

Skip-Attention: Improving Vision Transformers by Paying Less Attention

Shashanka Venkataramanan^{1*}[†], Amir Ghodrati^{1*}, Yuki M. Asano²

Fatih Porikli¹, Amirhossein Habibian¹

¹Qualcomm AI Research[‡], ²QUVA Lab, University of Amsterdam

shashanka.venkataramanan@inria.fr ghodrati@qti.qualcomm.com

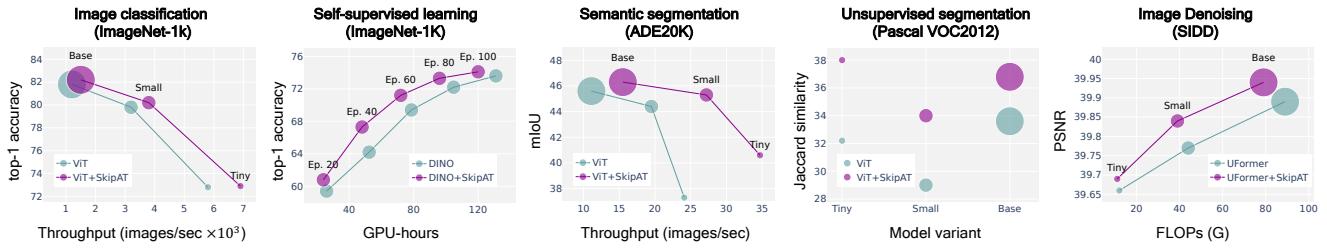


Figure 1. **Performance of SKIPAT across 5 different tasks.** Our novel SKIPAT parametric function achieves superior accuracy *vs.* efficiency trade-off over the baseline transformer on a wide array of tasks.

Abstract

This work aims to improve the efficiency of vision transformers (ViT). While ViTs use computationally expensive self-attention operations in every layer, we identify that these operations are highly correlated across layers – a key redundancy that causes unnecessary computations. Based on this observation, we propose **SKIPAT**, a method to reuse self-attention computation from preceding layers to approximate attention at one or more subsequent layers. To ensure that reusing self-attention blocks across layers does not degrade the performance, we introduce a simple parametric function, which outperforms the baseline transformer’s performance while running computationally faster. We show the effectiveness of our method in image classification and self-supervised learning on ImageNet-1K, semantic segmentation on ADE20K, image denoising on SIDD, and video denoising on DAVIS. We achieve improved throughput at the same-or-higher accuracy levels in all these tasks.

1. Introduction

The transformer architecture [50] has become an important and highly influential model family, due to its simplicity, scalability, and its wide range of applications. While originally stemming from the domain of natural language processing (NLP), with the advent of the Vision transformer (ViT) [15], this has become a standard architecture in computer vision, setting various state-of-the-art (SoTA) performances on tasks ranging from representation learning, semantic segmentation, object detection and video understanding [4, 5, 18, 30, 31].

However, the original formulation of the transformer includes a quadratic computational complexity with respect to the number of input tokens. Given that this number typically ranges from 14^2 for image classification all the way to $128^2 = 16\text{K}$ for image denoising, this constraint on memory and compute severely limits its applicability. To tackle this problem, there have been three sets of approaches. The first leverages redundancies across input tokens and simply reduces computation by efficient sampling, *e.g.*, dropping or merging redundant tokens [17, 46, 63]. This, however, means that the final output of the ViT is not spatially continuous and can thus not be used beyond image-level applications such as semantic segmentation or object localization. The second set of approaches aims to cheaply estimate the

*equal contribution

[†]Work done during internship at Qualcomm AI Research

[‡]Qualcomm AI Research is an initiative of Qualcomm Technologies,

Inc

attention computation, but generally at the cost of reduced performances [10, 65]. Finally, another line of works aims to merge convolutional architectures with the transformer, yielding hybrid architectures [29, 29, 39]. While these increase speed, they do not tackle the fundamental problem of the quadratic complexity, and often introduce an exorbitant number of design choices (essentially a union of those of the transformer and CNNs).

In this work, we propose a novel, so far unexplored approach to solving this problem: simply approximating the computationally expensive blocks of the transformer with a much faster, simpler parametric function. To arrive at this solution, we first thoroughly analyse the crucial multi-head self-attention (MSA) block of the ViT. Through this analysis, we find that the attention of the CLS tokens to the spatial patches has a very high correlation across the transformer’s blocks, thus leading to unnecessary computations. This motivates our approach to leverage attention from an early part of the model and simply reuse it for deeper blocks – basically “skipping” subsequent SA calculations instead of re-computing them at every layer.

Based on this, we go one step further and explore if the entire MSA block of a layer can be skipped by reusing the representation from previous layers. We find that a simple parametric function inspired from ResNeXt’s depthwise convolutions [62] can outperform the baseline performance – while being computationally faster in terms of throughput and FLOPs. Our method is general-purpose and can be applied to a ViT in any context: Figure 1 shows that our novel parametric function for Skipping Attention (SKIPAT) achieves superior accuracy *vs.* efficiency trade-off compared to the baseline transformer on a wide variety of tasks, datasets and model sizes.

In summary, our main contributions are as follows:

1. We propose a novel plug-in module that can be placed in any ViT architecture for reducing the costly $\mathcal{O}(n^2)$ Self-Attention computations (subsection 4.3)
2. We achieve state-of-the-art performances in terms of throughput at same-or-better accuracies for ImageNet, Pascal-VOC2012, SIDD, DAVIS and ADE20K (in the latter of which we obtain 40% speedup) (section 5)
3. We further demonstrate the generality of our method by obtaining a 26% reduction in self-supervised pre-training time (at no downstream accuracy loss) and by demonstrating superior on-device latency (subsection 5.2, subsection 5.1)
4. Finally, we analyse the sources of performance gains and extensively ablate our method to provide a model family which can be used for trading off accuracy and throughput (subsection 5.6)

2. Related Work

There has been great effort made to improve the efficiency of vision transformers (ViT) [15] from multiple aspects:

Token sampling improves the efficiency either by restructuring images during the tokenization step [21, 66], pruning the redundant tokens over training [26, 46] or dynamically at inference [7, 17, 43, 63]. Despite their effectiveness in reducing the computational cost in image classification, token sampling methods are hardly applicable to dense prediction tasks, *e.g.* semantic segmentation and image denoising, where the output image should be spatially continuous. Our approach is complementary to these lines of work and performs favorably against them as validated experimentally. Moreover, given that we keep representing all tokens throughout the network, our approach is applicable to both classification and dense prediction tasks.

Hybrid architectures integrate efficient convolutional modules into vision transformers [32, 36, 39] by adoption of MobileNet blocks in Uniformer [29], MobileNetV2 blocks in MobileViT [35] or using stacks of convolutions in the image tokenization step [19, 59]. Similarly, we use convolutions to speed up vision transformers, however, instead of crafting customized blocks as in [29, 35, 36, 39], we adhere to the original transformer architecture and approximate entire MSA computations through convolutions.

Efficient attentions address the quadratic cost of the self-attention operation in vision transformers by global down-sampling of key and value embeddings [54, 59], performing self-attention in local windows [31], alternating between local and global self-attentions [10, 35, 39], or replacing self-attention with a simple pooling [65]. However, reducing the self-attention to a local neighborhood hinders their ability to model the long range dependencies and leads to a significant performance drop with moderate speed up [69]. Moreover, some of the introduced operations come with no efficient support, *e.g.* cyclic shift in Swin [31], limiting their actual efficiency gains in terms of latency. Different to this, our method relies on the strong, yet inefficient self-attention operator at a few blocks and lighter, accurate attention estimators in other blocks. As the estimators only rely on standard convolutional operations, our method translates to actual latency gains. Related to this paper, [55, 60, 64] observed the redundancies in attention maps, for NLP tasks. However, instead of simply copying attention maps [60, 64], we propose an efficient parametric function that, as we show, are critical to achieve a high throughput *whilst* retaining high model performance in vision tasks. 同时

