{N \times d}$, where d is the dimension of embedding. The classification token \mathbf{X}_{cls} is appended to accumulate information from other tokens and predict the class. Thus, the input to the transformer is $\mathbf{X} = [\mathbf{X}_{\text{cls}}, \mathbf{X}_P] \in \mathbb{R}^{L \times d}$, where $L=N+1$ for short. The transformer encoder consists of a self-attention layer and a feed-forward network. In self-attention, the input tokens \mathbf{X} go through three linear layers to produce the query (\mathbf{Q}), key (\mathbf{K}), and value (\mathbf{V}) matrices respectively, where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L \times d}$. The attention operation is conducted as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(\mathbf{Q}\mathbf{K}^T / \sqrt{d})\mathbf{V}. \quad (2)$$

For multi-head self-attention (MHSA), H self-attention modules are applied to \mathbf{X} separately, and each of them produces an output sequence.

Token-level Sparsity Previous works (Rao et al. 2021; Kong et al. 2022) suggests that pruning less informative tokens, such as the background, has little impact on the accuracy. Thus, we can improve training efficiency at the token level by eliminating redundant tokens. However, these works require additional token selector module to evaluate the importance of each token. Moreover, they target fine-tuning, which is incompatible with training from scratch, as additional losses need to be introduced. In this work, we aim to prune tokens without introducing additional modules and train ViTs from scratch with original training recipes.

In Eq. 2, \mathbf{Q} and \mathbf{K} can be regarded as a concatenation of L token vectors: $\mathbf{Q} = [q_1, q_2, \dots, q_L]^T$ and $\mathbf{K} = [k_1, k_2, \dots, k_L]^T$. For the i -th token, the attention probability $\mathbf{a}_i = \text{Softmax}(q_i \cdot \mathbf{K}^T / \sqrt{d}) \in \mathbb{R}^L$ shows the degree of correlation of each key k_i with the query q_i . The output $\mathbf{a}_i \cdot \mathbf{V}$ can be considered as a linear combination of all value vectors, with \mathbf{a}_i being the combination coefficients. Thus, we can assume that \mathbf{a}_i indicates the importance score of all tokens. Typically, a large attention value suggests that the corresponded token is important to the query token i .

For ViT, the final output only depends on the [CLS] token. Thus, the attention map of this special tokens $\mathbf{a}_{\text{cls}} = \text{Softmax}(q_{\text{cls}} \cdot \mathbf{K}^T / \sqrt{d})$ represents the extent to which the token contributes to the final result. To this end, we utilize $\tilde{\mathbf{a}}_{\text{cls}} \in \mathbb{R}^N$, which excludes the first element of \mathbf{a}_{cls} , as the criterion to select tokens. In MHSA, we use the average attention probability of all heads $\bar{\mathbf{a}}_{\text{cls}} = \frac{1}{H} \sum_{i=1}^H \tilde{\mathbf{a}}_{\text{cls}}^{(h)}$. We select

$K(K < N)$ most important patch tokens based on the value of \mathbf{a}_{cls} and define $R_T = \frac{K}{N}$ as the token keep ratio. Thus, only $1 + R_T N$ tokens are left in the following layers.

For ViT variants without the [CLS] token, we calculate the importance scores of tokens by summing each column of the attention map (Wang, Zhang, and Han 2021). In detail, we use the cumulative attention probability $\sum_i \mathbf{a}_i \in \mathbb{R}^N$ to select informative tokens. We hypothesize that tokens that are important to many query tokens should be informative.

Attention-level Sparsity Since attention maps are usually sparse, dense attention is unnecessary. Thus, we also introduce sparsity at attention level by pruning attention connections. In detail, given a query token, we only calculate its attention connections with a few selected tokens. Previous ViT variants usually utilize some hand-craft predetermined sparse patterns, such as a neighborhood window (Liu et al. 2021) or dilated sliding window (Beltagy, Peters, and Cohan 2020). However, objects of different sizes and shapes may require different attention connections. Thus, these data-agnostic method is limited. Some input-dependent methods apply the deformable attention (Zhu et al. 2020) to find corresponding tokens. However, these works also require additional modules to learn the selected tokens.

In this work, we utilize the attention vector of each patch token to decide critical connections. Thus, no additional modules are introduced. In detail, given the image patch token \mathbf{P}_i and its corresponding query q_i , we use $\bar{\mathbf{a}}_i$ as the saliency criteria. In detail, we take the index of $R_A N$ tokens with the largest value of $\bar{\mathbf{a}}_i$, where $R_A \ll 1$ is the attention keep ratio. Thus, each patch token \mathbf{P}_i only need to calculate the attention score with the selected $R_A N$ tokens for the rest layers.

