Token Pooling in Vision Transformers for Image Classification

Dmitrii Marin†, Jen-Hao Rick Chang⋆, Anurag Ranjan⋆, Anish Prabhu⋆
Mohammad Rastegari⋆, Oncel Tuzel⋆
†University of Waterloo, ⋆Apple
jenhao_chang@apple.com

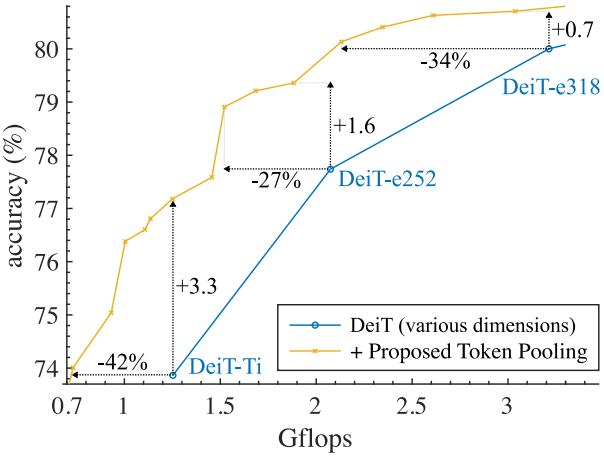
Abstract

*Pooling is commonly used to improve the computation-accuracy trade-off of convolutional networks. By aggregating neighboring feature values on the image grid, pooling layers downsample feature maps while maintaining accuracy. In standard vision transformers, however, tokens are processed individually and do not necessarily lie on regular grids. Utilizing pooling methods designed for image grids (*e.g.*, average pooling) thus can be sub-optimal for transformers, as shown by our experiments. In this paper, we propose Token Pooling to downsample token sets in vision transformers. We take a new perspective — instead of assuming tokens form a regular grid, we treat them as discrete (and irregular) samples of an implicit continuous signal. Given a target number of tokens, Token Pooling finds the set of tokens that best approximates the underlying continuous signal. We rigorously evaluate the proposed method on the standard transformer architecture (ViT/DeiT) and on the image classification problem using ImageNet-1k. Our experiments show that Token Pooling significantly improves the computation-accuracy trade-off without any further modifications to the architecture. Token Pooling enables DeiT-Ti to achieve the same top-1 accuracy while using 42% fewer computations.*

1. Introduction

Vision transformers [10, 14, 21, 43, 55] have demonstrated strong results comparable to Convolutional Neural Networks (CNN). However, the high computational cost limits their use in resource-restricted, real-time, or low-powered scenarios. In this paper, we aim to improve the computational efficiency of standard vision transformers on image classification by downsampling tokens, *i.e.*, removing redundant tokens across transformer layers.

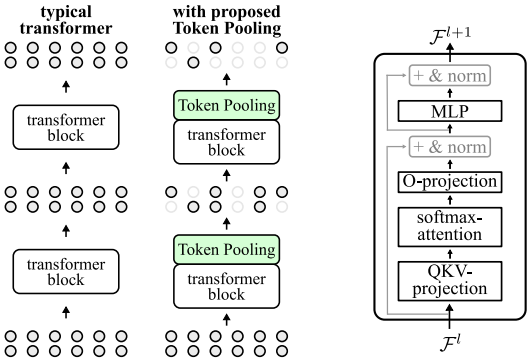
Downsampling is one of the main techniques used in CNNs to improve computational efficiency. In CNNs, features are explicitly designed to lie on regular xy-grids, due to the repeated use of convolution with localized receptive fields. The xy-grid structure allows downsampling techniques like max/average pooling to aggregate neighboring feature values on the grids and reduce grid dimensions. However, the xy-grid structure is not maintained in standard transformers. In transformers, the xy-location of a token is only hinted as feature values via positional encoding at the first (or a few) layer. Tokens are processed as a set by multi-



(a) Accuracy *vs.* computation

(b) Token Pooling via cluster analysis of token representations.

Figure 1: We propose Token Pooling, a novel token downsampling method, for standard vision transformers like ViT [10] and DeiT [43]. (a) Inserting Token Pooling layers into DeiT significantly improves the trade-off between accuracy and computation on ImageNet-1k. (b) Token Pooling treats tokens as discrete samples of a continuous signal. This motivates us to use cluster analysis to aggregate neighboring token information automatically. We show the input images and the token clusters at the 6-th layer of DeiT-S.



(a) Token Pooling (b) Transformer block

Figure 2: (a) We insert Token Pooling layer after every transformer block without modifying any of the architecture. This enables to study the effect solely caused due to the downsampling layer. (b) shows a transformer block.

head attention, and each of them is processed separately by Multi-Layer Perceptrons (MLP). In other words, in standard transformers like ViT/DeiT [10, 43][1], besides the first layer taking image patches as input, tokens do not lie on regular grids. We need a dedicated downsampling technique.

We propose *Token Pooling*, a downsampling operator for standard transformers, aiming to improve the computational efficiency on image classification problems. Motivated by nonuniform sampling and image compression [2, 23, 30, 44], we view tokens as discrete and irregular samples of a continuous signal and formulate token downsampling as an optimization problem that minimizes the reconstruction error of the underlying continuous signal from the downsampled tokens. Surprisingly, we show that the solution to this optimization problem can be easily found via clustering analysis like K-Means and K-Medoids (see Figure 1b).

We conduct thorough study of the proposed and various prior downsampling techniques, including random/importance sampling the tokens, score-based token pruning [13, 34], and the convolution-with-stride downsampling method that generalizes average pooling and is widely utilized in recent vision transformers [21, 25, 49]. We focus on studying the sole effect of the downsampling layers — we directly insert downsampling layers after each transformer block *without* optimizing any other architecture, *e.g.*, feature dimension and the number of layers, as shown in Figure 2a. Our results show that Token Pooling outperforms baseline downsampling techniques (Figure 4) and significantly improves the computation-accuracy trade-off of vanilla vision transformers like ViT (Figure 1a).