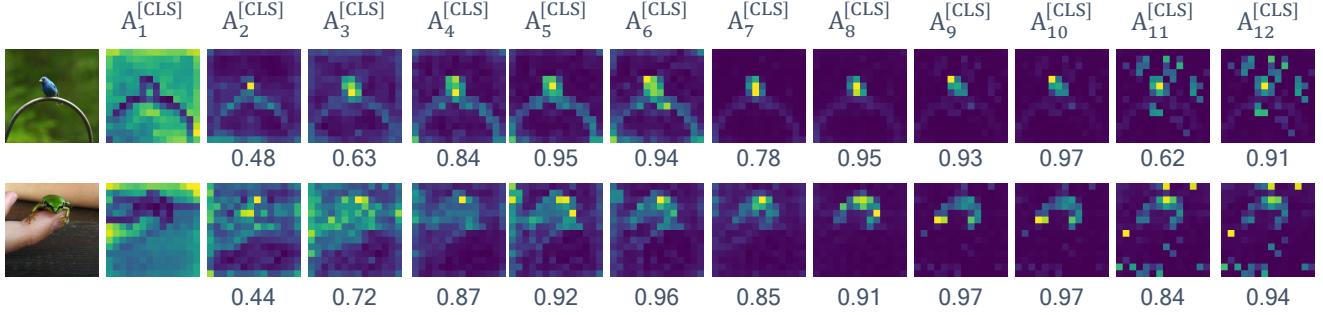


Figure 2. **Attention correlation.** Mean of the attention heads from the CLS token of a pretrained ViT-T/16 at different layers from the validation set of ImageNet-1K. Numbers below each attention map indicates the cosine similarity of $A_l^{[\text{CLS}]}$ with $A_{l-1}^{[\text{CLS}]}$.

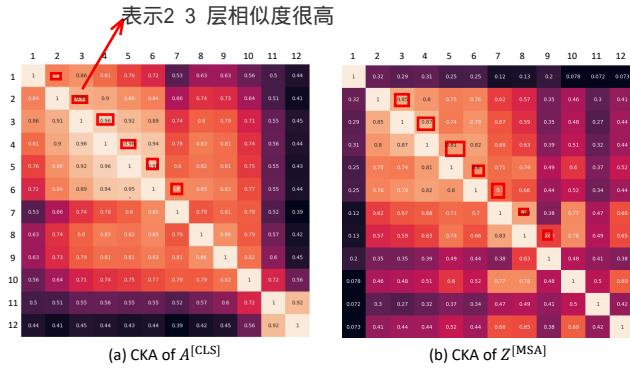


Figure 3. **CKA analysis of $A^{[\text{CLS}]}$ and $Z^{[\text{MSA}]}$** across different layers of pretrained ViT-T/16 on the validation set of Imagenet-1K. Vanilla ViT-T/16 has high correlation across both attention maps (layer 3 to 10) and $Z^{[\text{MSA}]}$ (layer 2 to 8)

Hierarchical architectures introduce hierarchical representations, as a long-standing principle in computer vision, to vision transformers [19, 31, 40, 54, 69]. Using a multi-scale representation significantly improves the memory and computational cost of the 表示类似的 isotropic architectures, such as ViT. More recently, the idea has been extended to more complex architectures with U-Net [57] or multi-branch structures [20]. Our work is complementary to these works, as they do not tackle the fundamental problem of reducing the quadratic complexity of the self-attention operator. We experimentally validate the effectiveness of our method on such isotropic and hierarchical architectures.

3. Related Work

There has been great effort made to improve the efficiency of vision transformers (ViT) [15] from multiple aspects:

Token sampling improves the efficiency either by restructuring images during the tokenization step [21, 66], pruning the redundant tokens over training [26, 46] or dynamically at inference [7, 17, 43, 63]. Despite their effectiveness in reducing the computational cost in image clas-

sification, token sampling methods are hardly applicable to dense prediction tasks, e.g. semantic segmentation and image denoising, where the output image should be spatially continuous. Our approach is complementary to these lines of work and performs favorably against them as validated experimentally. Moreover, given that we keep representing all tokens throughout the network, our approach is applicable to both classification and dense prediction tasks.

Hybrid architectures integrate efficient convolutional modules into vision transformers [32, 36, 39] by adoption of MobileNet blocks in Uniformer [29], MobileNetV2 blocks in MobileViT [35] or using stacks of convolutions in the image tokenization step [19, 59]. Similarly, we use convolutions to speed up vision transformers, however, instead of crafting customized blocks as in [29, 35, 36, 39], we adhere to the original transformer architecture and approximate entire MSA computations through convolutions.

Efficient attentions address the quadratic cost of the self-attention operation in vision transformers by global down-sampling of key and value embeddings [54, 59], performing self-attention in local windows [31], alternating between local and global self-attentions [10, 35, 39], or replacing self-attention with a simple pooling [65]. However, reducing the self-attention to a local neighborhood hinders their ability to model the long range dependencies and leads to a significant performance drop with moderate speed up [69]. Moreover, some of the introduced operations come with no efficient support, e.g. cyclic shift in Swin [31], limiting their actual efficiency gains in terms of latency. Different to this, our method relies on the strong, yet inefficient self-attention operator at a few blocks and lighter, accurate attention estimators in other blocks. As the estimators only rely on standard convolutional operations, our method translates to actual latency gains. Related to this paper, [55, 60, 64] observed the redundancies in attention maps, for NLP tasks. However, instead of simply copying attention maps [60, 64], we propose an efficient parametric function that, as we show, are

critical to achieve a high throughput whilst retaining high model performance in vision tasks.

Hierarchical architectures introduce hierarchical representations, as a long-standing principle in computer vision, to vision transformers [19, 31, 40, 54, 69]. Using a multi-scale representation significantly improves the memory and computational cost of the isotropic architectures, such as ViT. More recently, the idea has been extended to more complex architectures with U-Net [57] or multi-branch structures [20]. Our work is complementary to these works, as they do not tackle the fundamental problem of reducing the quadratic complexity of the self-attention operator. We experimentally validate the effectiveness of our method on such isotropic and hierarchical architectures.

4. Skip-Attention

4.1. Preliminaries

Vision Transformer. Let $x \in \mathbb{R}^{h \times w \times c}$ be an input image, where $h \times w$ is the spatial resolution and c is the number of channels. The image is first tokenized into $n = hw/p^2$ non-overlapping patches, where $p \times p$ is patch size. Each patch is projected into an embedding $z_i \in \mathbb{R}^d$ using a linear layer to obtain the tokenized image:

$$Z_0 = (z_1; \dots; z_n) \in \mathbb{R}^{n \times d} \quad (1)$$

Here, “;” denotes row-wise stacking. Positional embeddings are added to Z_0 to retain positional information. The token embeddings are then input to a $\mathcal{L} = \{1, \dots, L\}$ layer transformer whose output is denoted as Z_L . In the supervised setting, a learnable token $z^{[\text{CLS}]} \in \mathbb{R}^d$ is prepended to the tokenized image in (1) as $Z_0 := (z^{[\text{CLS}]}; Z_0) \in \mathbb{R}^{(n+1) \times d}$.

Transformer Layer. Every layer of the transformer consists of a multi-head self attention (MSA) block followed by a multi-layer perceptron (MLP) block. In the MSA block, the input, $Z_{l-1} \in \mathbb{R}^{n \times d}$, for $l \in \mathcal{L}$, is first projected into three learnable embeddings $\{Q, K, V\} \in \mathbb{R}^{n \times d}$. The attention matrix A , is calculated as

$$A := \sigma \left(\frac{QK^T}{\sqrt{d}} \right) \in \mathbb{R}^{n \times n} \quad (2)$$

where $\sigma(\cdot)$ denotes the row-wise softmax operation. The “multi-head” in MSA is defined by considering h attention heads where each head is a sequence of $n \times \frac{d}{h}$ matrix. The attention heads are reprojected back to $n \times d$ using a linear layer which is combined with the value matrix as

$$Z^{\text{MSA}} := AV \in \mathbb{R}^{n \times d} \quad (3)$$

The output representations from the MSA block is then input to the MLP block which comprises two linear layers

separated by a GeLU activation [24]. At a given layer l , the computational flow of representations through a transformer block is denoted as

$$Z_l \leftarrow Z_l^{\text{MSA}} + Z_{l-1}, \quad (4)$$

$$Z_l \leftarrow \text{MLP}(Z_l) + Z_l. \quad (5)$$

Both the MSA and MLP blocks have residual connections with layer normalization (LN) [3]. While MSA blocks in each layer of the transformer learn representations independently, in the next subsection, we show that empirically there exist high correlation across these layers.