Token&Attention Selector (TA-Selector) We further combine the above token selector and attention connections selector into a single module, named *TA-Selector*. As shown in the right part of Figure 2, the TA-Selector is inserted after the self-attention module of one transformer encoder layer. Given the attention matrix $\mathbf{A} = \text{Softmax}(\mathbf{Q}\mathbf{K}^T/\sqrt{d}) \in \mathbb{R}^{L \times L}$, we first utilize the attention probability of [CLS] tokens \mathbf{a}_{cls} (the blue row of Figure 2) to locate $R_T N$ most informative tokens. We denote the index of selected tokens as set \mathcal{T} . We then extract the attention map of select tokens $\mathbf{A}_{\mathcal{T}}$, whose element is defined by $\mathbf{A}[i, j]$ with $\{i, j\} \in \mathcal{T} \times \mathcal{T}$ (the ticked red cell). Then we perform the attention-selector on $\mathbf{A}_{\mathcal{T}}$. In summary, $1 + R_T N$ tokens are left after the TA-selector, and each token only calculates the attention with $R_T R_A N$ tokens in the following transformer encoder layers.

With TA-Selector, we dynamically prune both tokens and attention connections at the same time. As the attention map at early layer may be inaccurate (Xu et al. 2021), we follow a hierarchical pruning scheme, which incorporate the TA-Selector between transformer encoder layers several times. Thus, we gradually prune tokens and attention connections with the network going deeper.

Sparsity Determination of Each Level

We focus on exploring the maximum compression sparsity (ratio) for each level while preserving accuracy. We leverage

a hierarchical data redundancy reduction scheme by setting the sparsity one by one in order: example → token → attention. This is based on two reasons: the characteristics of each level and the training efficiency. The example level is relatively independent, there is no sparsity trade-off between the example level and the TA level. Example level sparsity ratio contributes directly and the most to the time saving, so it should be considered first. After that, token and attention. When performing token-attention, we also prioritize pruning tokens over attention. There are two reasons: 1). Token contributes more speed-up compared to attention even with a small keep ratio (Set the attention keep ratio R_A to 0.2 with 4.8% speed up in Table 4). 2). Pruning attention is less sensitive to accuracy, which means it can be used to squeeze out the redundancy from a finer perspective after performing example and token sparsity. There are more complex ways to determine the sparsity, such as by adding constraints during training, which may add additional computation costs to training. We first run each level with a different keep ratio to see the performance degradation trend (0.9, 0.8, 0.7, etc.). Then, these results can be considered as a look-up table to guide the decision of Tri-level sparsity.

Experiments

Datasets and Implementation Details

Our experiments are conducted on ImageNet-1K (Deng et al. 2009), with approximately 1.2 million images. We report the accuracy on the validation set with 50k images. The image resolution is 224×224 . We also report results on Imagenet-V2 matched frequency (Recht et al. 2019) to control overfitting. We apply our framework to different backbones including DeiT (Touvron et al. 2021), Swin (Liu et al. 2021), PiT (Heo et al. 2021), and LV-ViT (Jiang et al. 2021) with the corresponding settings. In detail, the network is trained for 300 epochs on 4 NVIDIA V100 GPUs and is optimized by AdamW (Loshchilov and Hutter 2019) with weight decay 0.05. The batch size is set to 512, The learning rate is set to 5×10^{-4} initially, and is decayed with a cosine annealing schedule. For example level sparsity, we apply the remove-and-restore approach every 30 epochs with 5% data removal for each iteration. Since this approach takes extra time and the accuracy decreases slightly after each iteration, we only iterate the number of times in which all the removed data can be covered. Hence, we set the different number of iterations for different keep rates. For 10% removed data, we iterate 3 times (30th, 60th, 90th epoch). For 20% removed data, we iterate 5 times (30th, 60th, 90th, 120th, 150th epoch). For token and attention level sparsity, we insert our TA-Selector before the 4th, 7th, 10th layer for hierarchical pruning scheme as in (Rao et al. 2021). Besides, during the training process, we adopt a warm-up strategy for the TA-Selector. The token keep ratio R_T starts from 1.0 and is gradually reduced to the given R_T with a cosine annealing schedule. For simplicity and fair comparison, we set keep ratio to 0.7, 0.8, and 0.9 in our main experiments, denoting as Tri-Level/0.7, Tri-Level/0.8 and Tri-Level/0.9, respectively.