Contributions. The paper makes these contributions:

- We conduct an extensive study of prior downsampling techniques for visual transformers by comparing their computation-accuracy trade-offs.
- We analyze the computational cost of vision-transformer components and limitations of prior score-based downsampling methods. We show that softmax attention layers perform low-pass filtering on the tokens and thus produce redundant tokens by construction.
- We propose a novel token downsampling technique, Token Pooling, which achieves a significant improvement in the computation-accuracy trade-off on ImageNet-1k.

2. Related work

In this section, we introduce vision transformers, and review existing methods that improve the efficiency of transformers including existing token-downsampling methods.

2.1. Vision transformers

Vision transformers [10, 14, 21, 21, 25, 43] utilize the transformer architecture that is originally designed for Natural Language Processing (NLP) by Vaswani *et al*. [45] and further popularized by Radford *et al*. [31] and Devlin *et al*. [9]. A standard vision transformer is a composition of L transformer blocks that take a set of input tokens and return another set of tokens. The input tokens of the first transformer block are features representing image patches. In standard transformer architecture [45], the number of tokens remains the same across the whole network. To perform classification, a separate classification token is inserted to estimate the probabilities of individual classes.

Let the set of N tokens at depth l be $\mathcal{F}^l = \{\boldsymbol{f}_0^l, \ldots, \boldsymbol{f}_N^l\}$ where $\boldsymbol{f}_i^l \in \mathbb{R}^M$ is the feature of the i-th token. A transformer block ϕ at depth l processes $\mathcal{F}^l$ by a Multi-head Self-Attention (MSA) layer and a point-wise Multi-Layer Perceptron (MLP) shown in Figure 2b. Let matrix $\boldsymbol{F} \in \mathbb{R}^{N\times M}$ be a row-wise concatenation of tokens $\mathcal{F}^l$. Then[2],

$$\phi(\boldsymbol{F}) \;=\; \mathrm{MLP}(\mathrm{MSA}(\boldsymbol{F})), \text{ such that} \qquad (1)$$

$$\mathrm{MSA}(\boldsymbol{F}) \;=\; [\boldsymbol{O}_1, \boldsymbol{O}_2, \ldots, \boldsymbol{O}_H]\boldsymbol{W}^O, \qquad (2)$$

where H is the number of heads, matrix $\boldsymbol{W}^O \in \mathbb{R}^{M\times M}$ is a learnable parameter of the block, $[,]$ is column-wise concatenation and $\boldsymbol{O}_h \in \mathbb{R}^{N\times d}$ is the output of h-th attention head for $d = M/H$:

$$\boldsymbol{O}_h = \boldsymbol{A}_h\boldsymbol{V}_h, \text{ such that}$$
$$\boldsymbol{A}_h = \mathrm{softmax}(\boldsymbol{Q}_h\boldsymbol{K}_h^\top/\sqrt{d}) \in \mathbb{R}^{N\times N}. \qquad (3)$$

Keys $\boldsymbol{K}_h$, queries $\boldsymbol{Q}_h$ and values $\boldsymbol{V}_h$ are linear projections of the input tokens (QKV projections):

$$\boldsymbol{Q}_h = \boldsymbol{F}\boldsymbol{W}_h^Q, \quad \boldsymbol{K}_h = \boldsymbol{F}\boldsymbol{W}_h^K, \quad \boldsymbol{V}_h = \boldsymbol{F}\boldsymbol{W}_h^V, \qquad (4)$$

[1]DeiT shares the same architecture as ViT but is trained with a refined recipe to improve data efficiency. Thus, we conduct experiments on DeiT.

[2]For compactness, we omit layer norm and skip-connections, see [45].

where $\boldsymbol{W}_h^Q \in \mathbb{R}^{M\times d}$, $\boldsymbol{W}_h^K \in \mathbb{R}^{M\times d}$, $\boldsymbol{W}_h^V \in \mathbb{R}^{M\times d}$ are learnable linear transformations. Note that the token feature dimension remains the same across the entire network. Moreover, the numbers of input and output tokens are the same, *i.e.*, $|\mathcal{F}^{l+1}| = |\mathcal{F}^l| = N$.

By inserting a token downsampling layer between transformer blocks, we reduce the number of tokens and thereby decreases the computational cost. Since downsampling inevitably loses information, the question is: how do we downsample/select tokens such that the performance of the model is not significantly degraded?

2.2. Efficient transformers

Similar to many machine learning models, the efficiency of transformers can be improved via meta-parameter search [15, 39], automated neural architecture search [11, 38, 50], manipulating the input size and resolution of feature maps [15, 27, 54], pruning [19], quantization [16], and sparsification [12], *etc*. For example, Dosovitskiy *et al*. [10] and Touvron *et al*. [43] obtain a family of ViT and DeiT models, respectively, by varying the input resolution, the number of heads H, and the feature dimensionality M. Each of the models operates with a different computational requirement and accuracy. In the following, we review techniques developed to improve the efficiency of transformers.

2.2.1 Efficient self-attention

The softmax-attention layer (3) has a quadratic time complexity w.r.t. the number of tokens, *i.e.*, $\mathcal{O}(N^2)$. In many NLP applications where every token represents a word or a character, N can be large, making attention a computation bottleneck [7, 32]. While many works improve the time complexity of attention layers, as we will see in Section 3.1, they are not the bottleneck in most vision transformers.

The time complexity of an attention layer can be reduced by restricting the attention field-of-view and thus imposing sparsity on $\boldsymbol{A}_h$ in (3). This can be achieved using the spatial relationship between tokens in the image/text domain [3, 4, 26, 29, 33, 53] or based on token values using locality-sensitive hashing, sorting, compression, clustering *etc*. [18, 20, 40, 41, 46–48]. Prior works have also proposed attention mechanisms with lower time complexity, *e.g.*, $\mathcal{O}(N)$ or $\mathcal{O}(N\log N)$ [5, 17, 28, 40]. Note that the goal of these above methods is to reduce the time complexity of the attention layer — the number of tokens remains the same across the transformer blocks. In contrast, our method reduces the number of tokens *after* attention has been computed. Thereby, we can utilize these methods to further improve the overall efficiency of transformers.

