SIMA: SIMPLE SOFTMAX-FREE ATTENTION FOR VISION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, vision transformers have become very popular. However, deploying them in many applications is computationally expensive partly due to the Softmax layer in the attention block. We introduce a simple yet effective, Softmax-free attention block, SimA, which normalizes query and key matrices with simple ℓ_1-norm instead of using Softmax layer. Then, the attention block in SimA is a simple multiplication of three matrices, so SimA can dynamically change the ordering of the computation at the test time to achieve linear computation on the number of tokens or the number of channels. We empirically show that SimA applied to three SOTA variations of transformers, DeiT, XCiT, and CvT, results in on-par accuracy compared to the SOTA models, without any need for Softmax layer. Interestingly, changing SimA from multi-head to single-head has only a small effect on the accuracy, which further simplifies the attention block.

1 INTRODUCTION

Recently, vision transformers have become very popular. Compared to CNNs, they achieve better accuracy, however, deploying transformers in devices with smaller computational resources is challenging. One reason is that a transformer model calls the Softmax layer several times which calls $exp(.)$ operation consequently. We know that the $exp(.)$ operation is costly particularly in smaller devices with limited computational resources. For instance, implementing $exp(.)$ on FGPA is much more costly compared to implementing simple multiplication or addition operations.

As an example observation, Table A1 of Ivanov et al. (2021) measures the run-time of each component for a BERT encoder on V100 GPUs. Softmax consumes more time compared to any other components including query (Q), key (K), value (V) operation (Softmax: 453 μs , QKV projections: 333 μs, QK^T: 189 μs). This is remarkable since the FLOPS of Softmax is much lower than those other components (Softmax: 0.2 GFLOPS, QKV projections: 25.7 GFLOPS, QK^T: 4.3 GFPLOS). Similar observation are made in Stevens et al. (2021); Vasyltsov & Chang (2021).

We are interested in simplifying the attention mechanism by removing the Softmax layer. We believe one role of the Softmax layer is to normalize the attention values so that tokens can compete with each other. Our main idea is to enable this competition by normalizing the query and key matrices with their ℓ_1-norm before multiplying them. Then, removing the Softmax layer results in the whole attention mechanism to boil down to simply multiplying three matrices "query", "key", and "value".

As a bi-product, due to the associative property of multiplication, there are two possible orderings of multiplying these three matrices at the test time. Depending on the ordering, the computation can be quadratic on the number of tokens, N, or that of channels, D. Hence, we can reduce the computation further by dynamically deciding on the ordering at the test time by comparing N and D without affecting the training process. Moreover, since we normalize the vectors before multiplying, our method is numerically more stable so we use half-precision floating point without overflowing. Note that this is not our main novelty and the same trick can be used on few other linear attention models.

The attention mechanism deals with the tokens without considering their ordering. This is an interesting property that opens the door to many applications. For instance, the distribution of the tokens is relatively robust compared to CNNs when we mask (drop) 75% of the tokens in masking auto-encoder (MAE He et al. (2021)). Moreover, the tokens can be seen as a non-ordered set that can come from various sources (e.g., multiple cameras or non-camera sensors). Note that this permutation

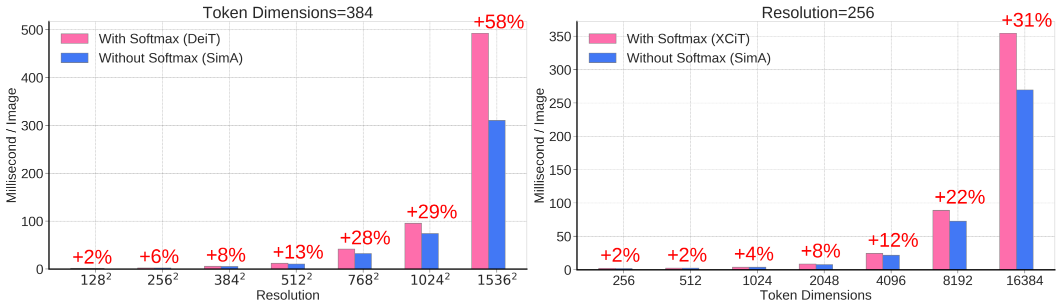

Figure 1: **Effect of Softmax on inference time:** We evaluate performance of each model on a single RTX 8000 GPU with batch size of 8. When comparing the baseline to our method (SimA), we fix the order of (QK^TV) to have the same dot product complexity as the baseline. For example, when comparing with DeiT, if $N > D$, then it is more efficient to do $\hat{Q}(\hat{K}^TV)$ for our method, but we do $(\hat{Q}\hat{K}^T)V$ to have same complexity as DeiT($O(N^2D)$). We do this to solely evaluate the effect of Softmax on the computation time. **Left:** We fix the token dimension to 384 and increase the image resolution. At 1536×1536 resolution, DeiT is 58% slower than our method due to the overhead of $exp(.)$ function in Softmax. **Right:** We fix the resolution and increase the capacity of the model (dimensions of Q and K). With 8192 dimensions, XCiT is 22% slower due to Softmax overhead.

equivariance property does not exist in some other models like MLP-Mixer Tolstikhin et al. (2021). Hence, instead of using MLP-Mixer that does not have Softmax by default, we are interested in removing Softmax from the original transformers to keep this permutation equivariance property.

We perform experiments with our simple attention block, denoted SimA, by using it in standard vision transformers, DeiT, CvT, and XCiT. Our method achieves on-par results with SOTA on ImageNet classification, MS-COCO object detection and segmentation, and also self-supervised learning.

In summary, our SimA attention block does not use Softmax, which makes it computationally efficient generally (see Fig. 1), and on the edge devices specifically. SimA can dynamically choose to be linear on N or D at the test time depending on the image resolution or the number of tokens. Changing Multi-head attention to Single-head one or changing GELU activation function to ReLU, has a very small effect on the accuracy of SimA. This makes SimA simple and effective for various applications.

2 Method

2.1 Background on Vision Transformers:

Self-Attention Block: The original vision transformer Dosovitskiy et al. (2020) uses the self-attention block introduced in Vaswani et al. (2017). Self-attention block gets $X \in \mathbb{R}^{N \times D}$ as the input where N is the number of tokens and D is the dimensionality of each token. Then $W_q \in \mathbb{R}^{D \times D}$, $W_k \in \mathbb{R}^{D \times D}$ and $W_v \in \mathbb{R}^{D \times D}$ projects X into three $N \times D$ matrices: query ($Q = XW_q$), key ($K = XW_k$) and value ($V = XW_v$). We calculate attention matrix $A \in \mathbb{R}^{N \times N}$ defined as $A = Softmax(QK^T/\sqrt{D})$ where $Softmax$ is applied to each row independently, so each row in A sums to one. Then, we calculate the output $O = AV$. Each row of $O \in \mathbb{R}^{N \times D}$ corresponds to one token and since rows of A sum to one, each token in a weighted average of the values of all tokens.