4.2. Motivation: Layer Correlation Analysis

Attention-map correlation. The MSA block in ViT encodes the similarity of each patch to every other patch as an $n \times n$ attention matrix. This operator is computationally expensive with $\mathcal{O}(n^2)$ complexity (2). As ViTs scale up, *i.e.*, as n increases, the complexity grows quadratically and this operation becomes a bottleneck. Recent NLP works [51, 52] have shown that self-attention across adjacent layers in SoTA language models exhibit very high correlation. This raises the question – *is it worth to compute self-attention at every layer of a vision transformer?*

To address this question, we analyze the correlation of the self-attention maps across different layers of ViT. As shown in Figure 2, the self-attention maps from the class token, $A^{[\text{CLS}]}$, exhibit high correlation especially in the intermediate layers. The cosine similarity between $A_{l-1}^{[\text{CLS}]}$ and $A_l^{[\text{CLS}]}$ can be as high as 0.97, as indicated in the bottom of each attention map in Figure 2. Similar behavior is observed from other token embeddings, which we analyze in the supplementary material. We quantitatively analyze this correlation across all the samples of the validation set of ImageNet-1K, by computing the Centered Kernel Alignment (CKA) [12, 27] between $A_i^{[\text{CLS}]}$ and $A_j^{[\text{CLS}]}$ for every $i, j \in \mathcal{L}$. CKA measures the similarity between representations obtained from intermediate layers of the network, where a high value of CKA indicates high correlation between the representations. From Figure 3 (a), we observe that ViT-T has a high correlation across $A^{[\text{CLS}]}$ especially from layer 3 through 10.

Feature correlation. In ViTs, the high correlation is not just limited to $A^{[\text{CLS}]}$, but the representation from MSA blocks, Z^{MSA} , also show high correlation throughout the model [42]. To analyze the similarity across these representations, we compute the CKA between Z_i^{MSA} and Z_j^{MSA} for every $i, j \in \mathcal{L}$. We observe from Figure 3 (b), that Z^{MSA} also have high similarity across adjacent layers of the model especially in the earlier layers, *i.e.*, from layer 2 through 8.

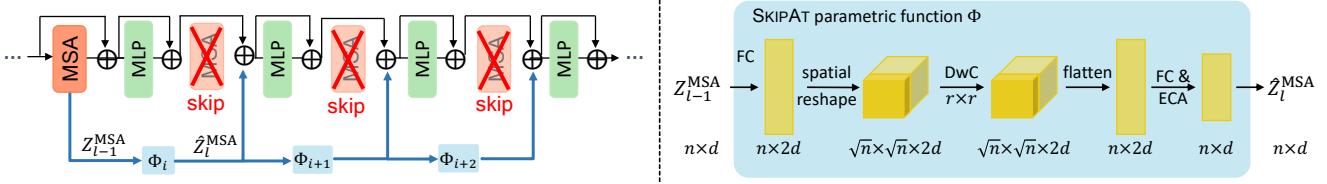


Figure 4. **SKIPAT framework** We illustrate SKIPAT on ViT [15]. The SKIPAT parametric function (Φ) uses representations of the MSA block (in solid color) Z_{l-1}^{MSA} as input, which undergoes a series of transformations. An element-wise summation (\oplus) with the output of the MLP block from layer $l-1$ and \hat{Z}_l^{MSA} is used as input to the MLP block at layer l . The MSA operation (crossed out) is thus not computed and is discarded from the computational graph. With SKIPAT the total number of layers remains unchanged.

4.3. Improving Efficiency by Skipping Attention

Based on our observation of high representation similarity across MSA blocks of a transformer (subsection 4.2), we propose to leverage the correlation across both the attention matrix and the representations from the MSA block to improve the efficiency of vision transformers. Instead of computing the MSA operation (3) independently at every layer, we explore a simple and effective strategy to utilize dependencies across the features from these layers.

In particular, we propose to skip MSA computation in one or more layers of a transformer by reusing representations from its adjacent layers. We term this operation as *Skip Attention* or *SKIPAT*. As the compute and memory benefit from skipping the entire MSA block is greater than skipping just the self-attention operation ($\mathcal{O}(n^2d + nd^2)$ vs. $\mathcal{O}(n^2d)$), in this paper we focus on former. However, instead of directly re-using features, *i.e.*, copying the features from the source MSA block to one or more adjacent MSA blocks, we introduce a parametric function. The parametric function ensures that directly reusing features does not affect the translation invariance and equivariance in these MSA blocks and acts as a strong regularizer to improve model generalization.

SKIPAT parametric function Let $\Phi : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times d}$ denote the parametric function that maps output of the MSA block from $l-1$ to l as $\hat{Z}_l^{MSA} := \Phi(Z_{l-1}^{MSA})$. Here, \hat{Z}_l^{MSA} is the approximation of Z_l^{MSA} . The parametric function can be as simple as an identity function, where Z_{l-1}^{MSA} is directly reused. Instead of computing MSA operation at l , we use Z_{l-1}^{MSA} as the input to the MLP block at l . When using an identity function, due to the absence of MSA operation at l , the relation across tokens is no longer encoded in the attention matrix, which affects representation learning. To mitigate this, we introduce the SKIPAT parametric function inspired from ResNeXt [62] as shown in Figure 4, to encode local relations among tokens. The SKIPAT parametric function consists of two linear layers and a depth-wise convolution (DwC) [9] in between, as follows:

$$\hat{Z}_l^{MSA} := \text{ECA}\left(\text{FC}_2\left(\text{DwC}\left(\text{FC}_1(Z_{l-1}^{MSA})\right)\right)\right) \quad (6)$$

In the case of supervised learning, we first separate the **CLS** embeddings from $Z^{MSA} \in \mathbb{R}^{(n+1) \times d}$ into class embeddings $Z_C^{MSA} \in \mathbb{R}^d$ and the patch embeddings to $Z_P^{MSA} \in \mathbb{R}^{n \times d}$. The patch embeddings are then input to the first linear layer $\text{FC}_1 : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times 2d}$, which expands the channel dimension. This is followed by $\text{DwC} : \mathbb{R}^{\sqrt{n} \times \sqrt{n} \times 2d} \rightarrow \mathbb{R}^{\sqrt{n} \times \sqrt{n} \times 2d}$ with kernel $r \times r$ to capture cross-token relations. Note that before the DwC operation, we spatially reshape the input matrix to a feature tensor. The output of the DwC is then flattened back to a vector and fed to the last FC layer $\text{FC}_2 : \mathbb{R}^{n \times 2d} \rightarrow \mathbb{R}^{n \times d}$ which reduces the channel dimension back to its initial dimension d . We use GeLU activations after FC_1 and DwC. Following [53], we use efficient channel attention module (ECA) after FC_2 to enhance the cross-channel dependencies. The ECA module first aggregates the features along the channel dimension using global average pooling (GAP). A 1×1 convolution with adaptive kernel size proportional to channel dimension is applied followed by sigmoid activation. This operation of the ECA module enhances cross-channel dependencies. We then concatenate the embedding of the class-token with the output of the ECA to obtain \hat{Z}_l^{MSA} .

SKIPAT framework. The overall framework of SKIPAT is illustrated in Figure 4. SKIPAT can be incorporated into any transformer architecture which we empirically show in subsection 5.4. Depending on the architecture, one can skip the MSA operation in one or more layers of the transformer. In ViT, as we empirically observe that representations from the MSA block, Z^{MSA} , have high correlations from layer 2 through 7 (subsection 4.2), we employ the SKIPAT parametric function in these layers. This means that we use the Z_2^{MSA} as input to the SKIPAT parametric function and skip MSA operations in layers 3-8. Instead, the features from the output of the SKIPAT parametric function is used as input to the MLP block. The computation flow of representations is now

modified to

$$Z_l \leftarrow \Phi(Z_{l-1}^{\text{MSA}}) + Z_{l-1} \quad (7)$$

$$Z_l \leftarrow \text{MLP}(Z_l) + Z_l \quad (8)$$

Due to the presence of residual connections in the MSA and MLP blocks, which is standard in ViT [15], the MLP blocks at layer 3 through 8 learn representations independently and cannot be discarded from the computational graph. It is important to note that, with SKIPAT the total number of layers in ViT remain unchanged, but there are fewer MSA blocks.

Complexity: MSA vs. SKIPAT The self-attention operation involves three operations. Firstly, the token embeddings are projected into query, key and value embeddings, secondly, attention matrix A is computed as dot product between Q and K and finally, the output representations are computed as dot product between A and V . This results in a complexity of $\mathcal{O}(4nd^2 + n^2d)$. Since $d \ll n$, the complexity of MSA block can be reduced to $\mathcal{O}(n^2d)$.