Recently, Wu *et al*. [51] proposed a new attention-based layer that learns a small number of query vectors to extract information from the input feature map. Roy *et al*. [35] cluster queries and keys to sparsify attention matrices and speed-up attention. Similarly, Wu *et al*. [52] proposed centriod transformer that unrolls a custom soft K-means as a special learnable attention layer. The effect is akin to PoWER-BERT [13] (with static strategy), which will be compared to extensively in Section 5. In comparison, our method has no learnable parameters, and it directly minimizes the reconstruction error due to token downsampling, by treating tokens as irregular samples of an underlying continuous signal.

2.2.2 Existing token downsampling methods

Grid-downsampling. Grid-downsampling techniques assume tokens to lie on a regular grid, which is usually achieved by arranging tokens using their initial locations on the image. The regular grid structure allows typical downsampling methods, such as max/average pooling or uniform sub-sampling to be used. For example, Liu *et al*. [21], Heo *et al*. [14], and Wang *et al*. [49] use convolutions with stride (*i.e*., subsampling after convolution) to downsample the feature maps formed by the tokens; similarly, Dai *et al*. [6] use average pooling. Note that as we have discussed in Section 1, standard transformer blocks do not retain the image grid structure and treat tokens as unordered sets. To encourage tokens to form a grid structure, Liu *et al*. [21] modify the transformer block and restrain the attention span of MSA to neighboring tokens on the input image grid. This inductive bias in the architecture — trading attention span for computation efficiency — enables their method to achieve state-of-the-art results in various applications.

In this paper, we focus on understanding the sole effect of token downsampling layers — we retain the standard transformer block architecture, *without* modifying the transformer block like Liu *et al*. [21]. As our analysis will show, without any modification to the transformer blocks, downsampling tokens via convolution with stride (or equivalently average pooling) actually performs worse than a simple algorithm that randomly select/remove tokens, verifying our hypothesis about the grid structure.

Score-based token downsampling. In the area of NLP, Goyal *et al*. [13] introduce PoWER-BERT to drop tokens based on a measure of *significance score*, which is defined as the total attention given to a token from all other tokens. Specifically, the significance scores of all tokens in the l-th transformer block, $\boldsymbol{s}^l \in \mathbb{R}^N$, is computed by

$$\boldsymbol{s}^l = \sum_{h=1}^{H} {\boldsymbol{A}_h^l}^\top \mathbf{1}, \tag{5}$$

where $\boldsymbol{A}_h^l$ is the attention weights of head h defined in (3). They only pass K_l tokens with the highest scores in $\boldsymbol{s}^l$ to the

next transformer block. Similar to our setting, the pruning is performed on all blocks.

PoWER-BERT is trained using a three-stage process. First, given a base architecture, they pretrain a model without pruning. In the second stage, a soft-selection layer is inserted after each transformer block, and the model is finetuned for a small number of epochs. Once learned, the number of tokens to keep, K_l, for each layer is computed from the soft-selection layers. Last, the model is finetuned again with the less significant tokens pruned. See [13] for details.

Recently, Rao *et al*. [34] proposed Dynamic-ViT that also uses scores to prune tokens. Unlike PoWER-BERT, which computes significance scores from attention weights, Rao *et al*. use a dedicated sub-network with learned parameters. The method requires knowledge distillation, Gumbel-Softmax, and straight-through estimators on top of the DeiT training and architecture.

We will analyze score-based methods in Section 3.2 and compare with them in Section 5.

3. Analysis

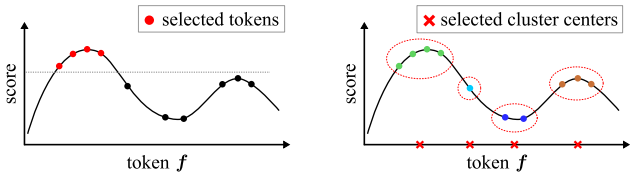
This section addresses three questions. First, we identify the computational bottleneck of vision transformers. Second, we discuss the limitations of score-based downsampling. Finally, we analyze how softmax-attention affects the redundancy of tokens in a transformer. The finding is important for designing downsampling algorithms.

3.1. Computation analysis of vision transformers

We analyze the time complexity and computational costs (measured in flops) of vision transformers (ViT/DeiT). We breakdown the computation into four categories: softmax-attention (3), QKV projections (4), O projection (2) and MLP (1). As shown in Table 1, in all these vision transformers, the main computational bottleneck is the fully-connected layers that spend over 80% of the total computation. In comparison, softmax-attention only takes less than 15%. Note that we explicitly break down the multi-head attention into the softmax-attention, QKV and O projections, as they have different time complexity. This decomposition reveals that the QKV and O projections spend most of the computations of the multi-head self-attention. The fully connected layers have a time complexity of $\mathcal{O}(LNM^2)$. By downsampling tokens (*i.e*., reducing N), we improve the time complexity without significantly reducing their capacity (*i.e*., the feature dimension M and number of layers L).

3.2. Limitations of score-based downsampling

Existing score-based token downsampling methods like PoWER-BERT and Dynamic-ViT utilize scoring functions to determine the tokens to keep (or prune). They keep tokens with the top-K scores and discard the rest. Since



(a) Score-based (b) Proposed

Figure 3: Score-based downsampling methods [13, 34] *vs.* the proposed method. In the figure, the x-axis represents the token values (in one dimension), and the y-axis represents their scores. Suppose four tokens are to be selected. (a) Score-based methods select tokens with higher scores. Since the scoring function is continuous, all tokens in the left lobe will be selected, resulting in redundancy and information loss in the right lobe. (b) The proposed method first forms four clusters to approximate the set of tokens, then selects the cluster centers. Thus, the output tokens are a more accurate representation of the original token set than the score-based methods.

these scoring functions are continuous with limited Lipschitz constants, tokens that are close in the feature space will be assigned similar scores. Therefore, similar tokens will likely either be all kept or discarded, as illustrated in Figure 3a. As our experiments show, this redundancy (in the kept tokens) and severe information loss (in the pruned tokens) deteriorate the computation-accuracy trade-off of the score-based downsampling methods.