Additionally, Multi-Head Self-Attention (MSA) transformers divide Q, K, and V of each token into H heads, where each head has its own attention over the corresponding head in all tokens. For example, $Q = [Q_1; Q_2; ...; Q_H]$ where $Q_i \in \mathbb{R}^{N \times \frac{D}{H}}$ is the query matrix for the i'th head. Then, we calculate H self-attention for all heads in parallel and concatenate the outputs to get $O = [O_1; O_2; ...; O_H]$. Finally, the self-attention block has an additional output projection $W_{proj} \in \mathbb{R}^{D \times D}$, thus the final output of the self-attention block is OW_{proj} which is of size $\mathbb{R}^{N \times D}$.

Cross-covariance Attention Block (XCA): Vanilla self-attention block has a complexity of $O(DN^2)$ which is quadratic on N. Ali et al. (2021); Shen et al. (2021) introduce an attention mechanism that is linear on N. In XCA, we calculate the attention matrix with $A = K^TQ$ where A is a $D \times D$ matrix. Next, we apply Softmax on each columns, so that columns sum to one. Then we calculate output

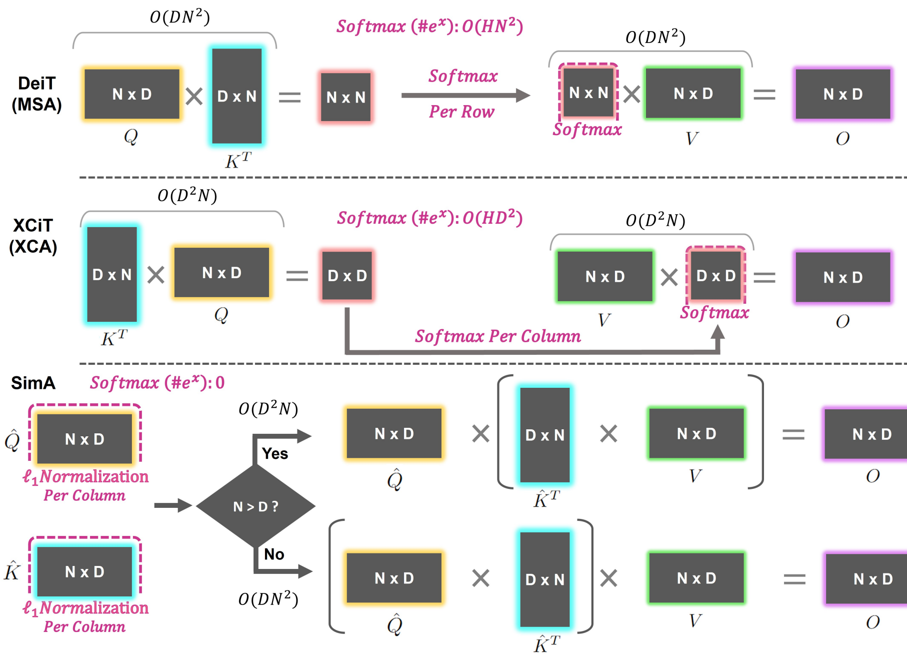

Figure 2: **Our Simple Attention (SimA):** First, we normalize each channel in Q and K with ℓ_1-norm across the tokens, to get $\hat{Q}$ and $\hat{K}$. Next, we can choose either $(\hat{Q}\hat{K}^T)V$ or $\hat{Q}(\hat{K}^T V)$ depending on the number of input tokens N. Compared to XCA and MSA, our method has following benefits: (1) It is free of Softmax, hence it is more efficient. (2) At test time we can dynamically switch between $(\hat{Q}\hat{K}^T)V$ and $\hat{Q}(\hat{K}^T V)$ based on the number of input tokens (e.g., different image resolution).

as $O = VA$. Note that A is an attention of channels on each other rather than tokens. Compared to vanilla self-attention (MSA), XCA has complexity of $O(D^2N)$. Since XCA is linear on N, it is more efficient when $N \gg D$ and it is less efficient when $N \ll D$.

Vision Transformer Block: Vision transformers architecture contains n consecutive Vision Transformer blocks. Each block has MSA block followed by a Feed-Forward Network (FFN) both with skip connection. FFN is a simple 2-layer MLP which projects tokens from D dimension to $4D$ and again back to D dimensions. FFN uses GELU Hendrycks & Gimpel (2016) as the activation function. Moreover, we use LayerNorm Ba et al. (2016) on each token before forwarding them through MSA or FFN blocks. The following two updating rules summarize each block of the vision transformer:

$$(Step1) \quad X \leftarrow X + MSA(\text{LayerNorm}_1(X)) \quad , \quad (Step2) \quad X \leftarrow X + FFN(\text{LayerNorm}_2(X))$$

2.2 Simple Attention (SimA):

Our main goal is to reduce the computation by removing the Softmax ($exp(.)$) layer. We believe one of the roles of the Softmax layer is to normalize the attention so that each token is a weighted average of the values of all tokens. This ensures that the attention values are bounded. Hence, we introduce an alternative normalization method that does not need a Softmax layer.

In the regular attention block, if a channel in Q and/or K has large values, that channel may dominate the dot product QK^T. This results in other channels being ignored in calculating the attention. We believe this may be one of the reasons leading to superior performance of the multi-head attention (MSA) compared to the single-head one. Since in MSA, the dominating channel can dominate a single head only leaving the other heads still operational. We propose a method to take this solution to the extreme where we can normalize each channel in Q and K across tokens so that different channels become more comparable. We do this by simply dividing the values of each channel by the ℓ_1 norm of that channel across all tokens:

$$\hat{Q}^i := \frac{Q^i}{|Q^i|_1} \quad \text{and} \quad \hat{K}^i := \frac{K^i}{|K^i|_1}$$

where Q^i is the i'th column of Q (values of the i'th channel for all tokens) and $\hat{Q}$ and $\hat{K}$ are the normalized query and key matrices. Given this simple normalization method, we remove the Softmax layer, so the attention block can be written as:

$$O = \hat{Q}\hat{K}^T V$$

where $O \in \mathbb{R}^{N \times D}$. Similar to standard transformers, we use this block for each head separately, concatenate the outputs, and finally apply the output projection OW_{proj}.

One can assume $\hat{Q}\hat{K}^T$ is the attention matrix that quantifies the effect of a token on another token. Interestingly, if the query and key vectors have a large angle, the attention values can become negative, meaning that a token can affect another one negatively. This is in contrast to regular transformers where the attention is always non-negative. A simple pseudo-code is provided in the appendix.

Due to our normalization, the attention values are bounded between $-D$ and D. The extremes happen when only a single row of Q and a single row of K are nonzero. In this case, all other tokens will have zero query and key vectors. One may divide the attention by D to bound it between -1 and 1. This constant scalar multiplier can be absorbed into W_v, the projection matrix for V.