METHOD	TRAINING TIME REDUCED	MACs (G) SAVING	1K ACC(%)	V2 ACC(%)
DeiT-T				
DENSE	-	-	72.2	65.0
DYNAMICViT/0.7 (RAO ET AL. 2021)	0%	30.0%	71.8 (-0.4)	64.6
S ² VITE (CHEN ET AL. 2021B)	10.6%	23.7%	70.1 (-2.1)	-
TRI-LEVEL/0.9	15.2%	17.4%	72.6 (+0.4)	65.9
TRI-LEVEL/0.8	29.4%	31.9%	72.1 (-0.1)	65.0
TRI-LEVEL/0.7	35.6%	38.5%	71.9 (-0.3)	64.6
DeiT-S				
DENSE	-	-	79.8	73.6
IA-RED ² (PAN ET AL. 2021)	0%	32.0%	79.1 (-0.7)	-
DYNAMICViT/0.7 (RAO ET AL. 2021)	0%	36.7%	79.3 (-0.5)	72.8
UVC (YU ET AL. 2022B)	0%	42.4%	79.4 (-0.4)	-
DYNAMICViT/0.7* (RAO ET AL. 2021)	7.2%	34.7%	77.6 (-2.2)	71.1
EViT/0.8 (LIANG ET AL. 2022)	6.0%	13.0%	79.8 (-0.1)	73.4
EViT/0.7 (LIANG ET AL. 2022)	20.0%	35.0%	79.5 (-0.3)	73.2
EViT/0.6 (LIANG ET AL. 2022)	25.0%	43.0%	78.9 (-0.9)	72.3
S ² VITE (CHEN ET AL. 2021B)	22.7%	31.6%	79.2 (-0.6)	-
TRI-LEVEL/0.9	15.7%	17.0%	79.9 (+0.1)	73.8
TRI-LEVEL/0.8	29.8%	31.3%	79.5 (-0.3)	73.3
TRI-LEVEL/0.7	34.3%	37.6%	79.3 (-0.5)	72.9
DeiT-B				
DENSE	-	-	81.8	75.8
IA-RED ² (PAN ET AL. 2021)	0%	33.0%	80.3 (-1.5)	-
DYNAMICViT/0.7 (RAO ET AL. 2021)	0%	37.4%	80.7 (-1.1)	74.2
EViT/0.7 (LIANG ET AL. 2022)	15.2%	35.0%	81.3 (-0.5)	75.1
TRI-LEVEL/0.9	10.2%	15.6%	81.6 (-0.2)	75.3
TRI-LEVEL/0.8	23.7%	29.1%	81.3 (-0.5)	75.1
SWIN-T				
DENSE	-	-	81.2	75.1
DYNAMICSWIN/0.9 (RAO ET AL. 2022)	0%	7%	81.0 (-0.2)	75.0
TRI-LEVEL/0.9	13.1%	15.6%	81.2 (-0.0)	75.1
TRI-LEVEL/0.8	21.1%	25.3%	81.0 (-0.1)	74.9
SWIN-S				
DENSE	-	-	83.2	76.9
DYNAMICSWIN/0.7 (RAO ET AL. 2022)	0%	21%	83.2 (-0.0)	76.9
WDPRUNING (YU ET AL. 2022A)	0%	12.6%	82.4 (-0.6)	76.1
TRI-LEVEL/0.8	20.2%	24.1%	83.2 (-0.0)	76.8

Table 1: The training time reduced, MACs (G) saving, and Top-1 accuracy comparison on the ImageNet-1K and Imagenet-V2 matched frequency. “*” refers to our reproduced results of DynamicViT training from scratch.

Experimental Results

We report the Top-1 accuracy of each model with different keep ratio. For efficient training metrics, we evaluate the reduced running time and MACs (G) saving of the entire training process. Our main results are shown in Table 1. Remarkably, our efficient training method can even improve the ViT accuracy rather than compromising it (15.2% time reduction with 72.6% Top-1 accuracy on Deit-T, and 15.7% time reduction with 79.9% Top-1 accuracy on Deit-S). This demonstrates the existence of data redundancy in ViT.

We also compare our method with other efficient ViTs aiming at reducing redundancy. Most of the existing efforts target only efficient inference. They either apply to the fine-tuning phase (Rao et al. 2021; Pan et al. 2021) or introduce additional modules with additional training cost (Chen et al. 2021b; Yu et al. 2022b). In contrast to others, we can accelerate both the training and inference process. Our method significantly reduces the training time while restricting the accuracy drop in a relatively small range. Notably, we are able to reduce the training time by 35.6% for DieiT-T with a negligible 0.3% accuracy degradation, and 4.3% for DieiT-S with a 0.5% decrease in accuracy, which outperforms existing pruning methods in terms of accuracy and efficiency.

Generalization on other Architectures and Tasks

To verify the generalizability of our approach, we conduct experiments with other vision transformer models and evaluate

Backbone	AP ^b	AP ^b ₅₀	AP ^b ₇₅	AP ^m	AP ^m ₅₀	AP ^m ₇₅
ResNet50 (He et al. 2016)	41.0	61.7	44.9	37.1	58.4	40.1
PVT-S (Wang et al. 2021)	43.0	65.3	46.9	39.9	62.5	42.8
Swin-T	46.0	68.1	50.3	41.6	65.1	44.9
Tri-level	46.0	68.0	50.3	41.6	65.2	44.9

Table 2: Results on COCO object detection and instance segmentation.

them on object detection and instance segmentation tasks.

Swin Transformer We further apply our method to other ViT variants, such as Swin Transformer (Liu et al. 2021). Instead of using the [CLS] token, it applies a global average pooling on features of all tokens. Thus, we use the cumulative attention probability to select tokens. For the example level sparsity, we also replace the variance of [CLS] token’s attention map with that of the cumulative attention map. Results are shown in Table 1. We are able to reduce the training time by 13.1% for Swin-T and 20.2% for Swin-S with no accuracy drop. Thus, our method can generalize to other ViT architectures. More results for other architectures such as LV-ViT (Jiang et al. 2021) and other datasets such as ImageNet-Real and CIFAR can be found in the Appendix.

Object detection and Instance Segmentation Experiments are conducted on COCO 2017 (Lin et al. 2014), which contains 118K training, 5K validation and 20K test-dev images. As shown in Table 2, we use the Mask R-CNN framework and replace different backbones. FLOPs are measured at 800 × 1280 resolution. Our Tri-level model denotes the 13.1% training time reduction model in Table 1.