3.3. Attention as a low-pass filter

Given a query vector $\boldsymbol{q}$, a set of key vectors $\mathcal{K} = \{\boldsymbol{k}_1, \dots, \boldsymbol{k}_N\}$, the corresponding value vectors $\mathcal{V} = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_N\}$ and a scalar $\alpha > 0$, softmax-attention computes the output via

$$\boldsymbol{o}(\boldsymbol{q}) = \frac{1}{z(\boldsymbol{q})} \sum_{i=1}^{N} \exp(\alpha\, \boldsymbol{q} \cdot \boldsymbol{k}_i)\, \boldsymbol{v}_i, \tag{6}$$

where $z(\boldsymbol{q}) = \sum_{i=1}^{N} \exp(\alpha\, \boldsymbol{q} \cdot \boldsymbol{k}_i)$. Note that we write $\boldsymbol{o}(\boldsymbol{q})$ to indicate that the output vector $\boldsymbol{o}$ is a function of the query $\boldsymbol{q}$. If the query vector and all key vectors are normalized to have a fixed ℓ_2 norm, we can rewrite (6) as

$$\begin{aligned}
\boldsymbol{o}(\boldsymbol{q}) &= \frac{1}{z'(\boldsymbol{q})} \sum_{i=1}^{N} \exp\Big(-\frac{\alpha}{2}\|\boldsymbol{q} - \boldsymbol{k}_i\|^2\Big)\, \boldsymbol{v}_i \\
&= \frac{1}{z'(\boldsymbol{q})} \int \exp\Big(-\frac{\alpha}{2}\|\boldsymbol{q} - \boldsymbol{k}\|^2\Big) \left(\sum_{i=1}^{N} \delta(\boldsymbol{k} - \boldsymbol{k}_i)\boldsymbol{v}_i\right) \mathrm{d}\boldsymbol{k} \\
&= \frac{1}{z'(\boldsymbol{q})}\, G\left(\boldsymbol{q}; \frac{1}{\alpha}\right) * S(\boldsymbol{q}; \mathcal{K}, \mathcal{V}),
\end{aligned} \tag{7}$$

where $*$ represents high-dimensional convolution, $z'(\boldsymbol{q}) = \sum_i \exp\big(-\frac{\alpha}{2}\|\boldsymbol{q} - \boldsymbol{k}_i\|^2\big) = G\big(\boldsymbol{q}; \frac{1}{\alpha}\big) * S(\boldsymbol{q}; \mathcal{K}, 1)$ is the nor-

Layer	Complexity	Computation (10^9 Flops)			
		ViT-B/384 (N=577)	ViT-B (N=197)	DeiT-S (N=197)	DeiT-Ti (N=197)
softmax-attn.	$\mathcal{O}(LN^2M)$	6.18	0.72	0.36	0.18
QKV proj.	$\mathcal{O}(LNM^2)$	12.25	4.18	1.05	0.26
O proj.	$\mathcal{O}(LNM^2)$	4.08	1.39	0.35	0.09
MLP	$\mathcal{O}(LNM^2)$	32.67	11.15	2.79	0.70
Total	$\mathcal{O}(LNM(M+N))$	55.5	17.6	4.6	1.3

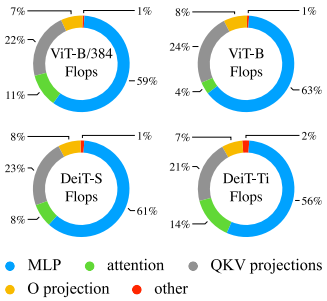


Table 1: Time complexity and computation breakdown of ViT [10] and DeiT [43]. L is the number of transformer blocks, N is the number of input tokens, and M is the feature dimensionality. All models take input images of size 224×224 except ViT-B/384, which uses 384×384. The softmax-attention layers constitute a fraction (15% or less) of the total compute, whereas fully-connected layers (MLP and projections) spend over 80%.

malization scalar function, $G(\boldsymbol{q}; \sigma^2) = \exp(-\|\boldsymbol{q}\|^2/2\sigma^2)$ is an isometric Gaussian kernel, and $S(\boldsymbol{q}; \mathcal{K}, \mathcal{V}) = \sum_{i=1}^{N} \delta(\boldsymbol{q} - \boldsymbol{k}_i)\, \boldsymbol{v}_i$ is a high-dimensional sparse signal, which is composed of N delta functions located at $\boldsymbol{k}_i$ with value $\boldsymbol{v}_i$. According to (7), given query vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_N$, there are two conceptual steps to compute softmax-attention:

1. filter $S(\boldsymbol{q}; \mathcal{K}, \mathcal{V})$ with a Gaussian kernel to get $\boldsymbol{o}(\boldsymbol{q})$, which is a high-dimensional continuous signal, and
2. sample $\boldsymbol{o}(\boldsymbol{q})$ at coordinates $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_N$ to get the output vectors $\boldsymbol{o}_1, \ldots, \boldsymbol{o}_N$.

Since Gaussian filtering is low-pass, $\boldsymbol{o}(\boldsymbol{q})$ is a smooth signal. Therefore, the output tokens of the attention layer, *i.e.*, discrete samples of $\boldsymbol{o}(\boldsymbol{q})$, contain similar feature values. In other words, there exists redundant information in the output tokens, and we can prune this redundancy to without losing much of the important information [37].