The cost of Softmax: Both XCA and MSA use Softmax for normalization. Softmax needs running $exp(.)$ which is costly. MSA uses Softmax on a matrix of size $N \times N$ while XCA uses Softmax on a matrix of size $D \times D$. Hence, the order of $exp(.)$ operations is $O(HN^2)$ for MSA and $O(HD^2)$ for XCA. Therefore, Softmax will be bottleneck when increasing the number of tokens (higher image resolutions) in MSA and number of channels (higher capacity transformers) in XCA. On the other hand, our attention block does not use $exp(.)$ operation at all. Moreover, in the last row of Table 1, we show that changing GELU to ReLU in SimA gets comparable accuracy to the main experiment (79.6% vs 79.8%). This version of SimA does not use any $exp(.)$ operation at the inference time. The reduction in the computation cost of Softmax is shown in Fig. 1-left for increasing N and in Fig. 1-right for increasing D. Fig. 1 shows the speed-up due to removing Softmax only and does not include the speed-up due to changing the order of multiplications. We believe removing the cost of $exp(.)$ operation can have a large impact particularly in edge devices.

3 Related Work

Vision Transformers: Convolutional Neural Networks (CNNs) have become ubiquitous as the most commonly used network architecture for computer vision tasks Krizhevsky et al. (2012); He et al. (2016); Chollet (2017); Howard et al. (2017). Transformers have recently emerged as a promising alternative to CNNs. Transformers Vaswani et al. (2017) rely entirely on self-attention mechanism and was originally introduced for NLP tasks. ViT Dosovitskiy et al. (2020) adapts transformers to obtain convolution-free architecture for computer vision tasks by dividing each image in 16×16 patches and considering each patch as a token input. DeiT Touvron et al. (2021b) improves training efficiency of ViT on smaller dataset. The Scaled Dot-Product Attention module Vaswani et al. (2017) used by transformers rely on the softmax operation for normalization. Unlike CNNs/MLP Krizhevsky et al. (2012); Simonyan & Zisserman (2014); Touvron et al. (2021a); Tolstikhin et al. (2021) based architectures, softmax is an important part of transformer architecture. In this paper, we address replacing the softmax ($exp(.)$) operation in the self-attention module of vision transformers.

Efficient Vision Transformers: Transformers have a large memory footprint, so deploying them on edge devices with limited resources is difficult. LeViT Graham et al. (2021) uses down-sampling in stages to improve efficiency. Mehta & Rastegari (2021); Wu et al. (2021a) integrate convolution in transformer. Liu et al. (2021); Ho et al. (2019) improve the self-attention efficiency by limiting the attention of each token to subset of tokens. Lu et al. (2020) uses distillation to improve the efficiency of the network. Fayyaz et al. (2021); Rao et al. (2021); Marin et al. (2021) decrease the number of tokens by token pruning. Although these works limit the computation generally, softmax or $exp(.)$ function is still required to calculate the attention. Our idea is orthogonal to these methods since we can replace attention block in any transformer with our $exp(.)$ free attention block.

Linear Attention: Vanilla attention has $O(N^2D)$ computation and memory complexity, where N is number of input tokens and D is dimension of each token. Some works target this issue by replacing vanilla attention with a linear attention with $O(ND^2)$ complexity. XCiT Shen et al. (2021); Ali et al. (2021) uses attention across feature channels rather than tokens. Some works use similarity kernels to approximate softmax, thus it is possible to have linear complexity by doing $\phi(Q)(\phi(K)^T\phi(V))$ instead of $(\phi(Q)\phi(K)^T)\phi(V)$ where $\phi(x)$ is the kernel function. Katharopoulos et al. (2020) uses $\phi(x) = 1 + elu(x)$, whereas Lu et al. (2021); Peng et al. (2021) use Gaussian kernel functions. Xiong et al. (2021) use SVD decomposition and Choromanski et al. (2020) use positive random features to approximate softmax. Wang et al. (2020) approximate attention with a low rank matrix. All these methods either use exponential function. For example, SOFT Lu et al. (2021) removes Softmax without reducing the number of $exp(.)$ operations. Our ideas are different since we aim to remove the costly $exp(.)$ operation. Moreover, the focus of those methods is to have linear attention with respect to number of tokens which is not the main focus of this paper. A recent work in the NLP community, CosFormer Qin et al. (2022), passes Q and K through a ReLU unit and normalizes their product. It also adds a re-weighting method that improves the locality of the data using $sine(.)$ and $cosine(.)$ functions. Our idea is simpler and we apply it to visual recognition rather than NLP. Moreover, cosine re-weighting in CosFormer requires $4\times$ more FLOPs in K and V dot product compared to ours.

Softmax Approximation: Softmax is an expensive operation on hardware since it requires $exp(.)$ operation. More specifically, softmax in transformer architecture contributes to major part of computation when the input is large Stevens et al. (2021). Banerjee et al. (2020) approximates softmax with Taylor expansions, whereas Gao et al. (2020); Du et al. (2019); Ham et al. (2020); Zhu et al. (2020) target designing a hardware architecture to approximate softmax. Softermax Stevens et al. (2021) uses a low-precision implementation of 2^x. Zafrir et al. (2019) uses lower precision computation. Prato et al. (2019); Lin et al. (2020) use quantized softmax. While these works approximate Softmax at the hardware, we replace Softmax completely with ℓ_1 normalization at the model architecture.

4 Experiments

We evaluate effectiveness of SimA attention block by replacing self-attention in three popular vision transformer families: DeiT, XCiT and CvT. We evaluate our model on image classification, object detection, image segmentation, and self-supervised learning.

4.1 Image Classification

Dataset: We train on ImageNet1K Deng et al. (2009) and report Top-1 accuracy on the validation set.

Implementation Details: We use PyTorch Paszke et al. (2019) and Timm Wightman (2019) libraries to train our models with a setup similar to Ali et al. (2021); Touvron et al. (2021b). We use AdamW Loshchilov & Hutter (2017) optimizer. We train CvT and DeiT models with 300 epochs and XCiT models with 400 epochs. We set the batch size to 1024 and weight decay to 0.05. We use cosine scheduling with an initial learning rate of $5e-4$. We use Stochastic depth drop rate Huang et al. (2016) of 0.05. Data augmentations are the same as those in Touvron et al. (2021b) including Rand-Augment Cubuk et al. (2020), CutMix Yun et al. (2019) and Mixup Zhang et al. (2017). Following Ali et al. (2021); Touvron et al. (2021c), we train our models with images of resolution 224 and evaluate it using images with a crop ratio of 1.0. Training DeiT-S or XCiT-S12/16 with 8 RTX 6000 GPUs takes approximately 100 hours.