Deployment on Edge Devices

We also evaluate our method on an embedded FPGA platform, namely, Xilinx ZCU102. The platform features a Zynq UltraScale + MPSoC device (ZU9EG), which contains embedded ARM CPUs and 274k LUTs, 2520 DSPs, and 32.1Mb BRAMs on the programmable logic fabric. The working frequency is set to 150MHz for all the designs implemented through Xilinx Vitis and Vitis HLS 2021.1. The 16-bit fixed-point precision is adopted to represent all the model parameters and activation data. We measure both on device training latency and inference latency. In detail, the training latency is measured on FPGA using a prototype implementation that supports layer-wise forward and backward computations. We report the average per epoch training time. Both inference and training use the batch size of 1. As in Table 3, our method can also accelerate the training and inference on edge devices.

Ablation Study

Sub-method Effectiveness

We conduct ablation studies to evaluate the contribution of each sparse method separately. The results of DeiT-S with different keep ratios at each sparse level are shown in Table 4. Removing redundant data in each sub-method can even lead to improvements in accuracy while reducing training time. At the same keep ratio, the example level sparsity contributes the most to the training time reduction, as it directly removes the

Method	MACs (G)	Training	Inference
	Saving	Latency (hour)	Latency (ms)
DeiT-T			
Dense	-	13.79	10.91
Tri-Level/0.9	17.4%	11.49	9.68
Tri-Level/0.8	31.9%	9.61	8.50
Tri-Level/0.7	38.5%	8.91	7.36
DeiT-S			
Dense	-	32.44	25.68
Tri-Level/0.9	17.0%	27.28	22.92
Tri-Level/0.8	31.3%	22.55	20.25
Tri-Level/0.7	37.6%	21.15	17.61

Table 3: The inference latency and the average per epoch training latency on Xilinx ZCU102 FPGA board.

Level	Method	Keep Ratio	Training Time Reduced	1K Acc. (%)	V2 Acc. (%)
Example	Dense	1.0	-	79.8	73.6
	Random	0.8	19.9%	78.4	71.7
	Active Bias	0.8	4.3%	79.1	72.5
	Forgetting	0.8	16.1%	79.0	72.4
	ours	0.9	9.9%	80.1	73.8
	ours	0.8	19.8%	79.7	73.5
	ours	0.7	29.8%	79.2	72.6
Token	Static	0.7	20%	75.2	68.3
	DynamicViT	0.7	0%	79.3	72.8
	IA-RED ²	0.7	0%	79.1	-
	Evo-ViT	0.7	13.0%	79.4	-
	ours	0.9	6.2%	79.9	73.8
	ours	0.8	13.3%	79.7	73.4
	ours	0.7	20.6%	79.4	73.2
Attention	Magnitude	0.5	2.8%	78.9	72.3
	Longformer	0.2	4.8%	78.8	72.0
	ours	0.4	3.3%	79.8	73.5
	ours	0.3	3.6%	79.7	73.5
	ours	0.2	4.8%	79.7	73.4

Table 4: Sub-method effectiveness on DeiT-S under different keep ratios. We also compare with existing individual example/token/attention sparsity methods.

entire image. Thus, it can be considered the coarse-grained level of data reduction. Meanwhile, the attention level has less reduction in training time even at low keep ratios because it only contributes to the matrix multiplication of the MHSA module. Thus, it can be considered a fine-grained level of data reduction.

Comparison of Different Methods

To verify the advantages of our method, we compare it with other data sparsity strategies on DeiT-S. As shown in Table 4, *Random* denotes randomly updating the training data with the same frequency and number of updates as our method. Under the same keep ratio, it shows a significant accuracy degradation (78.4% vs. 79.7%). We also compare it with the existing work (Toneva et al. 2018), which conducts a two-phase method for example evaluation and removal. The accuracy is lower (79.0% vs. 79.7%) and the training acceleration is limited (16.1% vs. 19.8%).

Visualization

Example Evaluation. We visualize both of the most forgotten (hardest) training examples and the unforgettable (easiest)

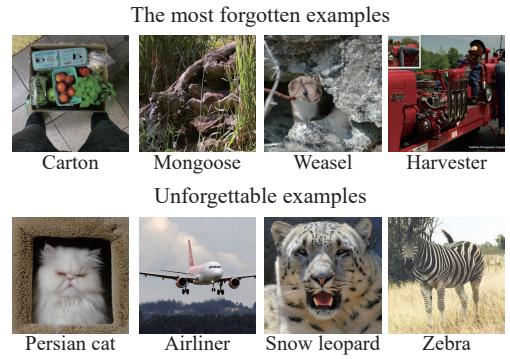


Figure 3: Visualization of the most forgotten examples and unforgettable examples.

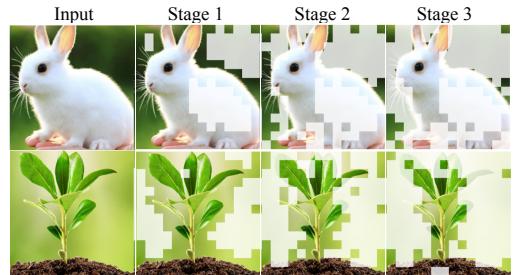


Figure 4: Visualization of token pruning. The masked patches represent the less informative tokens that are pruned.

est) training examples obtained by our proposed online example filtering method. As shown in Figure 3, the hardest examples generally contain more sophisticated content in the image, making them more informative for training. On the contrary, the less informative examples are usually easier to be recognized, and they contain conspicuous characteristics of their categories. This observation is similar to the previous works (Toneva et al. 2018; Yuan et al. 2021a), while our online example filtering method can take the advantage of data efficient training throughout the entire training process.