Note that our analysis is based on the normalized query and key vectors, which can be achieved by inserting a normalizing layer before the softmax-attention layer without significantly affecting the performance of a transformer, as demonstrated by Kitaev *et al*. [18]. It has also been empirically observed by Goyal *et al*. [13] and Rao *et al*. [34] that even without the normalization, transformers produce tokens with similar values. To demonstrate this, in all our experiments, we use standard multi-head attention *without* normalizing keys and queries. We conduct an ablation study with normalized keys and queries in Appendix F.

4. Token pooling

Pruning tokens inevitably loses information. In this section, we formulate a new token downsampling principle enabling strategical tokens selection that preserves the most information. Based on this principle, we formulate and discuss several Token Pooling algorithms.

Given a set of output tokens $\mathcal{F} = \{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$ of a transformer block, our goal is to find a smaller set of tokens $\hat{\mathcal{F}} = \{\hat{\boldsymbol{f}}_1, \ldots, \hat{\boldsymbol{f}}_K\}$ that minimizes the reconstruction error of $\mathcal{F}$ due to the downsampling. As we have shown in Section 3.3, we view $\mathcal{F}$ as discrete samples of a continuous signal $\boldsymbol{u}(\boldsymbol{f})$. Similarly, $\hat{\mathcal{F}}$ represents another signal $\hat{\boldsymbol{u}}(\boldsymbol{f})$. Since $K < N$, $\hat{\boldsymbol{u}}$ has fewer degree of freedom (*e.g.*, lower frequency) than $\boldsymbol{u}$ and thereby may not recover individual values of $\mathcal{F}$ exactly when we sample $\hat{\boldsymbol{u}}(\boldsymbol{f})$ at $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$.

To measure the reconstruction loss, we can construct a continuous signal $\hat{\boldsymbol{u}}(\boldsymbol{f}; \hat{\mathcal{F}})$ by interpolating $\hat{\mathcal{F}}$. The reconstruction error is computed via

$$\ell(\mathcal{F}, \hat{\mathcal{F}}) = \sum_{\boldsymbol{f}_i \in \mathcal{F}} \|\boldsymbol{f}_i - \hat{\boldsymbol{u}}(\boldsymbol{f}_i; \hat{\mathcal{F}})\|^2. \tag{8}$$

Note that similar reconstruction errors are used in nonuniform sampling literature [2, 23, 44]. To simplify our formulation and reduce computation, we use *nearest-neighbor* interpolation to construct $\hat{\boldsymbol{u}}$. As a consequence, the reconstruction error (8) becomes

$$\ell(\mathcal{F}, \hat{\mathcal{F}}) = \sum_{\boldsymbol{f}_i \in F} \min_{\hat{\boldsymbol{f}}_j \in \hat{F}} \|\boldsymbol{f}_i - \hat{\boldsymbol{f}}_j\|^2, \tag{9}$$

which is the asymmetric Chamfer divergence between $\mathcal{F}$ and $\hat{\mathcal{F}}$ [1, 24] and can be minimized by the K-Means algorithm, *i.e.*, clustering the tokens in $\mathcal{F}$ into K clusters.

The proposed Token Pooling layer is defined in Algorithm 1. It downsamples input tokens via clustering the tokens and returns the cluster centers (the average of the tokens in a cluster). As we have shown above, the algorithm directly minimizes the reconstruction error (9) caused by the downsampling. Intuitively, clustering the tokens provides an accurate and diverse approximation of the original set of tokens, compared to the top-K selection used by score-based downsampling methods, as shown in Figure 3b. Note that Token Pooling is robust to the initialization of cluster centers, as shown in Appendix B. Below, we provide details of the clustering algorithms.

K-Means. We use the K-Means algorithm to minimize (9) via the following iterations:

$$a(i) \leftarrow \arg\min_j \|\boldsymbol{f}_i - \hat{\boldsymbol{f}}_j\|, \forall i \in \{1, \ldots, N\} \tag{10}$$

$$\hat{\boldsymbol{f}}_j \leftarrow \sum_{i=1}^{N} [a(i) = j] \boldsymbol{f}_i \Big/ \sum_{i=1}^{N} [a(i) = j], \forall j \in \{1, \ldots, K\} \tag{11}$$

where $[]$ is the Iverson bracket and a is the cluster assignment function. The overall algorithm complexity is $\mathcal{O}(TKNM)$ where T is the number of iterations. The majority of the computation is spent on the repetitive evaluation of the distances between tokens and centroids in (10).

K-Medoids. The high complexity of $K-Means$ can over-shadow the benefit of carefully curating a smaller set of more informative tokens. Thus, we use the more efficient K-Medoids algorithm by replacing (11) with:

$$n(j) \leftarrow \arg\min_{i:a(i)=j} \sum_{i':a(i')=j} \|\boldsymbol{f}_i - \boldsymbol{f}_{i'}\|^2 \text{ and } \hat{\boldsymbol{f}}_j \leftarrow \boldsymbol{f}_{n(j)}. \tag{12}$$

These steps minimize objective (9) under the *medoid constraint*: $\hat{\mathcal{F}} \subseteq \mathcal{F}$.

The time complexity of the K-Medoids algorithm is $\mathcal{O}(TKN + N^2M)$, which is substantially lower in practice than K-means as the distances between tokens are computed once. Empirically, it converges within 5 iterations. Apart from the distance matrix computation, the cost is negligible w.r.t. the cost of the other layers. The complexity can be further improved using techniques proposed by Tiwari *et al*. [42]. In Table 3, 4 & 5 of the supplementary, we empirically compare the computation (measured in flops) of K-means and K-Medoids. As we will see in Section 5, utilizing a low-complexity algorithm to solve Equation (8) reveals the benefits of information preservation during token downsampling and significantly improves computation-accuracy tradeoff of standard transformers.