We use SimA along with the following three transformer architectures to show its generalization:

- **DeiT**: Touvron et al. (2021b) is a well-known transformer architecture based on ViT Dosovitskiy et al. (2020). We use DeiT-S which has the following settings: patch size$=16$, embedding dimensions$=384$, number of heads$=6$ and layers$=12$. Self-attention in DeiT has complexity of $O(DN^2)$ which is quadratic on the number of tokens N.

- **XCiT**: Ali et al. (2021) is a state-of-the-art vision transformer architecture with a linear attention. XCiT has 2 major differences compared to DeiT: (1) XCiT has Local Patch Interaction (LPI) in each block, which consists of one depth-wise 3×3 convolution followed by Batch Normalization, GELU and another depth-wise 3×3 convolution. (2) XCiT has separate class attention layers similar to Touvron et al. (2021c). The `CLS` token is added at the end of the initial self-attention stage and class attention layers are used to aggregate information from image tokens to the class token. This modification adds extra parameters and computation to the model.

We replace SimA in three variant of XCiT: XCiT-S12/16, XCiT-T12/8 and XCiT-T24/16. XCiT-S12/16 has a patch size of 16, embedding dimension of 384, 8 heads, 12 layers, and 2 class attention

Table 1: **ImageNet classification:** We denote replacing Softmax attention with SimA by X $\rightarrow$ SimA. Softmax column indicates the number of $exp(.)$ operations in the attention block. N is the number of tokens, D is the token dimension, H is the number of heads, M is the local window size, and R is the reduction ratio. We also report ResNet50 RA (with RandAug Cubuk et al. (2020)). Models indicated by * use teacher during training. EfficientNet outperforms our method, but it is a convolutional network and uses more FLOPs at higher image resolution. SOFT also has $exp(.)$ function in the backbone which is costly. Purple rows are our method while blue rows are comparable baselines. Our method is a $exp(.)$ free transformer and has on-par accuracy with SOTA transformers. To simplify SimA even further, we investigate two more variations in yellow rows: (1) Replacing GELU with ReLU, (2) Replacing multi-head attention with single head attention. Interestingly, SimA has comparable performance even with single head attention and ReLU. Note that the ReLU version does not need any $exp(.)$ operation at the inference time.

	Model	params	FLOPs	Resolution	Softmax/# exp	Top1-Acc
CNN	ResNet18 He et al. (2016)	12M	1.8B	224	0	69.8
Transformer	XCiT-T24/16 Ali et al. (2021)	12M	2.3B	224	HD^2	79.4
Transformer	XCiT-T24/16 $\rightarrow$ SimA	12M	2.3B	224	0	79.8
Transformer	XCiT-T12/8 Ali et al. (2021)	7M	4.8B	224	HD^2	79.7
Transformer	XCiT-T12/8 $\rightarrow$ SimA	7M	4.8B	224	0	79.4
CNN	ResNet50 RA Cubuk et al. (2020)	25M	3.9B	224	0	77.6
	EfficientNet-B5 RA Cubuk et al. (2020)	30M	9.9B	456	0	83.9
	RegNetY-4GF Radosavovic et al. (2020)	21M	4.0B	224	0	80.0
	ConvNeXt-T Liu et al. (2022)	29M	4.5B	224	0	82.1
MLP	ResMLP-S24 Touvron et al. (2021a)	30M	6.0B	224	0	79.4
	MS-MLP-T Zheng et al. (2022)	28M	4.9B	224	0	82.1
	Hire-MLP-S Guo et al. (2021)	33M	4.2B	224	0	82.1
Hybrid	Twin-SVT-S Chu et al. (2021)	24M	3.7B	224	HM^2N	81.7
	CvT-13 Wu et al. (2021b)	20M	4.5B	224	HN^2	81.6
	CvT-13 $\rightarrow$ SimA	20M	4.5B	224	0	81.4
Transformer	Swin-T Liu et al. (2021)	29M	4.5B	224	HM^2N	81.3
	PVT-S Wang et al. (2021)	24M	4.0B	224	HN^2/R	79.8
	T2T-ViT-14 Yuan et al. (2021)	21M	5.2B	224	HN^2	80.7
	CaiT-XS24* Touvron et al. (2021c)	26M	19.3B	384	HN^2	84.1
	SOFT-S Lu et al. (2021)	24M	3.3B	224	HN^2	82.2
	DeiT-S* Touvron et al. (2021b)	22M	4.6B	224	HN^2	81.2
	XCiT-S12/16*Ali et al. (2021)	26M	4.8B	224	HD^2	83.3
	DeiT-S Touvron et al. (2021b)	22M	4.6B	224	HN^2	79.8
	XCiT-S12/16Ali et al. (2021)	26M	4.8B	224	HD^2	82.0
	DeiT-S $\rightarrow$ SimA	22M	4.6B	224	0	79.8
	XCiT-S12/16 $\rightarrow$ SimA	26M	4.8B	224	0	82.1
Multi-Head/GELU	DeiT-S $\rightarrow$ SimA	22M	4.6B	224	0	79.8
Multi-Head $\rightarrow$ Single-Head	DeiT-S $\rightarrow$ SimA	22M	4.6B	224	0	79.4
GELU $\rightarrow$ ReLU	DeiT-S $\rightarrow$ SimA	22M	4.6B	224	0	79.6

layers. XCiT-T12/8 is similar to XCiT-S12/16 with a patch size of 8, embedding dimension of 192, and 4 heads. XCiT-T24/16 is similar to XCiT-T12/8 with patch size of 16.

- **CvT**: We apply SimA to CvT Wu et al. (2021b), which is a SOTA hybrid convolution/transformer architecture. CvT has 3 stages. Each stage has a Convolution Token Embedding layer followed by transformer blocks. We use CvT-13 in our experiments which 13 blocks in total.

Results on ImageNet: We replace MSA and XCA blocks with our SimA block in DeiT, CvT and XCiT respectively, and train our models on ImageNet. Note that we train our models from scratch without distillation from a teacher. Results are in Table 1. In XCiT models, we get comparable results when replacing XCA block with SimA block. Compared to DeiT-S, our attention block performs on-par with DeiT-S. Moreover, our method with no Softmax layer, achieves comparable accuracy (0.2 point lower) compared to CvT-13. This suggests that one can replace attention block with SimA in these standard SOTA transformers without degrading their performance. Since SimA is $exp(.)$ free, it has the advantage over regular attention architectures in terms of efficiency and simplicity.

Comparison to Linear Attention: We compare SimA with other Linear Attention methods in NLP literature in Table 2. We train all methods with ImageNet-1K training set and report Top-1 accuracy on ImageNet-1K validation set. SimA has better or on-par accuracy compared to other methods. Additionally, SimA is $exp(.)$ free which is the main goal of this work.