The SKIPAT parametric function consists of two linear layers and one depth-wise convolution operation, which results in a $\mathcal{O}(2nd^2 + r^2nd)$ complexity, where $r \times r$ is the kernel size of the DwC operation. The overall complexity of SKIPAT can be reduced to $\mathcal{O}(nd^2)$ since $r^2 \ll d$. Thus, SKIPAT has fewer FLOPs than the MSA block as $\mathcal{O}(nd^2) < \mathcal{O}(n^2d)$ when n increases as transformers scale up.

5. Experiments

5.1. Image Classification

We use ViT-T/16 [15], ViT-S/16 [15] and ViT-B/16 [15] as our backbone on ImageNet-1K. For fair comparisons, we follow the experimental settings in [48] and evaluate SKIPAT against SoTA methods: A-ViT [63], Dynamic-ViT [38], SViTE [7], SPViT [26], ATS [17], PS-ViT [46], HVT [40] and Rev-Vit [34]. To the best of our knowledge, these are all the works that improve the efficiency of ViT without modifying its underlying architecture.

From Table 1, we observe that SKIPAT achieves the best accuracy vs. efficiency trade-off compared to all SoTA methods on different variants of ViT. Notably, we outperform baseline ViT-T, ViT-S and ViT-B by 0.1%, 0.4% and 0.4% respectively, while SoTA methods achieve lower accuracy or are on-par with the baseline. Since SKIPAT uses a parametric function to skip computing MSA blocks, our reduction in number of parameters and in FLOPs is comparable to the SoTA. In terms of throughput, SKIPAT is 19%, 21% and 25% faster than the baseline ViT-T, ViT-S and ViT-B respectively. Dehghani *et al.* [13] highlight the significance of using *throughput* as a metric to measure model efficiency: as the reduction in FLOPs does not necessarily correspond

BACKBONE	METHOD	TOP-1↑ (%)	PARAM↓ ($\times 10^6$)	GFLOPS↓	THROUGHPUT↑ (IM/s $\times 10^3$)
ViT-T/16	ViT [15]	72.8	5.7	1.2	5.8
	A-ViT [63]	71.0	5.7	0.8	6.3
	Dynamic ViT [43]	70.9	—	0.9	6.1
	SViTE [7]	71.7	4.0	0.9	6.2
	SPViT [26]	72.7	5.7	0.9	6.7
	ATS [17]	72.7	5.7	0.9	6.1
	PS-ViT [46]	72.6	—	0.7	6.6
	HVT [40]	70.2	5.7	0.7	7.2
SKIPAT		72.9	5.8	1.1	6.9
ViT-S/16	ViT [15]	79.8	22.4	4.6	3.2
	A-ViT [63]	78.6	22.4	3.6	3.4
	Dynamic ViT [43]	78.3	23.1	3.4	3.6
	SViTE [7]	80.2	13.1	2.7	3.5
	ATS [17]	79.7	22.4	2.9	3.3
	PS-ViT [46]	79.4	—	2.6	3.9
	SPViT [26]	79.3	22.1	2.7	3.5
	Rev-ViT [34]	79.8	22.4	4.6	3.6
SKIPAT		80.2	22.1	4.0	3.8
ViT-B/16	ViT [15]	81.8	87.3	17.6	1.2
	SViTE [7]	81.6	52.0	11.5	1.3
	Rev-ViT [34]	81.5	87.3	17.6	1.2
	PS-ViT [46]	81.5	—	9.8	1.6
SKIPAT		82.2	86.7	15.2	1.5

Table 1. **Image classification on ImageNet-1K.** Accuracy vs. efficiency comparison of SKIPAT with SoTA methods for image resolution 224×224 . For all the methods, we measure throughput (image/sec) with a batch size of 1024 on a single NVIDIA A100 GPU, averaged over the validation set of ImageNet-1K.

to improvements in latency, as it does not take into account the degree of parallelism or other hardware details. In line with this argument, we observe that while SoTA methods such as ATS [17] and SPViT [26] achieve large reduction in FLOPs, they actually have lower throughput when compared to SKIPAT. Furthermore, HVT [40] while achieving a higher gain in both throughput and FLOPs has poor top-1 accuracy (2.6% drop in ViT-T and 1.8% drop in ViT-S). Thus, SKIPAT demonstrates the ability to simultaneously improve both accuracy and throughput over SoTA methods.

Visualizing attention maps and Z^{MSA} correlation. We analyze the effect of the SKIPAT parametric function by visualizing the mean of attention heads of the CLS token from the last four layers of ViT-T/16. From Figure 5, we observe that while attention maps from vanilla ViT (last two layers) do not solely attend to the object, the attention maps from SKIPAT accurately focuses on the object. It is interesting to note that, the attention maps from SKIPAT are also capable of attending to multiple objects in the image (Figure 5: second example). We further analyze the CKA of the representations from MSA block across all the layers of ViT-T/16. From Figure 6, we observe that Z^{MSA} has lower correlation across layers except between the layers where the MSA operation is skipped (layer 3 to 8). However, unlike vanilla ViT (Figure 3 (b)) the correlation from each layer to every other layer is quite low. This shows that

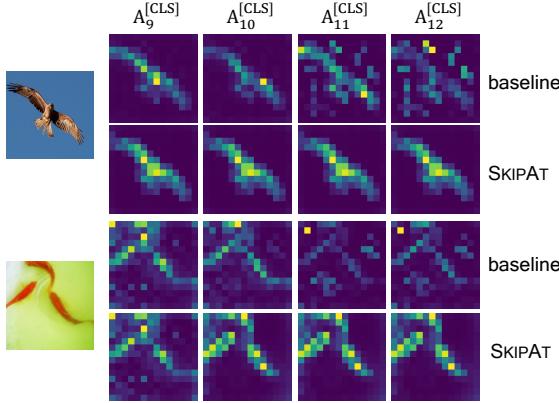


Figure 5. **Visualizing attention maps.** Mean of the attention of different heads from $A^{[\text{CLS}]}$ from last four layers of ViT-T/16 on the validation set of ImageNet-1K. Attention maps from last four blocks show that SKIPAT localizes the object better than vanilla ViT.



Figure 6. **CKA analysis of SKIPAT** shows that Z^{MSA} has lower correlation between layers. The high correlation is only between consecutive layers 2 through 8, where the MSA operation is skipped.

our SKIPAT parametric function acts as a strong regularizer and thus improves the representations of the model.

METHOD	JACCARD \uparrow	CORLOC \uparrow
ViT-T [15]	32.2	39.5
ViT-T + SKIPAT	38.0	41.5
ViT-S [15]	29.0	40.6
ViT-S + SKIPAT	34.0	41.2
ViT-B [15]	33.6	36.4
ViT-B + SKIPAT	36.8	37.2

Table 2. **Unsupervised Segmentation and Object Localization** using Jaccard similarity [5] and Correct Localization (CorLoc) [37], on the validation set of Pascal VOC2012. All models have been pretrained on ImageNet-1K in a supervised setting.

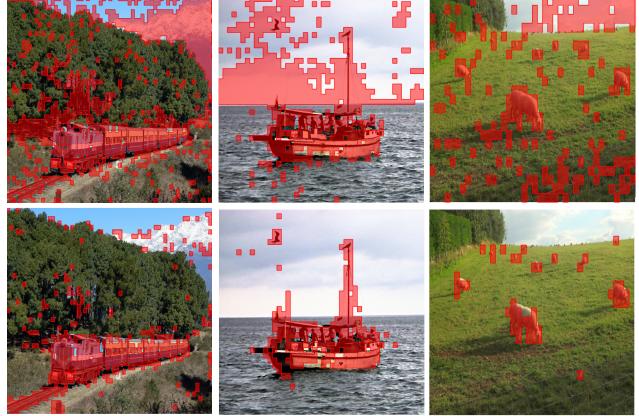


Figure 7. **Visualization of segmentation masks** using vanilla ViT-S/16 (*top*) and ViT-S + SKIPAT (*bottom*) pretrained supervisely on ImageNet-1K. We visualize masks obtained by thresholding the self-attention maps to keep 80% of the mass.