Weighted clustering. Reconstruction error (9) treats every token equally; however, each token contributes differently to the final output of an attention layer. Thus, we also consider a weighted reconstruction error: $\ell(\mathcal{F}, \hat{\mathcal{F}}; \boldsymbol{w}) = \sum_{\boldsymbol{f}_i \in F} \min_{\hat{\boldsymbol{f}}_j \in \hat{F}} w_i \|\boldsymbol{f}_i - \hat{\boldsymbol{f}}_j\|^2$ where $\boldsymbol{w} = [w_1, \ldots, w_N]$ are the positive weights corresponding to the individual tokens in $\mathcal{F}$. Appendix A details the clustering algorithms for the weighted case. A good choice of the weights is the significance scores (5), *i.e*., $w_i = s_i$. The significance score identifies tokens that influence the current transformer block most and thus should be approximated more precisely.

Note that when applying Token Pooling, it is important to separate special tokens like the classification token (and always keep them).

Algorithm 1: Token Pooling

input : tokens $\mathcal{F} = \{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$; target set size K; (optional) weights $\mathcal{W} = \{w_1, \ldots, w_N\}$
output: downsampled set $\hat{\mathcal{F}} = \{\hat{\boldsymbol{f}}_1, \ldots, \hat{\boldsymbol{f}}_K\}$

if $k \geq N$ **then return** F;
initialize cluster centers $\hat{\mathcal{F}}$ to be the K tokens from $\mathcal{F}$ with the highest weights ;
while *not converged* and *max number of iterations is not reached* **do**
  for $i \in \{1, ..., N\}$ **do** update cluster assignment $z(i) \leftarrow \arg\min_{j=1}^{K} \|\boldsymbol{f}_i - \hat{\boldsymbol{f}}_j\|$;
  for $j \in \{1, ..., K\}$ **do** update cluster center $\hat{\boldsymbol{f}}_j$ according to the chosen clustering algorithm, that is either (11), (12), (15) or (18) ;
return *weighted means of tokens in each cluster*

5. Experiments

Our implementation is based on DeiT [43], which uses the vanilla transformer architecture as ViT [10, 45] with a refined training recipe to improve data efficiency. We use the official DeiT implementation since it is a well-established baseline for vision transformer and has a well understood training pipeline. We insert downsampling layers after each transformer block. To evaluate the pure effect of downsampling, we keep the same all meta-parameters of DeiT, including the feature dimensionality, network depth, learning rate schedule, *etc*. We also do not use knowledge distillation. We conduct evaluation on ImageNet-1k [36], and our cost and performance metrics are flops and top-1 accuracy. Appendix H reports throughput of our models.

Methods. We study the effect of the following token downsampling methods:

1. *Convolution* downsampling (generalized average pooling). We implement grid-downsampling via convolution with stride 2, *i.e*., the tokens corresponding to the adjacent image patches are concatenated and mapped to $\mathbb{R}^M$. This design is widely used in many recent vision transformers [14, 21]. See details in Appendix D.
2. *PoWER-BERT.* We implement PoWER-BERT on DeiT, following [13].
3. *Random selection*, a simple baseline randomly selecting K tokens without replacement.
4. *Importance selection* chooses K tokens by drawing from a categorical distribution without replacement with probabilities proportional to the significance score (5).
5. *Token Pooling* with K-Means or K-Medoids algorithms, or their weighted versions, WK-Means or WK-Medoids, respectively. The weights are the significance scores (5).

Selection of $\boldsymbol{K} = [K_1, \ldots, K_L]$. To fairly compare our Token Pooling with PoWER-BERT and other baselines, all methods (except convolution downsampling for which we enumerate downsampling configurations, see Appendix D) use the same number of retained tokens for downsampling layers. Appendix C details the selection of $\boldsymbol{K}$.

Training protocol. (1) A base DeiT model (*e.g.*, DeiT-S) is trained using the original training [43]. (2) We insert soft-selection layers to the base DeiT model and finetune the model to automatically determine the number of tokens, $\boldsymbol{K}$. (3) We further finetune the model with individual downsampling methods using $\boldsymbol{K}$. Note that in the comparison, we also finetune the base DeiT model to ensure a fair comparison such that all models are trained with the same number of iterations, learning rate schedule, and optimizer, *etc*.

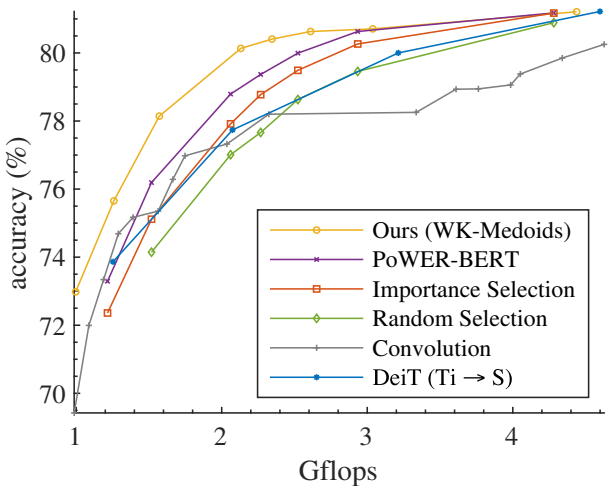
5.1. Main results

First, we apply different downsampling methods on DeiT-S. As a reference, we also show the results of various DeiT architectures (DeiT-Ti $\rightarrow$ S) formed by changing the feature dimension M. As shown in Figure 4a, random selection achieves a similar trade-off as lowering feature dimensionality M. While convolution with stride is better than adjusting M at the low-compute regime, it fails in high-compute regimes. Importance selection improves upon random selection but is still outperformed by PoWER-BERT. Our Token Pooling (with weighted K-Medoids) achieves the best trade-off in all regimes.