Table 2: **Linear Attention Comparison:** We compare SimA with previous linear attention methods introduced in NLP. We report ImageNet Top-1 validation accuracy. Note that the focus of these methods is to have linear attention with respect to the number of tokens while the main focus of SimA is to remove $exp(.)$ operation. * CosFormer is originally in NLP. We ran multiple versions of CosFormer with cosine re-weighting (multiple learning rates and weight decays) for the vision task, however, none of them converged. Moreover, CosFormer with cosine re-weighting requires $4\times$ more FLOPs compared to SimA in multiplying Q, K, and V matrices. More details are in the appendix.

Model	params	FLOPs	Softmax/#exp	Top1-Acc
Transformer Vaswani et al. (2017)	13M	3.9B	HN^2	79.1
Linformer Wang et al. (2020)	13M	1.9B	HN	78.2
Performer Choromanski et al. (2020)	13M	2.2B	ND	76.1
Nyströmformer Xiong et al. (2021)	13M	2.0B	HN	78.6
SOFT Lu et al. (2021)	13M	1.9B	HN^2	79.3
XCiT-T20/16 $\rightarrow$ SimA	12M	1.9B	0	79.2
XCiT w/ Efficient Attention Shen et al. (2021)	22M	4.8B	ND	80.9
CosFormer w/o re-weighting * Qin et al. (2022)	22M	4.8B	0	76.1
XCiT-S12/16 $\rightarrow$ SimA	22M	4.8B	0	82.1

Table 3: **Transfer to MS-COCO dataset:** Models with * are pretrained with a teacher on ImageNet. Swin-T has more parameters and Softmax overhead. XCiT-S12/8 has $4\times$ more tokens. Our method is $exp(.)$ free, thus it is more efficient for high resolution images and high capacity models (Fig. 1).

			Detection			Segmentation		
Backbone	params	$exp(.)$	AP^{box}	AP$^{box}_{50}$	AP$^{box}_{75}$	AP^{mask}	AP$^{mask}_{50}$	AP$^{mask}_{75}$
ResNet50 He et al. (2016)	44.2M	✗	41.0	61.7	44.9	37.1	58.4	40.1
PVT-Small Wang et al. (2021)	44.1M	✓	43.0	65.3	46.9	39.9	62.5	42.8
ViL-Small Zhang et al. (2021)	45.0M	✓	43.4	64.9	47.0	39.6	62.1	42.4
Swin-T Liu et al. (2021)	47.8M	✓	46.0	68.1	50.3	41.6	65.1	44.9
XCiT-S12/16*	44.3M	✓	45.3	67.0	49.5	40.8	64.0	43.8
XCiT-S12/8*	43.1M	✓	47.0	68.9	51.7	42.3	66.0	45.4
XCiT-S12/16	44.3M	✓	45.0	66.7	48.9	40.5	63.6	43.2
XCiT-S12/16 $\rightarrow$ SimA	44.3M	✗	44.8	66.5	48.8	40.3	63.2	43.3

4.2 Transfer To Object Detection and Semantic Segmentation

As shown in Fig. 1 and Stevens et al. (2021), softmax operation represents a large fraction of runtime in vision transformers, especially when the image resolution is high. In object detection and segmentation tasks we usually forward high resolution images. We demonstrate the transferability of SimA to these dense prediction tasks by fine-tuning our ImageNet pretrained model on them.

Dataset: We use MS-COCO Lin et al. (2014) dataset for these tasks. MS-COCO has 118K training images and 5K validation images with 80 categories. Images are annotated with bounding boxes and semantic segmentation masks.

Implementation Details: We follow Ali et al. (2021); Liu et al. (2021); Chen et al. (2019) for the setup and implementation. We use our pretrained model as the backbone of Mask RCNN He et al. (2017). Similar to Ali et al. (2021), we use FPN Lin et al. (2017) to extract features from layers 4, 6, 8 and 12 of the transformer. We use AdamW Loshchilov & Hutter (2017) optimizer with a learning rate of $1e-4$ and weight decay 0.05. We train our model for 36 epochs with batch size of 16 on 8 RTX2080Ti GPUs. Training takes 36 hours.

Results on MS-COCO: We compare our XCiT-S12/16 $\rightarrow$ SimA model with other vision transformers and ResNet in Table 3. We report the performance on the minival set. For a fair comparison, we limit the comparison to all models which are initialized with ImageNet1K pretrained backbones and trained with the same training time budget (3x schedule) on MS-COCO dataset. In comparison to other transformers, our method gets on-par performance while it is free of Softmax overhead on high resolution images or high capacity models (refer to Fig. 1).

4.3 Self-Supervised Learning

To show the generalizability of SimA, we train our SimA model on a pretext task for self-supervised learning (SSL). We use the non-contrastive task called DINO Caron et al. (2021) for SSL pre-training.

Table 4: **Self-Supervised Learning:** We train SimA attention block with DINO (SSL). Our method achieves performance comparable to transformer models with Softmax and trained for 100 epochs. Note that methods with different SSL task and higher number of epochs are not directly comparable.

SSL Method	Model	params	epochs	$exp(.)$	FLOPs	Linear	k-NN
ISD Tejankar et al. (2021)	ResNet50	25M	200	✗	3.9B	69.8	62.0
MoCo v2 He et al. (2020)	ResNet50	25M	200	✗	3.9B	69.9	-
MSF Koohpayegani et al. (2021)	ResNet50	25M	200	✗	3.9B	72.4	64.9
BYOL Grill et al. (2020)	ResNet50	25M	1000	✗	3.9B	74.3	66.9
MoBY Xie et al. (2021)	Swin-T	29M	300	✓	4.5B	75.0	–
DINO Caron et al. (2021)	ResNet-50	23M	300	✗	4.1B	74.5	65.6
DINO Caron et al. (2021)	ResMLP-S24	30M	300	✗	6.0B	72.8	69.4
DINO Caron et al. (2021)	ViT-S/16	22M	300	✓	4.6B	76.1	72.8
DINO Caron et al. (2021)	XCiT-S12/16	26M	300	✓	4.9B	77.8	76.0
DINO Caron et al. (2021)	ViT-S/16	22M	100	✓	4.6B	74.0	69.3
DINO Caron et al. (2021)	XCiT-S12/16	26M	100	✓	4.9B	75.8	71.6
DINO Caron et al. (2021)	XCiT-S12/16 $\rightarrow$ SimA	26M	100	✗	4.9B	75.5	71.2

We train our model on ImageNet train set ($1.2M$) without the use of ground-truth labels. DINO training is relatively expensive since it requires forwarding multi-crop augmentation through two models. Due to limited resources, we train our model and the baselines for 100 epochs. To train our XCiT-S12/16 $\rightarrow$ SimA model with DINO, we follow the training configuration of XCiT-S12/16 from the official repository of DINO Caron et al.. Similar to DINO, we use AdamW optimizer in PyTorch library with initial learning rate of 0.00025 with cosine scheduling. We use initial weight decay of 0.04 and increase it to 0.4 with cosine scheduling. We train for 100 epochs with minibatches of size 256. The training takes approximately 100 hours on four RTX-3090 GPUs. We use similar settings for training our method and the baseline (XCiT-S12/16).