Token Selection. We also visualize the tokens identified by our attention-based token selector in Figure 4. Note that the network can gradually remove the background tokens, which is less informative. Eliminating these redundant tokens can reduce the computation at both training and inference time. Moreover, the remaining tokens maintain class-specific attributes, which is helpful for recognition.

Conclusion

In this work, we introduce sparsity into data and propose an end-to-end efficient training framework to accelerate ViT training and inference. We leverage a hierarchical data redundancy reduction scheme, by exploring the sparsity of the number of training examples in the dataset, the number of patches (tokens) in each input image, and the number of connections between tokens in attention weights. Comprehensive experiments validate the effectiveness of our method. Future work includes extending our framework to other aspects such as weight redundancy.

References

- Amini, A.; Periyasamy, A. S.; and Behnke, S. 2021. T6D-Direct: Transformers for Multi-Object 6D Pose Direct Regression. *arXiv preprint arXiv:2109.10948* .
- Bao, H.; Dong, L.; and Wei, F. 2021. BEiT: BERT Pre-Training of Image Transformers. *arXiv preprint arXiv:2106.08254* .
- Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* .
- Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-end object detection with transformers. In *European Conference on Computer Vision*, 213–229. Springer.
- Chang, H.-S.; Learned-Miller, E.; and McCallum, A. 2017. Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems* 30.
- Chen, C.-F. R.; Fan, Q.; and Panda, R. 2021. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 357–366.
- Chen, H.; Wang, Y.; Guo, T.; Xu, C.; Deng, Y.; Liu, Z.; Ma, S.; Xu, C.; Xu, C.; and Gao, W. 2021a. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12299–12310.
- Chen, T.; Cheng, Y.; Gan, Z.; Yuan, L.; Zhang, L.; and Wang, Z. 2021b. Chasing Sparsity in Vision Transformers: An End-to-End Exploration. *arXiv preprint arXiv:2106.04533* .
- Chen, T.; Saxena, S.; Li, L.; Fleet, D. J.; and Hinton, G. 2021c. Pix2seq: A Language Modeling Framework for Object Detection. *arXiv preprint arXiv:2109.10852* .
- Chen, X.; Yan, B.; Zhu, J.; Wang, D.; Yang, X.; and Lu, H. 2021d. Transformer tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8126–8135.
- Dai, Z.; Cai, B.; Lin, Y.; and Chen, J. 2021. Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1601–1610.
- Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.
- Deng, J.; Yang, Z.; Chen, T.; Zhou, W.; and Li, H. 2021. TransVG: End-to-End Visual Grounding With Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 1769–1779.
- Dong, X.; Bao, J.; Zhang, T.; Chen, D.; Zhang, W.; Yuan, L.; Chen, D.; Wen, F.; and Yu, N. 2021. PeCo: Perceptual Codebook for BERT Pre-training of Vision Transformers. *arXiv preprint arXiv:2111.12710* .
- Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* .
- El-Nouby, A.; Neverova, N.; Laptev, I.; and Jégou, H. 2021. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644* .
- Guo, M.-H.; Cai, J.-X.; Liu, Z.-N.; Mu, T.-J.; Martin, R. R.; and Hu, S.-M. 2021. PCT: Point cloud transformer. *Computational Visual Media* 7(2): 187–199. ISSN 2096-0662. doi:10.1007/s41095-021-0229-5. URL <http://dx.doi.org/10.1007/s41095-021-0229-5>.
- He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. 2021. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377* .
- He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.
- Heo, B.; Yun, S.; Han, D.; Chun, S.; Choe, J.; and Oh, S. J. 2021. Rethinking Spatial Dimensions of Vision Transformers. In *International Conference on Computer Vision (ICCV)*.
- Hudson, D. A.; and Zitnick, C. L. 2021. Generative Adversarial Transformers. *Proceedings of the 38th International Conference on Machine Learning, ICML 2021* .
- Jiang, Z.-H.; Hou, Q.; Yuan, L.; Zhou, D.; Shi, Y.; Jin, X.; Wang, A.; and Feng, J. 2021. All Tokens Matter: Token Labeling for Training Better Vision Transformers. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 18590–18602. Curran Associates, Inc. URL <https://proceedings.neurips.cc/paper/2021/file/9a49a25d845a483fae4be7e341368e36-Paper.pdf>.
- Katharopoulos, A.; and Fleuret, F. 2018. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, 2525–2534. PMLR.
- Kim, B.; Lee, J.; Kang, J.; Kim, E.-S.; and Kim, H. J. 2021. HOTR: End-to-End Human-Object Interaction Detection with Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 74–83.
- Kong, Z.; Dong, P.; Ma, X.; Meng, X.; Niu, W.; Sun, M.; Ren, B.; Qin, M.; Tang, H.; and Wang, Y. 2022. SPViT: Enabling Faster Vision Transformers via Soft Token Pruning. *ECCV* .
- Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images .
- Liang, J.; Cao, J.; Sun, G.; Zhang, K.; Van Gool, L.; and Timofte, R. 2021. SwinIR: Image Restoration Using Swin Transformer. *arXiv preprint arXiv:2108.10257* .
- Liang, Y.; GE, C.; Tong, Z.; Song, Y.; Wang, J.; and Xie, P. 2022. EViT: Expediting Vision Transformers via Token Reorganizations. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=BjywvnXXVn_.

- Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer.
- Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022.
- Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*. URL <https://openreview.net/forum?id=Bkg6RiCqY7>.
- Lu, Z.; Liu, H.; Li, J.; and Zhang, L. 2021. Efficient Transformer for Single Image Super-Resolution. *arXiv preprint arXiv:2108.11084*.
- Misra, I.; Girdhar, R.; and Joulin, A. 2021. An End-to-End Transformer Model for 3D Object Detection. In *ICCV*.
- Nilsback, M.-E.; and Zisserman, A. 2008. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 722–729. IEEE.
- Pan, B.; Jiang, Y.; Panda, R.; Wang, Z.; Feris, R.; and Oliva, A. 2021. IA-RED²: Interpretability-Aware Redundancy Reduction for Vision Transformers. In *Advances in Neural Information Processing Systems*.
- Paul, M. e. a. 2021. Deep Learning on a Data Diet: Finding Important Examples Early in Training. In *Advances in Neural Information Processing Systems*.
- Rao, Y.; Liu, Z.; Zhao, W.; Zhou, J.; and Lu, J. 2022. Dynamic Spatial Sparsification for Efficient Vision Transformers and Convolutional Neural Networks. *arXiv preprint arXiv:2207.01580*.
- Rao, Y.; Zhao, W.; Liu, B.; Lu, J.; Zhou, J.; and Hsieh, C.-J. 2021. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. *arXiv preprint arXiv:2106.02034*.
- Recht, B.; Roelofs, R.; Schmidt, L.; and Shankar, V. 2019. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, 5389–5400. PMLR.
- Ryoo, M. S.; Piergiovanni, A.; Arnab, A.; Dehghani, M.; and Angelova, A. 2021. TokenLearner: What Can 8 Learned Tokens Do for Images and Videos? In *Advances in Neural Information Processing Systems*.
- Srinivas, A.; Lin, T.-Y.; Parmar, N.; Shlens, J.; Abbeel, P.; and Vaswani, A. 2021. Bottleneck transformers for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16519–16529.
- Tolstikhin, I.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Keysers, D.; Uszkoreit, J.; Lucic, M.; et al. 2021. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*.
- Toneva, M.; Sordoni, A.; Combes, R. T. d.; Trischler, A.; Bengio, Y.; and Gordon, G. J. 2018. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*.
- Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and J’egou, H. 2021. Training data-efficient image transformers & distillation through attention. In *ICML*.
- Trockman, A.; and Kolter, J. Z. 2022. Patches Are All You Need? *arXiv preprint arXiv:2201.09792*.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.
- Wang, H.; Zhang, Z.; and Han, S. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 97–110. IEEE.
- Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; and Shao, L. 2021. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.
- Xia, Z.; Pan, X.; Song, S.; Li, L. E.; and Huang, G. 2022. Vision Transformer with Deformable Attention. doi:10.48550/ARXIV.2201.00520.
- Xu, Y.; Zhang, Z.; Zhang, M.; Sheng, K.; Li, K.; Dong, W.; Zhang, L.; Xu, C.; and Sun, X. 2021. Evo-ViT: Slow-Fast Token Evolution for Dynamic Vision Transformer. *arXiv preprint arXiv:2108.01390*.
- Xue, F.; Wang, Q.; and Guo, G. 2021. TransFER: Learning Relation-aware Facial Expression Representations with Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3601–3610.
- Yang, F.; Yang, H.; Fu, J.; Lu, H.; and Guo, B. 2020. Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5791–5800.
- Yu, F.; Huang, K.; Wang, M.; Cheng, Y.; Chu, W.; and Cui, L. 2022a. Width & Depth Pruning for Vision Transformers. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 2022.
- Yu, S.; Chen, T.; Shen, J.; Yuan, H.; Tan, J.; Yang, S.; Liu, J.; and Wang, Z. 2022b. Unified Visual Transformer Compression. In *International Conference on Learning Representations*. URL <https://openreview.net/forum?id=9jsZiUgkCZP>.
- Yuan, G.; Ma, X.; Niu, W.; Li, Z.; Kong, Z.; Liu, N.; Gong, Y.; Zhan, Z.; He, C.; Jin, Q.; et al. 2021a. MEST: Accurate and Fast Memory-Economic Sparse Training Framework on the Edge. *Advances in Neural Information Processing Systems* 34.
- Yuan, L.; Chen, Y.; Wang, T.; Yu, W.; Shi, Y.; Jiang, Z.-H.; Tay, F. E.; Feng, J.; and Yan, S. 2021b. Tokens-to-Token ViT: Training Vision Transformers From Scratch on ImageNet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 558–567.

Zhao, H.; Jiang, L.; Jia, J.; Torr, P. H.; and Koltun, V. 2021.
Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 16259–16268.

Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; and Dai, J. 2020.
Deformable detr: Deformable transformers for end-to-end
object detection. *arXiv preprint arXiv:2010.04159* .

Appendix

Definition of Example Forgetting

Some definitions for the example forgetting approach applied in our example selector are as follows:

- **Forgetting Events:** A training example learned in the previous training process is being mis-classified in latter training process, in other words, forgotten by the network.
- **Learning Events:** A training example mis-classified in the previous training process is being classified correctly in the latter training process.
- **Unforgettable Examples:** A training example after learned by the network at a certain step will not be mis-classified (forgotten) during the rest of the training course. According to prior work (Toneva et al. 2018), the unforgettable examples are generally considered easy to be learned and less informative than others, so they can be removed based on the order of the number of forgotten events. Networks trained on the sub-dataset can still maintain generalization performance compared to networks trained on the full dataset.

We choose the forgetting score as an evaluation metric over other methods for the following reasons:

- **Better fit for our online examples filtering method:** The forgetting approach counts the number of times an example is correctly classified or not (forgotten). Examples that have 0 or low forgetting counts are defined as simple examples, and thus can be removed. Therefore, it is a direct way to guide removing examples.
- **Better for efficient training:** Our method does not involve reweighting all the examples (Chang, Learned-Miller, and McCallum 2017) or having interaction with the loss (Katharopoulos and Fleuret 2018). Our evaluation method is relatively independent of training, by counting the output logits misclassified times. So it can easily be applied to various models with different training schemes to improve training efficiency.
- **More suitable for combining with ViT:** The forgetting approach uses the output logit of the model. It aligns with calculating the variance of the attention map of the [CLS] token or the sum of each column of the attention map in our proposed method.

Explore Example Level Sparsity

Different Update Frequency

We experimented with the update frequency and number of updates for the data, and the results are shown in Table 5. We keep the same training epochs as DeiT, and train with 80% of the dataset. We apply the remove-and-restore approach with 5% data during each update. We first set the update frequency to once every 30 epochs, with a total of 10 times. However, the accuracy is only 78.7%. This is because when the data is updated, the training accuracy will descend due to the introduction of new data, and it takes 5-10 epochs to recover the accuracy. Therefore, the accuracy is low under the restriction of 300 epochs. We reduce the number of updates to only 5 updates, and no updates after 150 epochs to reduce the loss from updates. Since 20% of the data is removed and

Method	Keep Ratio	Top-1 Acc.(%)	Top-5 Acc.(%)
No update	0.8	78.9	94.4
30 epoch 10 times	0.8	78.7	94.3
30 epoch 5 times	0.8	79.7	94.8
40 epoch 5 times	0.8	78.6	94.3
40 epoch 4 times	0.8	79.0	94.5

Table 5: Different update frequency and number of updates for the example level on DeiT-S.

5% is updated each time, 5 times is enough to iterate 20% of the data. In addition, the training will converge faster because the quality of data becomes better. The latter 150 epochs can be regarded as a phase of retrain. Under this setting, the accuracy reaches 79.7%.

Shuffle vs. Non-Shuffle

As mentioned in the main paper, we remove a portion of examples from the dataset prior to training. Thus, when the dataset is updated during training, the newly removed data will be put into the existing pool of removed examples, and then the restored data will be retrieved from here as well. We found that shuffling the data in the pool before the restore operation can effectively improve the training accuracy. As shown in Table 6, shuffling the data before restoring can result in a 0.3% accuracy increase compared to not shuffling (79.7% vs. 79.4%). Under the 0.9 keep ratio, our proposed example level sparsity method can even improve the DeiT-S accuracy by 0.3% rather than compromising it.

Class Balance vs. Class Imbalance

When we perform remove-and-restore operations on the examples to update the training sub-dataset, there are two ways:
(i) Class balance approach, which removes the same percentage of data for each class according to the forgetting events.
(ii) Class imbalance approach, which calculates the forgetting events uniformly for all the examples, and then removes a certain percentage of the least forgettable data, regardless of the class label of the example. The results of both approaches are shown in Table 6. Class imbalance outperforms class balance method by 1.0% (79.4% vs. 78.4%) under 0.8 keep ratio on DeiT-S. This is due to the fact that different classes in the dataset have different levels of difficulty in training. If all classes remove data using the same ratio, then for difficult classes, forgettable examples may be removed, causing a decrease in training accuracy.

[CLS] Attention Map Selection

For ViTs with [CLS] tokens, we choose the attention map of [CLS] from different layers to evaluate the complexity of the images, and conduct experiments for both example level and Tri-level. As shown in Table 7, when running only the example level, using the attention map of the last layer can achieve the highest accuracy, as it is the most accurate for evaluating the importance of the token. When running

METHOD	KEEP RATIO	TOP-1 ACC.(%)	TOP-5 ACC.(%)
FULL DATA	1.0	79.8	94.9
CLASS BALANCE	0.8	78.4	94.2
IMBALANCE	0.8	79.4	94.7
IMBALANCE + SHUFFLE	0.8	79.7	94.8
IMBALANCE + SHUFFLE	0.9	80.1	95.0

Table 6: Comparison of different example level sparsity strategy on DeiT-S.