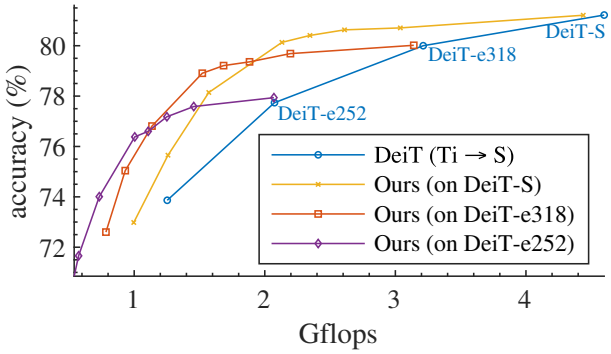
One interesting finding is that without any modifications to the transformer block, convolution with stride is outperformed by all other downsampling methods in the high compute regime, including the naive random selection , indicating loss of valuable information when aggregating tokens at intermediate layers using image grid.

Next, we apply Token Pooling to DeiT models with different M (DeiT-e252, DeiT-e318, and DeiT-S). Figure 4b shows trade-off curves for each DeiT model. Token Pooling enables each of the models to achieve a better computation-accuracy trade-off than simply varying the feature dimensionality M. For each computational budget, we improve accuracy by increasing M (to increase model capacity) with smaller $\boldsymbol{K}$ (to remove redundant computation). In other words, using Token Pooling enables us to use a larger model without increasing computational cost.

For the sake of completeness, Figure 5 shows the results of the proposed Token Pooling and state-of-the-art methods. Comparing with current state-of-the-art is *not* the main purpose of the paper. Thus we directly cite the results of [8, 25, 34, 49] *without* optimizing the training recipe (*e.g.*, total epochs, learning rate schedule, optimizer) for individual methods. Despite using the vanilla transformer block (DeiT), our results are comparable with or outperform recent vision transformers incorporated with modified transformer blocks. For example, PVT-Tiny is 75.5 % at 1.9 GFlops, and ours is 79.4 % at 1.9 GFlops; PVT-Small is 79.8 % at 3.8 GFlops, and ours is 80.7 % at 3 GFlops [49]. Our results (based on DeiT-S) are also on par with Swin: Swin-T is 81.3 % at 4.5 GFlops, ours is 81.4 % at 4.4 GFlops [21].



(a) Models based on DeiT-S



(b) Token Pooling on various DeiT architectures

Figure 4: **Main results.** (a) shows the accuracy when we apply different downsampling methods to DeiT-S. More is in Appendix G. (b) Token Pooling improves the trade-off of DeiT-S, DeiT-e318, DeiT-e252 (different feature dimensions M). At 1.2 Gflops, we should use Token Pooling + DeiT-e318 instead of DeiT-Ti.

Finally, we compare the best computation-accuracy trade-off achieved by Token Pooling (with WK-Medoids and varying M) with DeiT models in Figure 1a. As can be seen, utilizing Token Pooling, we significantly improve the computation costs of DeiT-Ti by 42% and improve the top-1 accuracy by 3.3 points at the same flops. Similar benefits can be seen on DeiT-e252 and DeiT-e318 in Figure 12 in the supplementary.

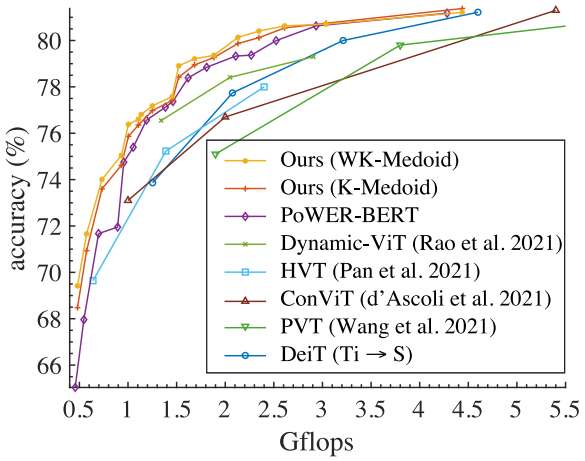


Figure 5: Comparison with current state-of-the-art methods. Note that this plot is for the sake of completeness — the goal of the paper is to study the effect of token downsampling, not competing with the state of the arts. Besides PoWER-BERT and DeiT, which are trained with similar recipe (epoch, learning rate schedule, optimizer) as ours, for other methods, we directly cite numbers from their papers.

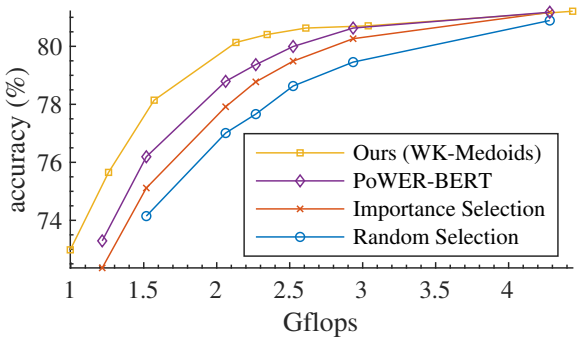
5.2. Ablation studies

Figure 6a compares methods utilizing significance scores. As can be seen, using importance selection improves upon the simple random selection. By minimizing the reconstruction error (9), our method achieves better cost-accuracy trade-off. Figure 6b evaluates Token Pooling with different clustering algorithms. Weighted versions outperform regular versions of K-Means and K-Medoids. Due to the higher time complexity, K-Means is outperformed by K-Medoids (the curves are shifted toward the right). See the metrics and flops used by clustering in table format in Appendix G. All Token Pooling variants outperform the baseline, demonstrating the effectiveness of our method.

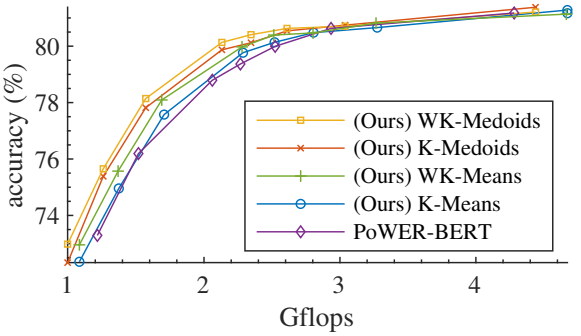
More ablation studies are in the appendices: clustering initialization (Appendix B), convolution downsampling/generalized average pooling (D), directly adding Token Pooling to a pretrained transformer (E), results with normalized keys and queries (F), detailed information ($\boldsymbol{K}$, flops, accuracy, clustering cost) of our models (G), and computation-accuracy trade-off measured by throughput on non-optimized implementation (H).