Results of SSL training: Following Caron et al. (2021); Abbasi Koohpayegani et al. (2020), we report k-NN and Linear evaluation metrics for evaluating the SSL models. For k-NN evaluation, we forward images of training and validation set through the frozen backbone and extract features. We report 20-NN on the validation set. For Linear evaluation, we freeze the backbone and train a linear layer on extracted features from the frozen backbone and report Top-1 accuracy on the ImageNet validation set. We adopt a similar approach to DINO Caron et al. (2021) for extracting features from XCiT architecture. We extract the classification tokens of the last two class attention layers and global average pooling of the last two regular attention layers. Each of those 4 vectors is of size 384. We concatenate them and train a linear layer of size 4×384 to 1000 classes of ImageNet1K. We use similar training settings as DINO to train a linear layer for both our method and the baseline (XCiT-S12/16). We train for 100 epochs with SGD optimizer and the following settings: learning rate: 0.001 with cosine scheduling, batch size: 1024, and weight decay: 0. Results are shown in Table 4. Our $exp(.)$ free method performs comparably with the baselines with 100 epochs of training.

4.4 Single-head vs Multi-head Attention

In the regular attention block, if a channel in Q and/or K has large values, that channel may dominate the dot product QK^T. We believe multi-head attention (MSA) mitigates this issue to some degree by containing the dominant channel in one head only so that the other heads can have reasonable effect in the final attention. In SimA, by doing ℓ_1 normalization of each channel in Q and K across tokens, different channels become more comparable in the dot product QK^T, so multi-head attention may not have a large effect anymore. To evaluate our hypothesis empirically, we train both DeiT-S $\rightarrow$ SimA and DeiT-S with single head attention only. Results are in Table. 5. Interestingly, we show that with single-head attention, our method gets comparable results (0.4 point lower) while the accuracy of DeiT-S drops by 2.8 points. This suggests that unlike the vanilla attention block, multi-head attention is not critically important in SimA, which leads to simplicity SimA even further.

Table 5: **Effect of Removing Multi-Head Attention:** In single head variation, our method degrades much less compared to DeiT probably due to normalization of Q and K.

Model	DeiT-S $\rightarrow$ SimA		DeiT-S	
Attention Heads	6 (Multi-Head)	1 (Single)	6 (Multi-Head)	1 (Single)
ImageNet Top-1 acc.	79.8	79.4 (-0.4)	79.8	77.0 (-2.8)

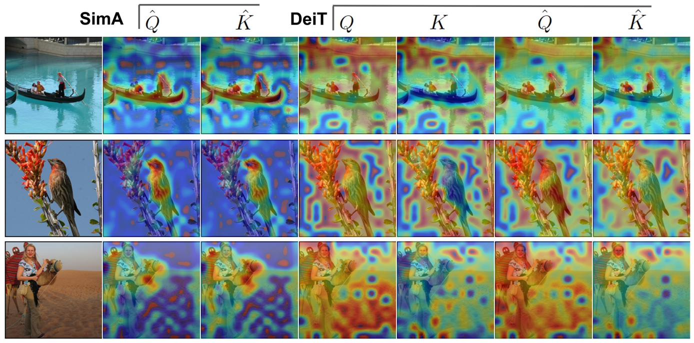

Figure 3: **Our method (SimA):** We extract $\hat{Q}$ and $\hat{K}$ from layer 12 of transformer. We get ℓ_2-norm of each token for $\hat{Q}$ and $\hat{K}$, normalize it to range [0,1] and overlay it as a heatmap on the image. Interestingly, magnitude of tokens represent the significance of tokens in our method. We provide more examples in the appendix.

4.5 Replacing GELU with ReLU

Similar to Softmax function, GELU activation function also uses $exp(.)$ operation, which is costly. We replace all GELU activation functions in DeiT-S → SimA with ReLU. We observe that DeiT-S → SimA with ReLU gets accuracy of 79.6 which is only 0.2 points lower than DeiT-S → SimA with GELU activation function. Note that SimA with ReLU does not use any $exp(.)$ operation at the inference time, leading to further efficiency of the model. Results are in Table 1 (yellow rows).

4.6 Effect of ℓ_1 Normalization

To see the effect of ℓ_1 normalization, we train our model without normalizing Q and K. We use XCiT-S12/16 → SimA with the same hyperparameters as our main experiment in Section 4.1. Note that without normalization, the range of QK^T can be from $-\infty$ to $+\infty$. None of our several trials converged as the training becomes unstable and results in a frequent NaN loss. Moreover, we replace ℓ_1 with ℓ_2 normalization, results in 2.9 points drop in accuracy (82.1% vs 79.2%).

4.7 Visualization

The dot product $\hat{Q}\hat{K}^T$ is correlated with the magnitude of $\hat{Q}$ and $\hat{K}$ vectors. Hence, we believe this magnitude can highlight the important tokens or image regions. This can be seen as a form of explanation or saliency map. First, we extract $\hat{Q}$ and $\hat{K}$ in the last layer of transformer (layer 12). Then, we calculate the ℓ_2-norm of $\hat{Q}$ along the channel dimension to get a single non-negative scalar for each token. We reshape this $N \times 1$ vector to the image shape, up-sample it to original image size, normalize it to range $[0, 1]$, and overlay it on the image as a heatmap. We repeat the same for $\hat{K}$. As shown qualitatively in Fig. 3, such a visualization highlights the important regions of the image. Moreover, we study the same visualization on standard DeiT with the ℓ_2-norm of both Q and $\hat{Q}$ and get a relatively flat heatmap. Note that we report these results for qualitative understanding of the model and do not evaluate it quantitatively or compare it with other network explanation methods. Also, note that the comparison with DeiT is not fair since in DeiT, Q and K are not necessarily comparable as the normalization happens in the Softmax operation after multiplying them.

5 Conclusion

We introduced SimA, a simple attention block that does not involve $exp(.)$ operation, to reduce the computational cost of transformers particularly at edge devices. SimA performs normalization on key and query matrices before multiplying them, enabling dynamically switching between $O(DN^2)$ or $O(D^2N)$ depending on the number of tokens (e.g., image resolution). Our extensive experiments show that while reducing the cost of inference, SimA achieves on-par results compared to SOTA methods on various benchmarks including ImageNet classification, MS-COCO object detection and segmentation, and self-supervised learning. Moreover, a single-head variation of SimA, which is even simpler, achieves on-par accuracy compared to SOTA multi-head attention models. We believe SimA can encourage research in this direction leading to easier adoption of transformers on edge devices with limited resources.