Probing self-attention maps in ViTs. We further analyze whether pretrained ViTs can attend to semantically meaningful regions of the image when evaluated on a different dataset without fine-tuning it. We follow the evaluation protocol in [5], and visualize the segmentation masks produced from the final layer of the pretrained SKIPAT on the Pascal-VOC12 [16] validation set. From Figure 7,¹ we observe that while vanilla ViT-S/16 does not accurately attend to the object, SKIPAT is able to localize objects quite accurately without any fine-tuning. To quantify this observation, we follow [5] and use the Jaccard similarity between predicted segmentation mask and ground truth mask. As shown in Table 2, SKIPAT outperforms different variants of vanilla ViT with a significant gap in terms of Jaccard similarity. Additionally, we measure the quality of the generated maps for unsupervised object localization using CorLoc [37] as the evaluation metric. From Table 2, we observe that SKIPAT achieves notable gains across all variants of ViT.

Performance on mobile device. To verify the efficiency of SKIPAT on low-power devices, we measure its inference time (averaged over 20 iterations) on a Samsung Galaxy S22 device powered by Qualcomm “Snapdragon® 8 Gen. 1 Mobile Platform” with a Qualcomm® Hexagon™ processor², for image resolutions of 224×224 and 384×384 using ViT-T/16. The inference is performed on Neural Processing Unit in 8-bit precision. As shown in Table 3, SKIPAT improves the runtime by 19% for image size of 224×224 . The gain is even larger at 34% for image resolution $384 \times$

¹The original image sources, before masking, from left to right:
Kangra valley train (CC BY-SA 4.0)
Ecuadorian fishing boat (CC BY-SA 2.0)
Sheep near Snowshill (CC BY-SA 2.0)

²Snapdragon and Qualcomm Hexagon are products of Qualcomm Technologies, Inc. and/or its subsidiaries.

384, since the number of token increases. Thus, skipping computationally-heavy MSA blocks increases throughput by large margins and is confirmed even on mobile hardware.

METHOD	224×224	384×384
ViT-T/16	5.65	20.49
ViT-T/16 + SKIPAT	4.76	15.22

Table 3. **On-device latency** (in msec) of vanilla ViT *vs.* SKIPAT for different image resolutions on a Samsung Galaxy S22 powered by Qualcomm Snapdragon 8 Gen. 1.

5.2. Self-Supervised Learning with DINO

Next, we show the generality of SKIPAT as its use in the backbone for self-supervised representation learning (SSL), using DINO [5]. Since, SSL methods are quite expensive in the pretraining stage in terms of compute and training time, we illustrate that SKIPAT achieves comparable performance to using a ViT but with shorter training time. Following the experimental settings of DINO [5], we use ViT-S/16 [15] as our student and teacher networks with SKIPAT parametric function. We pretrain both baseline and ours using DINO for 100 epochs. We observe that SKIPAT achieves almost the same performance as fully trained DINO with around 26% less training time (73.3% in 96 GPU-hours *vs.* 73.6% in 131 GPU-hours). When trained on 100 epochs, we observe that SKIPAT outperforms DINO by 0.5% (74.1% *vs.* 73.6%). We show the performance of SKIPAT to downstream tasks in the supplementary material.

5.3. Semantic Segmentation on ADE20K

We go beyond classification and show the performance of SKIPAT to dense prediction tasks such as seman-

METHOD	BACKBONE	MIoU↑	GFLOPs↓	THROUGHPUT↑
Semantic FPN [25]	ResNet-101 [65]	40.7	261	24.1
	PoolFormer-S36 [65]	42.0	191	8.4
	PoolFormer-M36 [65]	42.4	271	5.4
UperNet [61]	ResNet-18 [23]	39.9	886	17.1
	ResNet-101 [23]	44.9	1031	12.0
	Swin-T [31]	45.8	945	14.2
	ConvNeXt-T [32]	46.7	939	15.7
	ViT-T [15]	37.3	212	24.1
Uformer [57]	ViT-T + SKIPAT	40.6	173	34.7
	ViT-S [15]	44.4	360	19.5
	ViT-S + SKIPAT	45.3	283	27.2
	ViT-B [15]	45.6	787	11.1
	ViT-B + SKIPAT	46.3	633	15.5

Table 4. **Semantic Segmentation results on ADE20K.** All models are pretrained on ImageNet-1k and fine-tuned on ADE20K. Following Swin [31] and ConvNeXt [32], we report mIoU with multi-scale testing. FLOPs and throughput are calculated on the input size of 2048×512 . Throughput of all models are measured with a batch size of 1 on a single NVIDIA A100 GPU, averaged over 100 forward passes.

METHOD	PSNR↑	SSIM↑	GFLOPs↓	THROUGHPUT↑
UNet [45]	39.65	-	35	-
DAGL [38]	38.94	0.953	255	-
DeamNet [44]	39.47	0.957	145	-
MPRNet [68]	39.71	0.958	573	-
NBNNet [8]	39.75	0.959	91	-
Restormer [67]	40.02	0.960	140	-
Uformer-T [57]	39.66	-	12	17.6
Uformer-T + SKIPAT	39.69	0.959	11	22.2
Uformer-S [57]	39.77	0.959	44	15.1
Uformer-S + SKIPAT	39.84	0.960	39	18.9
Uformer-B [57]	39.89	0.960	89	9.2
Uformer-B + SKIPAT	39.94	0.960	77	10.9

Table 5. **Image denoising** on SIDD dataset using PSNR and SSIM [56] as the evaluation metrics in the RGB space. FLOPs and throughput are calculated on the input size of 256×256 , on a single NVIDIA V100 GPU, averaged over the test set of SIDD.

tic segmentation. We follow the experimental settings in [31, 32] and use MMSegmentation [11] to evaluate SKIPAT on ADE20K [70]. We observe from Table 4, that SKIPAT consistently outperforms all variants of ViT with 15% fewer FLOPs and 25% improved throughput. Interestingly, SKIPAT-S (ViT-S + SKIPAT) achieves 8% higher mIoU while being faster than ViT-T. Furthermore, SKIPAT-S has comparable mIoU with Swin-T [31] whilst having 3× fewer FLOPs and being 1.7× faster. Comparing to fully convolution-based architectures, SKIPAT-T (ViT-T + SKIPAT) is on par with ResNet-18 in mIoU while having 4.7× fewer FLOPs and being 1.8× faster.

5.4. Image Denoising

SKIPAT can also generalize to low-level tasks such as image denoising on SIDD [1], which consists of images with real-world noise. We also demonstrate that SKIPAT can generalize to other transformer architectures. In particular, we apply it on Uformer [57], a SoTA image denoising model. Uformer is a U-shaped hierarchical network with Swin transformer blocks as the encoder and decoder, and skip connections between them. In SKIPAT, we skip window self-attention (WSA) block in each decoder block by reusing attention of the corresponding encoder block via SKIPAT parametric function. Detailed implementation is in the supplementary material. Following the experimental settings in [57], we observe in Table 5 that SKIPAT outperforms the baseline Uformer variants with the 25% higher throughput on average. Furthermore, we observe that SKIPAT-B (Uformer-B + SKIPAT) achieves comparable performance with Restormer [67], in terms of PSNR and SSIM, which is the SoTA image denoising method while having 2× fewer FLOPs. Thus, we show the ability of SKIPAT to generalize to different tasks and also across architectures.

METHOD	FastDVDNet [47]	PaCNet [49]	VRT [30]	UniFormer [28]	UniFormer+ SKIPAT
PSNR↑	34.04	34.79	36.52	35.24	35.16
GFLOPS↓	41.9	34.8	708.8	93.2	77.1

Table 6. **Video denoising** Quantitative comparison (average RGB channel PSNR) with state-of-the-art methods for video denoising on DAVIS, with additive noise level $\sigma = 30$. FLOPs are calculated per frame per patch size of 256×256 .

FUNCTION Φ	KERNEL	CHANNEL EXPANSION	TOP-1↑ (%)	THROUGHPUT↑ (img/sec $\times 10^3$)
ViT-T	-	-	65.8	5.8
IDENTITY	-	-	61.1	8.5
CONV	5×5	-	65.4	5.2
DwC	5×5	-	65.6	7.8
SKIPAT	3×3	-	67.1	7.3
SKIPAT	5×5	2	67.7	6.9
SKIPAT	7×7	-	67.4	6.6
SKIPAT	-	0.5	64.4	7.4
SKIPAT	5×5	1	65.9	7.2
SKIPAT	5×5	2	67.7	6.9

Table 7. **Ablations** using ViT-T/16 on ImageNet-1K for 100 epochs. We measure throughput (image/sec) with a batch size of 1024 on a single NVIDIA A100 GPU, averaged over the validation set of ImageNet-1K.