6. Conclusions

This paper provides two insights of standard vision transformers: first, their computational bottleneck is the fully-connected layers, and second, attention layers generate redundant representations due to the connection to Gaussian filtering. Token Pooling utilizes both insights and significantly improves the computation-accuracy balance of



(a) Ours *vs.* methods using significance score



(b) Variants of Token Pooling

Figure 6: Ablation studies of (a) downsampling methods using significance score, and (b) proposed Token Pooling using different clustering algorithms. The base model is DeiT-S for all methods.

DeiT, compared to existing downsampling techniques, on ImageNet-1k. With the increasing use of vision transformers, we believe our analyses, the proposed Token Pooling, and thorough evaluations are valuable and of interest to the vision community.

Limitations. As standard transformers, Token Pooling treats tokens as unordered sets. Thereby, even though the xy information of a token is embedded in feature values (via initial positional encoding), it is not immediately accessible at the output and can complicate downstream tasks like semantic segmentation and pose estimation where xy information is needed. This property is shared by all compared downsampling methods (except for the convolution downsampling). When xy information is required, techniques proposed by Wu *et al*. [51] and Marin *et al*. [22] can be utilized. For example, we can query the output tokens at individual xy locations to fill in a regular image grid [51].

The paper studies the effect of Token Pooling and various downsampling methods on the standard transformer architecture. It is left as future work to analyze them on modified/new transformers and on other applications.

References

[1] Harry G Barrow, Jay M Tenenbaum, Robert C Bolles, and Helen C Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. Technical report, SRI International Menlo Park CA Artificial Intelligence Center, 1977.

[2] Ricardo AF Belfor, Marc PA Hesp, Reginald L Lagendijk, and Jan Biemond. Spatially adaptive subsampling of image sequences. *Transactions on Image Processing*, 3:492–500, 1994.

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

[4] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[5] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *International Conference on Learning Representations (ICLR)*, 2021.

[6] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:4271–4282, 2020.

[7] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019.

[8] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. ConViT: Improving vision transformers with soft convolutional inductive biases. *International Conference on Machine Learning (ICML)*, 2021.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[10] 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. In *International Conference on Learning Representations (ICLR)*, 2020.

[11] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[12] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[13] Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. PoWER-BERT: Accelerating BERT inference via progressive word-vector elimination. In *International Conference on Machine Learning (ICML)*, pages 3690–3699, 2020.

[14] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[16] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018.

[17] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning (ICML)*, pages 5156–5165, 2020.

[18] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations (ICLR)*, 2020.

[19] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 598–605, 1990.

[20] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations (ICLR)*, 2018.

[21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, October 2021.

[22] Dmitrii Marin, Zijian He, Peter Vajda, Priyam Chatterjee, Sam Tsai, Fei Yang, and Yuri Boykov. Efficient segmentation: Learning downsampling near semantic boundaries. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2131–2141, 2019.

[23] Farokh Marvasti. *Nonuniform Sampling: Theory and Practice*. Springer Science & Business Media, 2012.

[24] Roey Mechrez, Itamar Talmi, Firas Shama, and Lihi Zelnik-Manor. Maintaining natural image statistics with the contextual loss. In *Asian Conference on Computer Vision (ACCV)*, pages 427–443, 2018.

[25] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable visual transformers with hierarchical pooling. *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[26] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning (ICML)*, pages 4055–4064. PMLR, 2018.

[27] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. ENet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint*

arXiv:1606.02147, 2016.

[28] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations (ICLR)*, 2021.

[29] Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2555–2565, 2020.

[30] Majid Rabbani. JPEG2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2):286, 2002.

[31] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[32] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations (ICLR)*, 2019.

[33] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. 32, 2019.

[34] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. *ArXiv*, abs/2106.02034, 2021.

[35] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

[36] 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 (IJCV)*, 115(3):211–252, 2015.

[37] Claude E Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, jan 1949.

[38] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MnasNet: Platform-aware neural architecture search for mobile. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2820–2828, 2019.

[39] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114, 2019.

[40] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models. *International Conference on Machine Learning (ICML)*, 2021.

[41] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning (ICML)*, pages 9438–9447, 2020.

[42] Mo Tiwari, Martin Jinye Zhang, James Mayclin, Sebastian Thrun, Chris Piech, and Ilan Shomorony. Banditpam: Almost linear time k-medoids clustering via multi-armed bandits. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[43] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. pages 10347–10357, 2021.

[44] Didem Unat, Theodore Hromadka, and Scott B Baden. An adaptive sub-sampling method for in-memory compression of scientific data. In *Data Compression Conference*, pages 262–271. IEEE, 2009.

[45] 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 (NeurIPS)*, 30:5998–6008, 2017.

[46] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

[47] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[48] Shuohang Wang, Luowei Zhou, Zhe Gan, Yen-Chun Chen, Siqi Sun, Yuwei Fang, Yu Cheng, and Jingjing Liu. Clusterformer: Clustering-based sparse transformer for long-range dependency encoding. In *ACL-IJCNLP 2021*, 2021.

[49] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *IEEE International Conference on Computer Vision (ICCV)*, pages 568–578, October 2021.

[50] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10734–10742, 2019.

[51] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *arXiv preprint arXiv:2006.03677*, 2020.

[52] Lemeng Wu, Xingchao Liu, and Qiang Liu. Centroid transformers: Learning to abstract with attention. *arXiv preprint arXiv:2102.08606*, 2021.

[53] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[54] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. ICNet for real-time semantic segmentation on high-resolution images. In *European Conference on Computer Vision (ECCV)*, pages 405–420, 2018.

[55] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6881–6890, 2021.