6 Ethics Statement:

Our core motivation is to reduce the computation at the inference, which has positive societal impacts including democratizing AI by reducing the need for computational resources and reducing the carbon footprint of the models at the inference time. However, similar to most AI methods, it can be harmful at the hands of the adversary. It can enable running transformers on the low-cost cameras that may be used in military or surveillance applications.

7 Reproducibility Statement:

To make our work reproducible, we report details of the implementation for each section. For all of our experiments, we report details of the training in Implementation Details. We include all hyperparameters we used for training our method. Moreover, we include details about the resources we used for training. We include the source code in the supplementary as well.

References

Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Compress: Self-supervised learning by compressing representations. *Advances in Neural Information Processing Systems*, 33: 12980–12992, 2020. 8

Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *Advances in neural information processing systems*, 34, 2021. 2, 4, 5, 6, 7, 15

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

Kunal Banerjee, Rishi Raj Gupta, Karthik Vyas, Biswajit Mishra, et al. Exploring alternatives to softmax function. *arXiv preprint arXiv:2011.11538*, 2020. 5

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Pytorch implementation of dino. `https://github.com/facebookresearch/dino`. 8

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021. 7, 8

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 7

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017. 4

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. *arXiv preprint arXiv:2009.14794*, 2020. 5, 7

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 M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 9355–9366. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/4e0928de075538c593fbdabb0c5ef2c3-Paper.pdf`. 6

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020. 5, 6

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. 5

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 4, 5

Gaoming Du, Chao Tian, Zhenmin Li, Duoli Zhang, Yongsheng Yin, and Yiming Ouyang. Efficient softmax hardware architecture for deep neural networks. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, pp. 75–80, 2019. 5

Mohsen Fayyaz, Soroush Abbasi Kouhpayegani, Farnoush Rezaei Jafari, Eric Sommerlade, Hamid Reza Vaezi Joze, Hamed Pirsiavash, and Juergen Gall. Ats: Adaptive token sampling for efficient vision transformers. *arXiv preprint arXiv:2111.15667*, 2021. 4

Yue Gao, Weiqiang Liu, and Fabrizio Lombardi. Design and implementation of an approximate softmax layer for deep neural networks. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. IEEE, 2020. 5

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12259–12269, 2021. 4

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 8

Jianyuan Guo, Yehui Tang, Kai Han, Xinghao Chen, Han Wu, Chao Xu, Chang Xu, and Yunhe Wang. Hire-mlp: Vision mlp via hierarchical rearrangement. *arXiv preprint arXiv:2108.13341*, 2021. 6

Tae Jun Ham, Sung Jun Jung, Seonghak Kim, Young H Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W Lee, et al. A^ 3: Accelerating attention mechanisms in neural networks with approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 328–341. IEEE, 2020. 5

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 4, 6, 7

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017. 7

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020. 8

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021. 1, 15

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 3

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019. 4

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. 4

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016. 5

Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li, and Torsten Hoefler. Data movement is all you need: A case study on optimizing transformers. *Proceedings of Machine Learning and Systems*, 3:711–732, 2021. 1

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020. 5

Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Mean shift for self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10326–10335, 2021. 8

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 4

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014. 7

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017. 7

Ye Lin, Yanyang Li, Tengbo Liu, Tong Xiao, Tongran Liu, and Jingbo Zhu. Towards fully 8-bit integer inference for the transformer model. *arXiv preprint arXiv:2009.08034*, 2020. 5

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021. 4, 6, 7

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5, 7

Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weiguo Gao, Chunjing Xu, Tao Xiang, and Li Zhang. Soft: Softmax-free transformer with linear complexity. *Advances in Neural Information Processing Systems*, 34, 2021. 5, 6, 7

Wenhao Lu, Jian Jiao, and Ruofei Zhang. Twinbert: Distilling knowledge to twin-structured bert models for efficient retrieval. *arXiv preprint arXiv:2002.06275*, 2020. 4

Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. Token pooling in vision transformers. *arXiv preprint arXiv:2110.03860*, 2021. 4

Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer, 2021. 4

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 5

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021. 5

Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. Fully quantized transformer for machine translation. *arXiv preprint arXiv:1910.10485*, 2019. 5

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. *arXiv preprint arXiv:2202.08791*, 2022. 5, 7, 16

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10428–10436, 2020. 6

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34, 2021. 4

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3531–3539, 2021. 2, 4, 7

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

Jacob R Stevens, Rangharajan Venkatesan, Steve Dai, Brucek Khailany, and Anand Raghunathan. Softermax: Hardware/software co-design of an efficient softmax for transformers. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 469–474. IEEE, 2021. 1, 5, 7

Ajinkya Tejankar, Soroush Abbasi Koohpayegani, Vipin Pillai, Paolo Favaro, and Hamed Pirsiavash. Isd: Self-supervised learning by iterative similarity distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9609–9618, October 2021. 8

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34, 2021. 2, 4

Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021a. 4, 6

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021b. 4, 5, 6

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 32–42, 2021c. 5, 6

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 4, 7

Ihor Vasyltsov and Wooseok Chang. Efficient softmax approximation for deep neural networks with attention mechanism. *arXiv preprint arXiv:2111.10770*, 2021. 1

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 5, 7

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 568–578, 2021. 6, 7

Ross Wightman. Pytorch image models, 2019. 5

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22–31, October 2021a. 4

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22–31, 2021b. 6

Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. Self-supervised learning with swin transformers. *arXiv preprint arXiv:2105.04553*, 2021. 8

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. 2021. 5, 7

Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 558–567, October 2021. 6

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019. 5

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pp. 36–39. IEEE, 2019. 5

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 5

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2998–3008, October 2021. 7

Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Mixing and shifting: Exploiting global and local dependencies in vision mlps. *arXiv preprint arXiv:2202.06510*, 2022. 6

Danyang Zhu, Siyuan Lu, Meiqi Wang, Jun Lin, and Zhongfeng Wang. Efficient precision-adjustable architecture for softmax function in deep learning. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(12):3382–3386, 2020. 5

A Appendix

A.1 Simple Pseudocode of SimA:

Since our method is simple, we include the pseudocode of SimA:

Algorithm 1 Pseudocode of SimA (Single Head) in a PyTorch-like style.

```
# self.qkv: nn.Linear(dim, dim * 3, bias=qkv_bias) ; query, key, value projection
# self.proj: nn.Linear(dim, dim, bias=output_proj_bias) ; output projection

def forward(self, x):
   B, N, D = x.shape # B: batch size, N: number of Tokens, D: Dimension of Tokens
   qkv = self.qkv(x).reshape(B, N, 3, D).permute(2, 0, 1, 3) # (3 x B x N x D)
   q, k, v = qkv[0], qkv[1], qkv[2] # split into query (B x N x D), key (B x N x D) and
        value (B x N x D)

   k = torch.nn.functional.normalize(k, p=1.0, dim=-2) # Normalized query (B x N x D)
   q = torch.nn.functional.normalize(q, p=1.0, dim=-2) # Normalized key (B x N x D)

   if (N/D) < 1:
      x = (q @ k.transpose(-2, -1)) @ v # (B x N x D)
   else:
      x = q @ (k.transpose(-2, -1) @ v) # (B x N x D)

   x = self.proj(x) # Output (B x N x D)
   return x
```

Visualization: Figure 4 provides more results similar to Figure 3. Please see Section 4.7 for details.