5.5. Video Denoising

We further apply our model to the temporal task of video denoising. As encoder and decoder backbone, we use UniFormer [28], a U-shaped hybrid encoder-decoder architecture with 3D convolutions and spatio-temporal global self-attention blocks. Detailed implementation is provided in the supplementary material. Similar to image denoising, we skip MSA blocks in the decoder, however, simply adopt a naive SKIPAT, where we reuse window self-attention matrix, A , of the corresponding encoder block using an Identity function. We empirically observe that reusing attention works better in this task, and shows the ability of our method to be applied for different scenarios. We follow the experimental settings in [47] and train SKIPAT on DAVIS [41] dataset. We train using Charbonnier loss [6] on patches of $7 \times 128 \times 128$ using a multiple-input, multiple-output (MIMO) paradigm (i.e. the model outputs 7 reconstructed frames from 7 input frames) for noise level $\sigma = 30$. From Table 6, we observe that SKIPAT performs on par with baseline UniFormer, while having 17% fewer FLOPs. This shows that SKIPAT can generalize to temporal tasks.

5.6. Ablations

All ablations are performed using ViT-T/16 on ImageNet-1K for 100 epochs to reduce the training time. Unless specified, following SKIPAT we skip the MSA blocks from layer 3 through 8 for all ablations.

Parametric function Φ . We study the effect of different parametric functions in terms of accuracy and throughput.

As discussed in subsection 4.3, Φ can be as simple as an identity function, where we directly reuse representations from a previous MSA block into one of more subsequent MSA blocks. From Table 7, using an identity function results in a 4.7% drop in top-1 accuracy while being 47% faster than baseline ViT. Using a convolution or DwC [9] with kernel size 5×5 as a parametric function leads to the same performance as the baseline. However, DwC is 0.2% better and 50% faster than convolution, and 34% faster than the baseline. SKIPAT parametric function outperforms all.

Kernel size. By default SKIPAT uses a DwC with kernel size of 5×5 . As shown in Table 7, while using a 3×3 kernel is faster than default SKIPAT by 6%, it is 0.6% worse in terms of accuracy. A larger kernel size has poor accuracy and lower throughput. However, irrespective of the kernel size, SKIPAT outperforms the baseline ViT-T by at least 1.4%, showing its ability to encode cross-token interactions.

Channel expansion. In the SKIPAT, the first linear layer FC_1 , expands the channel dimension from $d \rightarrow 2d$. Table 7 shows the impact of channel dimension, *i.e.*, when the channel expansion ratio of FC_1 is 1.0 ($d \rightarrow d$) and 0.5 ($d \rightarrow d/2$). We observe that while the lower channel expansion ratio improves the throughput, it performs worse than default SKIPAT. This could be due to sub-optimal representations encoded by the DwC due to fewer filters.

Skipping MSA in alternate configuration. Instead of skipping the MSA operation in the layers 3 – 8, we study the effect of skipping MSA operation at $l \in \{3, 5, 7, 9\}$. We observe the latter configuration outperforms the baseline ViT by 2.7% (65.8 vs. 67.5%). However, it performs 0.2% lower and is 8% slower than our default SKIPAT configuration.

6. Conclusion

We proposed SKIPAT, a plug-in module that can be placed in any ViT architecture for reducing the costly Self-Attention computations. SKIPAT leverages the dependency across MSA blocks and bypasses attention computation by re-using attention from previous MSA blocks. To ensure that the metaphorical sharing is caring we introduced a simple and light parametric function that does not affect the inductive bias encoded in MSA. The SKIPAT function is able capture cross-token relations and outperforms the baseline while being computationally faster in terms of throughput and FLOPs. We plugged SKIPAT into different transformer architectures and showed its effectiveness on 7 different tasks.

References

- [1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone

- cameras. In *CVPR*, 2018. 8
- [2] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A high-quality denoising dataset for smartphone cameras. In *CVPR*, 2018. 12
- [3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4
- [4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1
- [5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 1, 7, 8, 12, 13
- [6] Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *ICIP*, 1994. 9, 12
- [7] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *NeurIPS*, 2021. 2, 3, 6, 13
- [8] Shen Cheng, Yuzhi Wang, Haibin Huang, Donghao Liu, Haoqiang Fan, and Shuaicheng Liu. Nbnnet: Noise basis learning for image denoising with subspace projection. In *CVPR*, 2021. 8
- [9] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 5, 9
- [10] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *NeurIPS*, 2021. 2, 3, 13
- [11] MM Segmentation Contributors. MM Segmentation: Openmmlab semantic segmentation toolbox and benchmark. <https://github.com/open-mmlab/mmsegmentation>, 2020. 8, 12
- [12] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *JMLR*, 2012. 4
- [13] Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. The efficiency misnomer. In *ICLR*, 2022. 6
- [14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 12
- [15] 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 *ICLR*, 2020. 1, 2, 3, 5, 6, 7, 8, 13
- [16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>. 7, 12, 13
- [17] Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei, Sunando Sengupta, Hamid Reza Vaezi Joze, Hamed Pirsavash, and Juergen Gall. Adaptive token sampling for efficient vision transformers. In *ECCV*, 2022. 1, 2, 3, 6, 13
- [18] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *CVPR*, 2019. 1
- [19] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet’s clothing for faster inference. In *ICCV*, 2021. 2, 3, 4
- [20] Jiaqi Gu, Hyoukjun Kwon, Dilin Wang, Wei Ye, Meng Li, Yu-Hsin Chen, Liangzhen Lai, Vikas Chandra, and David Z Pan. Multi-scale high-resolution vision transformer for semantic segmentation. In *CVPR*, 2022. 3, 4
- [21] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *NeurIPS*, 2021. 2, 3
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 12
- [23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 8
- [24] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 4
- [25] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019. 8
- [26] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Bin Ren, Minghai Qin, Hao Tang, and Yanzhi Wang. Spvit: Enabling faster vision transformers via soft token pruning. In *ECCV*, 2022. 2, 3, 6, 13
- [27] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, 2019. 4
- [28] Kunchang Li, Yali Wang, Peng Gao, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unified transformer for efficient spatiotemporal representation learning. *CVPR*, 2022. 9, 12
- [29] Kunchang Li, Yali Wang, Gao Peng, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unified transformer for efficient spatial-temporal representation learning. In *ICLR*, 2021. 2, 3, 12
- [30] Jingyun Liang, Jiezheng Cao, Yuchen Fan, Kai Zhang, Rakesh Ranjan, Yawei Li, Radu Timofte, and Luc Van Gool. Vrt: A video restoration transformer. *arXiv preprint arXiv:2201.12288*, 2022. 1, 9, 12
- [31] 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. In *ICCV*, 2021. 1, 2, 3, 4, 8, 13
- [32] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022. 2, 3, 8, 13
- [33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 12
- [34] Karttikeya Mangalam, Haoqi Fan, Yanghao Li, Chao-Yuan Wu, Bo Xiong, Christoph Feichtenhofer, and Jitendra Malik. Reversible vision transformers. In *CVPR*, 2022. 6, 13
- [35] Sachin Mehta and Mohammad Rastegari. Mobilevit: Lightweight, general-purpose, and mobile-friendly vision transformer. In *ICLR*, 2021. 2, 3, 13
- [36] Sachin Mehta and Mohammad Rastegari. Separable self-

- attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680*, 2022. 2, 3
- [37] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization. In *CVPR*, 2022. 7
- [38] Chong Mou, Jian Zhang, and Zhuoyuan Wu. Dynamic attentive graph learning for image restoration. In *ICCV*, 2021. 6, 8
- [39] Junting Pan, Adrian Bulat, Fuwen Tan, Xiatian Zhu, Lukasz Dudziak, Hongsheng Li, Georgios Tzimiropoulos, and Brais Martinez. Edgevits: Competing light-weight cnns on mobile devices with vision transformers. In *ECCV*, 2022. 2, 3
- [40] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable vision transformers with hierarchical pooling. In *ICCV*, 2021. 3, 4, 6, 13
- [41] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 9
- [42] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? In *NeurIPS*, 2022. 4
- [43] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *NeurIPS*, 2021. 2, 3, 6, 13
- [44] Chao Ren, Xiaohai He, Chuncheng Wang, and Zhibo Zhao. Adaptive consistency prior based deep network for image denoising. In *CVPR*, 2021. 8
- [45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 8
- [46] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. In *CVPR*, 2022. 1, 2, 3, 6, 13
- [47] Matias Tassano, Julie Delon, and Thomas Veit. Fastdvdnet: Towards real-time deep video denoising without flow estimation. In *CVPR*, 2020. 9, 12
- [48] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 6, 12
- [49] Gregory Vaksman, Michael Elad, and Peyman Milanfar. Patch craft: Video denoising by deep modeling and patch matching. In *ICCV*, 2021. 9
- [50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1
- [51] Jesse Vig. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*, 2019. 4
- [52] Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284*, 2019. 4
- [53] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. In *CVPR*, 2020. 5
- [54] Wenhui 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 *CVPR*, 2021. 2, 3, 4, 13
- [55] Yujing Wang, Yaming Yang, Jiangang Bai, Mingliang Zhang, Jing Bai, Jing Yu, Ce Zhang, Gao Huang, and Yunhai Tong. Evolving attention with residual convolutions. In *ICML*, 2021. 2, 3
- [56] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004. 8
- [57] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. Uformer: A general u-shaped transformer for image restoration. In *CVPR*, 2022. 3, 4, 8, 12
- [58] Ross Wightman. Pytorch image models. <https://github.com/rwightman/pytorch-image-models>, 2019. 12
- [59] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvlt: Introducing convolutions to vision transformers. In *ICCV*, 2021. 2, 3
- [60] Tong Xiao, Yinqiao Li, Jingbo Zhu, Zhengtao Yu, and Tongran Liu. Sharing attention weights for fast transformer. In *IJCAI*, 2019. 2, 3
- [61] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 8, 12
- [62] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 2, 5
- [63] Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *CVPR*, 2022. 1, 2, 3, 6, 13
- [64] Chengxuan Ying, Guolin Ke, Di He, and Tie-Yan Liu. Lazyformer: Self attention with lazy update. *arXiv preprint arXiv:2102.12702*, 2021. 2, 3
- [65] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *CVPR*, 2022. 2, 3, 8, 13
- [66] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021. 2, 3, 13
- [67] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, 2022. 8
- [68] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *CVPR*, 2021. 8
- [69] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *ICCV*, 2021. 2, 3, 4
- [70] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 8, 12

A. Implementation details

A.1. Hyper-parameters

ImageNet-1K: Image classification. We train SKIPAT on the ILSVRC-2012 dataset [14] with 1000 classes (referred as ImageNet-1K). We follow the experimental settings of DeiT [48] and use the codebase from the timm library [58] to train ViT-T, ViT-S and ViT-B. We use the default 16×16 patch size, using an image resolution of 224×224 with total number of tokens $n = 196$. We train baseline ViT and SKIPAT for 300 epochs from scratch on 4 NVIDIA A100 GPUs using batch sizes of 2048 for ViT-T and 1024 for ViT-S and ViT-B.

ImageNet-1K: Self-supervised learning. We follow the experimental settings of DINO [5] and pre-train DINO and SKIPAT on ImageNet-1K using ViT-S/16 as the backbone. While likely the hyperparameters could be tuned further for our proposed SKIPAT method, we use same hyperparameters for both the baseline and ours, yielding a conservative estimate of our model’s performance. We pre-train both methods from scratch for 100 epochs using 4 NVIDIA A100 GPUs. For linear-probing, we freeze the backbone from the pre-training stage and fine-tune the classifier for 100 epochs, exactly as done in [5].

Pascal-VOC2012: Unsupervised object segmentation. We use the Pascal VOC 2012 [16] validation set for this experiment, containing 1449 images. We follow DINO and obtain unsupervised segmentation masks by thresholding the averaged self-attention map (extracted from the last layer of a pretrained ViT/SKIPAT model) to keep 80% of the mass. The Jaccard similarity J between a predicted mask, P , and ground-truth mask, G , is defined as:

$$J(P, G) = \frac{G \cap P}{G \cup P}$$

We report Jaccard similarity, averaged over all the samples.

ADE20K: Semantic segmentation. We evaluate SKIPAT on ADE20K [70], a widely-used semantic segmentation dataset, covering 150 semantic categories. The dataset includes 20K and 2K images in the training and validation set, respectively. Different variants of SKIPAT are evaluated using UperNet [61] as the backbone. We use our ImageNet-1K pretrained model to initialize the backbone and Kaiming [22] initialization for other layers. We use AdamW [33], with an initial learning rate of $6e-5$, weight decay of $1e-2$, and linear warmup of 1500 iterations. All models are trained for 160K iterations with a batch size of 16 using MMSegmentation repo [11]. We keep the same hyper-parameters for SKIPAT and ViT.

SIDD: Image denoising. We follow the experimental settings in Uformer [57] and train SKIPAT on the Smartphone Image Denoising Dataset (SIDD) [2] which consists of real-world noise. The training samples are first randomly cropped to 128×128 patches and input to the model, which is trained for 250 epochs using batch size 32. The model is then evaluated on images of size 256×256 .

DAVIS: Video denoising. We further apply our model to the temporal task of video denoising. We adopt the same U-shape encoder-decoder based architecture of UFormer. As the encoder and decoder backbone, we use UniFormer [28]. We train the model on noise level $\sigma = 30$ using Charbonnier loss [6] on patches of $7 \times 128 \times 128$ using a multiple-input, multiple-output (MIMO) paradigm [30] (*i.e.*, the model outputs 7 reconstructed frames from 7 input frames). During inference, a video is divided into 3D patches of $7 \times 128 \times 128$ with an overlap of 10 pixels. Each patch is fed to the model and the outputs are merged to obtain the final denoised video. Following [47], PSNR is calculated as averaged over videos. We use the same training hyper-parameters as image denoising.

A.2. Architecture

Image Classification. All baseline ViT variants have 12 layers in total, which remains unchanged with SKIPAT. Following the CKA analysis of Z^{MSA} in Figure 3(b) of our main paper, we skip computing the MSA blocks in layer 3 through 8 for all ViT variants and retrain it from scratch.

Image Denoising. We apply SKIPAT to Uformer [29] a SoTA image denoising model. Uformer is a U-shaped hierarchical network with Swin transformer blocks as the encoder and decoder, and skip connections between them. In SKIPAT, we skip window self-attention (WSA) block in each decoder block by reusing attention of the corresponding encoder block via SKIPAT parametric function. Let $Z_l^{\text{WSA}_e} \in \mathbb{R}^{n \times c}$ denote the output of the WSA block at layer l from the encoder and $Z_{l-1}^d \in \mathbb{R}^{n \times c}$ denote the output of the layer $l - 1$ from the decoder of Uformer. The input to the WSA block (which is skipped) at layer l of the decoder is given by

$$\hat{Z}_l^{\text{WSA}_d} = \Phi(Z_l^{\text{WSA}_e}; Z_{l-1}^d) \in \mathbb{R}^{n \times 2c} \quad (9)$$

Here, “;” denotes concatenation along the channel dimension. We show the framework of SKIPAT on Uformer in Figure 8

Video Denoising. we apply SKIPAT to UniFormer [28], a U-shaped hybrid encoder-decoder architecture with 3D convolutions and spatio-temporal global self-attention blocks. The encoder of UniFormer comprises two 3D

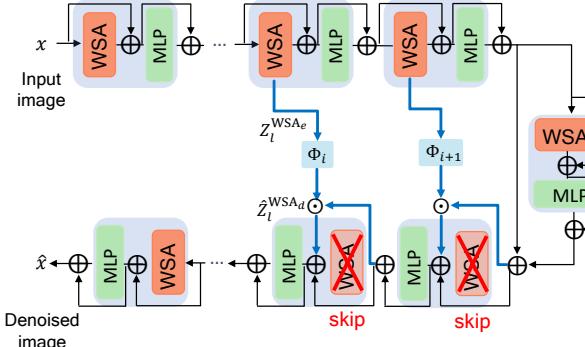


Figure 8. **Framework of SKIPAT on Uformer** Instead of standard MSA block in ViT, Uformer uses window self-attention (WSA) block similar to Swin Transformer. We skip WSA block in the layers close to the bottleneck.

convolution layers followed by two spatio-temporal transformer layers with global self-attention (MSA) blocks. A downsampling operation is used after every layer in the encoder. The decoder is symmetric to the encoder with two transformer layers followed by two 3D convolution layers with an upsampling operation between each layer. Similar to Uformer, skip connections are used between encoder and decoder. Similar to image denoising, we skip MSA blocks in the decoder, however, simply adopt a naive SKAT, where we reuse global self-attention matrix A_l , from the encoder at layer l in the corresponding decoder stage at the same layer using an Identity function. Let $A_l^e \in \mathbb{R}^{n \times n}$ denote the self-attention matrix at layer l from the encoder. The self-attention in the decoder stage at layer l is given by $A_l^d = I(A_l^e) \in \mathbb{R}^{n \times n}$, where $I(\cdot)$ is the identity function. We observe that skipping attention A using an identity function works better than skipping MSA blocks using the SKIPAT parametric function. This shows the generality of SKIPAT, regardless of approximating attention or MSA blocks.