Tri-level, the [CLS] token at the last layer contains less information due to the combination of token and attention level sparsity. Therefore, we choose the attention map of the third layer, which is the layer before token pruning to feed into our example selector. The accuracy is higher than using the last layer (72.6% vs. 72.4%).

METHOD	LAYER NUMBER	TOP-1 ACC.(%)	TOP-5 ACC.(%)
EXAMPLE LEVEL	w/o [CLS]	72.4	91.4
	3	72.8	91.5
	12	73.0	91.5
TRI-LEVEL	w/o [CLS]	72.2	91.1
	3	72.6	91.5
	12	72.4	91.4

Table 7: Comparison of different attention map selection on DeiT-T at 0.9 keep ratio.

Generalization on others Architectures and Datasets

Implementation Details

We describe the token pruning locations for Swin (Liu et al. 2021), (Jiang et al. 2021) and PiT (Heo et al. 2021). Swin has 4 stages, the configurations are: Swin-T: C = 96, layer numbers = {2, 2, 6, 2}; Swin-S: C = 96, layer numbers = {2, 2, 18, 2} The Patch Merging layer between each stage makes the token non-transitive through stages. Also, the 1,2, and 4 stage has only one pair of W-MSA and SW-MSA transformers, making it hard to prune tokens. Therefore, we perform token pruning on stage 3, with the most number of layers. In side stage 3, we prune token after the {1,2,3} layer for Swin-T, and {2,4,6} layer for Swin-S, for fair comparison against (Rao et al. 2022). For LV-ViT-S, we insert the token selector after the {4, 8, 12} layers. For LViViT-M, we insert the token selector after the {5, 10, 15} layers. For PiT-XS/S, we insert the token selector after the {1, 5, 10} layers, for fair comparison against (Rao et al. 2021; Kong et al. 2022; Liang et al. 2022).

Results on Other Datasets

We train on four datasets on DeiT-S: Imagenet Real (Deng et al. 2009), CIFAR-10 (Krizhevsky, Hinton et al. 2009), CIFAR-100 (Krizhevsky, Hinton et al. 2009), and Flower (Nilsback and Zisserman 2008) under the same hyperparameters. We evaluate the accuracy under different keep ratios: 1.0,

KEEP RATIO	REAL (%)	CIFAR-10 (%)	CIFAR-100 (%)	FLOWERS (%)
1.0	85.7	98.0	87.1	98.8
0.7	85.3±0.1 (-0.4)	97.9±0.0 (-0.1)	87.0±0.1 (-0.1)	98.8±0.0 (-0.0)
0.5	84.2±0.2 (-1.5)	97.2±0.0 (-0.8)	86.1±0.1 (-1.0)	98.7±0.0 (-0.1)

Table 8: Example-level sparsity of different datasets on DeiT-S.

0.7, and 0.5. Results are shown in Table 8. All the standard deviation of each sub-method is about $-0.2 \sim +0.2$

Results on Other ViT Variants

We apply our proposed method to LV-ViT (Jiang et al. 2021) and PiT (Heo et al. 2021), which outperforms state-of-the-art compression methods in terms of training efficiency and accuracy, as shown in Table 9. We also show the results on the COCO dataset in Table 10. We use the same settings as the (Heo et al. 2021), where we replace the backbones of the Deformable DETR (Zhu et al. 2020) and reduce the image resolution for training. Our model can reduce 19.3% training time without sacrificing any performance on both classification and object detection tasks compared to PiT-S.

METHOD	TRAINING TIME REDUCED	MACS (G) SAVING	TOP-1 ACC(%)
PiT-XS			
DENSE SPViT (KONG ET AL. 2022)	-	-	78.1
TRI-LEVEL	0%	18.6%	78.0 (-0.1)
PiT-S			
DENSE SPViT (KONG ET AL. 2022)	-	-	80.9
TRI-LEVEL	24.1%	28.6%	80.9 (-0.0)
LV-ViT-S			
DENSE DYNAMICViT (RAO ET AL. 2021)	-	-	83.3
EViT (LIANG ET AL. 2022)	0%	30.2%	83.0 (-0.3)
TRI-LEVEL	15.0%	29.0%	83.0 (-0.3)
LV-ViT-M			
DENSE DYNAMICViT (RAO ET AL. 2021)	-	-	84.0
SPViT (KONG ET AL. 2022)	0%	33.0%	83.8 (-0.2)
TRI-LEVEL	0%	42.2%	83.7 (-0.3)
TRI-LEVEL			
20.5%	24.1%	83.8 (-0.2)	

Table 9: Results on LV-ViT and PiT. We show the reduced training time, MACs (G) saving, and Top-1 accuracy comparison on the ImageNet-1K dataset.

BACKBONE	Avg. PRECISION AT IOU AP	Avg. PRECISION AT IOU AP ₅₀	Avg. PRECISION AT IOU AP ₇₅
RESNET50 (HE ET AL. 2016)	41.5	60.5	44.3
ViT-S	36.9	57.0	38.0
PiT-S	39.4	58.8	41.5
TRI-LEVEL	39.4	58.8	41.4

Table 10: COCO detection performance based on Deformable DETR. We evaluate the performance of our method with ResNet50, ViT-S, and PiT-S as pretrained backbones for object detection.