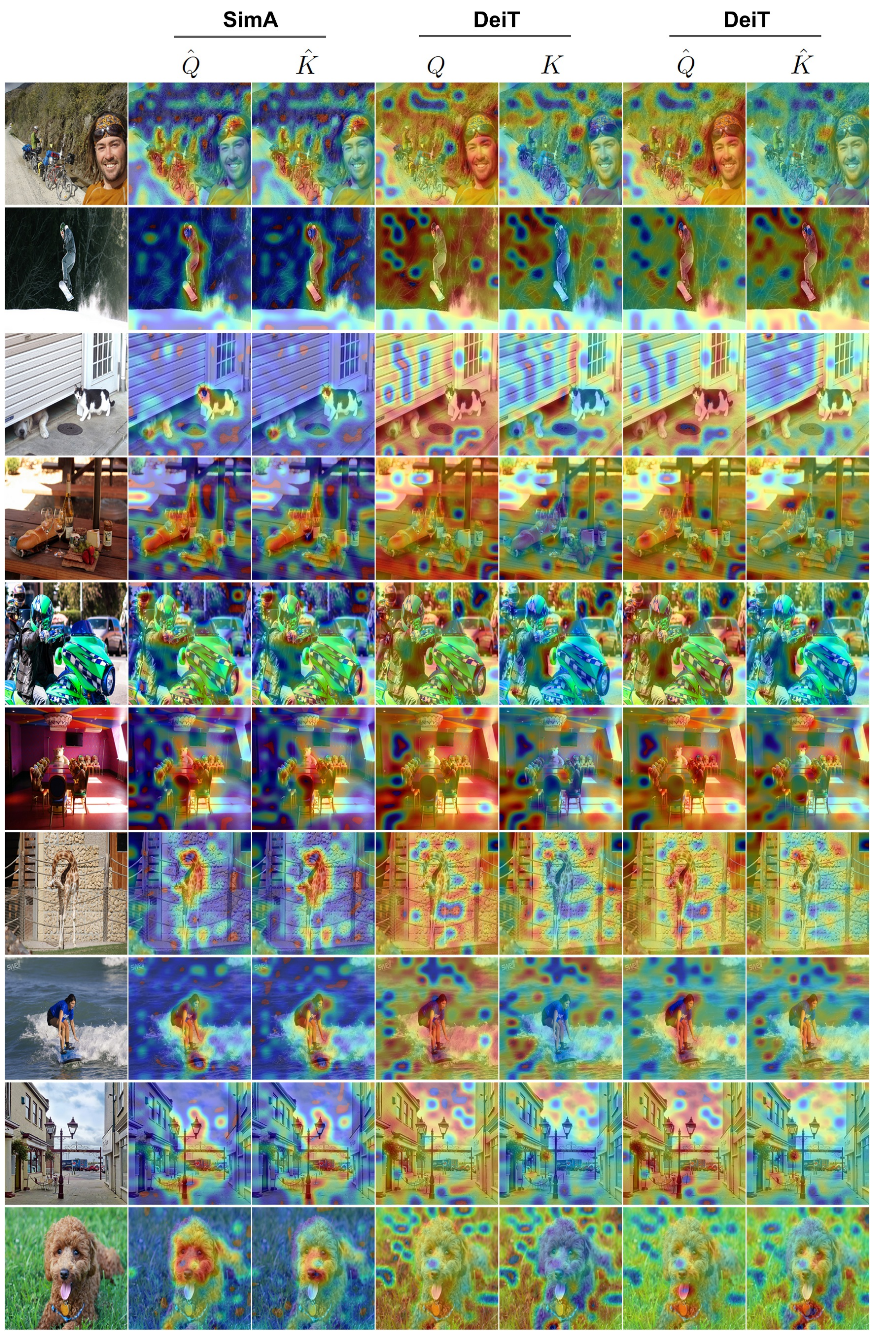

Figure 4: **Our method (SimA):** We extract $\hat{Q}$ and $\hat{K}$ from layer 12 of transformer. We get ℓ_2-norm of each token for $\hat{Q}$ and $\hat{K}$, normalize it to range [0,1] and overlay it as a heatmap on the image. Interestingly, magnitude of tokens represent the significance of tokens in our method. Note that all images are randomly selected from MS-COCO test set without any visual inspection or cherry picking.

A.2 SimA without LPI:

Although XCiT Ali et al. (2021) shows that LPI layer can improve the accuracy by 1.2 point, it limits the application of vanilla transformer (e.g., running masked auto encoder models like MAE He et al. (2021) is not straightforward). To show that our method is not dependent on LPI, we train our model without LPI. We observe that the accuracy drops by 1.2 point (82.1% vs 80.9%). Hence, although LPI boosts the accuracy, our method has comparable performance without LPI.

A.3 Details of Linear Attention Comparison:

CosFormer with cosine re-weighting requires $4\times$ more FLOPs compared to SimA in multiplying K and V matrices. Since CosFormer is developed for NLP, it assumes one dimensional indexing for the tokens. However, applying it to vision, we need to index the tokens with two indices to take advantage of the induced locality. To do so, one may introduce two cosine weights to Eq 10 of CosFormer Qin et al. (2022): one in x direction and the other one in y direction to come up with:

$$Q_{i,m}K_{j,n}^{T}cos(i-j)cos(m-n)$$

which can be expanded to:

$$Q_{i,m}K_{j,n}^{T}\Big(cos(i)cos(j)+sin(i)sin(j)\Big)\Big(cos(m)cos(n)+sin(m)sin(n)\Big)$$

which can be regrouped to:

$$\begin{aligned}=&\Big(Q_{i,m}cos(i)cos(m)\Big)\Big(K_{j,n}^{T}cos(j)cos(n)\Big)+\Big(Q_{i,m}cos(i)sin(m)\Big)\Big(K_{j,n}^{T}cos(j)sin(n)\Big)\\+&\Big(Q_{i,m}sin(i)cos(m)\Big)\Big(K_{j,n}^{T}sin(j)cos(n)\Big)+\Big(Q_{i,m}sin(i)sin(m)\Big)\Big(K_{j,n}^{T}sin(j)sin(n)\Big)\end{aligned}$$

Hence, for every attention value, CosFormer needs 4 dot products between Q and K vectors while our method needs only one dot product. Hence, following Eq. 12 of the CosFormer paper, CosFormer needs 4 times more FLOPS compared to our method in calculating the attention values (multiplying Q, K, and V matrices).