B. Additional experiments

Image classification. Here we extend our SoTA comparison with methods that go beyond vanilla ViT architectures. These methods include hierarchical (Swin, PVT, Poolformer, MobileViT, Twins-SVT) and Hybrid (ConvNext, CoAT) architectures. We provide the complete set of SoTA methods that improve the efficiency of ViT either by token sampling (extending Table 1 in our main paper), using hybrid architectures or window self-attention blocks in Table 8. Apart from methods that perform efficient token sampling, none of the other methods are directly comparable because they modify the underlying architecture of ViT, either by using window self-attention blocks or reducing the overall number of transformer layers.

BACKBONE	METHOD	TOP-1 (%)	PARAM ($\times 10^6$)	GFLOPS	THROUGHPUT (img/sec $\times 10^3$)
ViT-T/16	T2T-ViT [66]	71.7	5.8	1.1	–
	ConvNeXt (iso) [32]	72.7	5.7	1.1	5.8
	ViT [15]	72.8	5.7	1.2	5.8
	A-ViT [63]	71.0	5.7	0.8	6.3
	Dynamic ViT [43]	70.9	–	0.9	6.1
	SViT [7]	71.7	4.0	0.9	6.2
	SPViT [26]	72.7	5.7	0.9	6.7
	ATS [17]	72.7	5.7	0.9	6.1
	PS-ViT [46]	72.6	–	0.7	6.6
	HVT [40]	70.2	5.7	0.7	7.2
SKIPAT		72.9	5.8	1.1	6.9
ViT-S/16	ConvNext-T [32]	82.1	29.0	4.5	2.6
	ConvNeXt (iso) [32]	79.7	22.4	4.3	3.3
	Swin-T [31]	81.3	28.3	4.5	2.5
	T2T-ViT [66]	80.7	21.5	5.2	–
	CoaT-Lite-Small	81.9	20.0	4.0	–
	Poolformer-S24 [65]	80.3	21.0	3.4	–
	Twins-SVT-S [10]	81.7	24.0	2.8	–
	MobileViT-S [35]	78.4	5.6	2.0	–
	PVT [54]	79.8	24.5	3.8	–
	SKIPAT	80.2	22.1	4.0	3.8
ViT-B/16	Swin-S [31]	83.5	88.0	15.4	1.0
	Twins-SVT-B [10]	83.2	56.0	8.6	–
	PVT [54]	81.7	61.4	9.8	–
	ConvNeXt (iso) [32]	82.0	87.3	16.9	1.3
	SKIPAT	82.2	86.7	15.2	1.5

Table 8. **Image classification on ImageNet-1K.** Accuracy vs. efficiency comparison of SKIPAT with SoTA methods for image resolution 224×224 . For all the methods, we measure throughput (image/sec) with a batch size of 1024 on a single NVIDIA A100 GPU, averaged over the validation set of ImageNet-1K.

Unsupervised segmentation of DINO. We follow DINO [5] and evaluate the performance of baseline DINO vs. SKIPAT on unsupervised object segmentation on Pascal-VOC2012 [16] dataset. We follow the experimental setting as discussed in Appendix A and observe that baseline DINO has a Jaccard similarity of 45.3 while SKIPAT achieves 44.7. While SKIPAT outperforms DINO on image classification by 0.5%, we achieve comparable performance in terms of unsupervised object segmentation.

C. Additional ablations

Reusing self-attention. As mentioned in Subsection 3.3, we skip the Z^{MSA} in SKIPAT as the compute and memory benefit from skipping the entire MSA block is greater than

skipping just the self-attention operation. Here we study the effect of skipping just the self-attention operation. Let A_{l-1} denote the self-attention matrix at layer $l-1$, then the self-attention matrix at layer l is given by $\hat{A}_l = I(A_{l-1})$. Similar to SKIPAT we skip computing the self-attention matrix from layers 3 through 8. As parametric function Φ , we use an identity mapping and train ViT-T/16 from scratch for 100 epochs on ImageNet-1K. We observe from Table 9, that skipping the self-attention matrix results in a top-1 accuracy of 63.2% which is 2.1% higher than the skipping Z^{MSA} with an identity function (61.1% - Table 7 of main paper). However, skipping self-attention matrix results in 20% decrease in throughput (8500 \rightarrow 6800 images/sec) as compared to using an identity function to skip MSA block. It is interesting to note that skipping self-attention matrix results in a lower drop in performance as compared to skipping MSA block. However, applying a parametric function to skip self-attention can be challenging due to the properties of the self-attention matrix, and we leave this to future work.

SKIPAT in pretrained model. As mentioned in subsection A.2, we train SKIPAT with all variants of ViT from scratch. For completeness, we also study the effect of skipping the self-attention matrix and the MSA block on a pre-trained ViT-T using an Identity function, *without retraining*. We observe from Table 9 that skipping the self-attention computation in layers 3 through 8, results in a top-1 accuracy of 53.9%, while skipping MSA blocks results in top-1 accuracy of 47.8%. It is interesting to note that the drop in top-1 accuracy from skipping self-attention is merely 19% (72.8 \rightarrow 53.9) on average and does not result in an *extremely* large drop as one might expect. This shows that there indeed exists high correlation across self-attention and Z^{MSA} , which SKIPAT utilizes to improve the efficiency of the ViTs.

METHOD	TRAINING	TOP-1 (%)	THROUGHPUT
A	✓	63.2	6800
Z^{MSA}	✓	61.1	8500
A	✗	53.9	6800
Z^{MSA}	✗	47.8	8500

Table 9. **Ablations** on the effect of skipping the self-attention, A , and the MSA block, Z^{MSA} . In the first two rows, models are trained for 100 epochs. In the last two rows we use a pretrained ViT-T/16 and simply skip computations in blocks 3-8 during inference. For all the experiments with use Identity function as Φ .

D. CKA analysis of attention from ViT-T

As discussed in Section 3.2 of our main paper, we analyze the CKA of the self-attention matrix for all tokens between different layers of ViT-T/16 pretrained on ImageNet-1K.

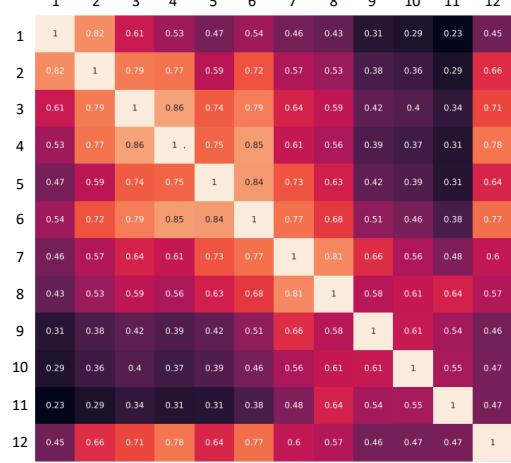


Figure 9. **CKA analysis of A** for all tokens from pretrained vanilla ViT-T/16 on the validation set of ImageNet-1K. We observe a high correlation for all tokens in A from layers 1 to 8.

Since in the supervised setting $A \in \mathbb{R}^{(n+1) \times (n+1)}$, we first remove the CLS token to obtain $A^P \in \mathbb{R}^{n \times n}$. We then compute the CKA of A_l^P for $l \in \mathcal{L}$. We observe from Figure 9, that there exists a high correlation across all the tokens from the self-attention matrix. Thus, reusing self-attention from different layers of the ViT can improve the overall throughput while yielding comparable accuracy as the baseline ViT.