Rethinking the Self-Attention in Vision Transformers

Kyungmin Kim[1]* Bichen Wu[2] Xiaoliang Dai[2] Peizhao Zhang[2] Zhicheng Yan[2]
Peter Vajda[2] Seon Joo Kim[1]

[1]Yonsei University [2]Facebook

Abstract

Self-attention is a corner stone for transformer models. However, our analysis shows that self-attention in vision transformer inference is extremely sparse. When applying a sparsity constraint, our experiments on image (ImageNet-1K) and video (Kinetics-400) understanding show we can achieve 95% sparsity on the self-attention maps while maintaining the performance drop to be less than 2 points. This motivates us to rethink the role of self-attention in vision transformer models.

1. Introduction

After the success of Transformers in natural language processing (NLP) [8, 4], Transformers have been adapted to several Computer Vision problems and shown successful results [9, 2, 5, 1]. Specifically, ViT [5] suggested purely Transformers-based network for image classification and TimeSFormer [1] for action classification.

While self-attention is effective, one concern is its scalability. Since the self-attention compares all pairs of elements, it results in a quadratic computational complexity to the number of tokens. When it comes to video inputs, the number of floating-point operations (FLOPs) gets much larger because of the additional temporal dimension.

To this end, we suggest a way to apply masking mechanism into attention map computation so that the comparison happens sparsely. This is based on the observation that some Transformer heads in earlier layers tend to attend on local regions rather than globally. In this case, computing similarity score for all the comparing elements leads to a waste of computation.

Our goal is to find the optimal mask patterns (with high masking/sparsity ratio) thus to minimize the performance drop. We generate mask patterns based on data-driven attention map distribution and show that ViT with data-driven masks can retain the loss of performance less than 2 points even with 95% sparsity ratio.

2. Analysis on the self-attention of ViTs

We first analyze the average attention maps of trained ViT models. For an image model, we use DeiT-Base [7] pre-trained on ImageNet-1K [3] and compute average attention maps over 1,281,168 ImageNet training samples. For a video model, we use TimeSFormer (ST) pre-trained on Kinetics-400 [6] and use Kinetics-400 training videos to obtain average attention maps. We provide detailed visualization and analysis as follows.

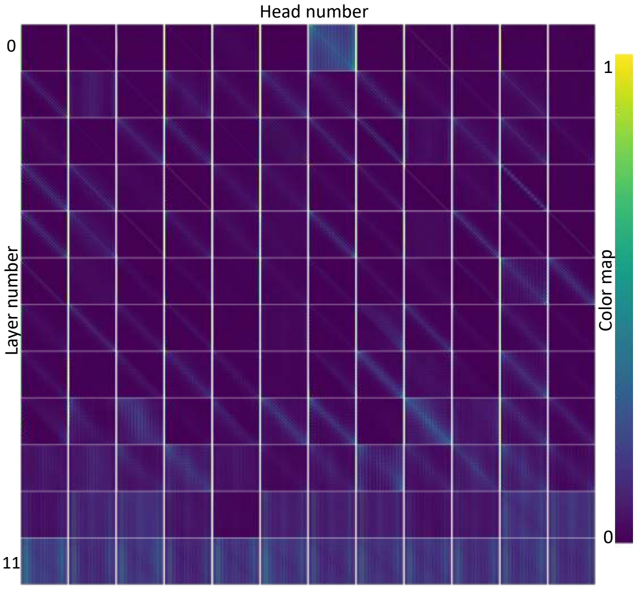
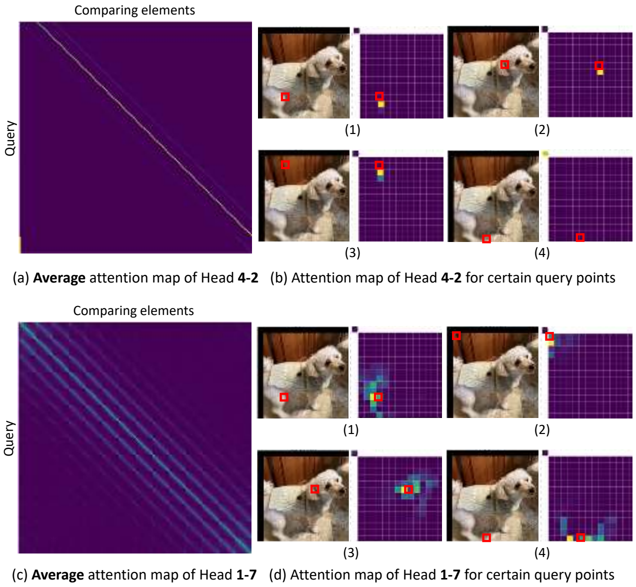


Figure 1: Average attention maps of 144 heads (12 heads×12 layers) of the image understanding model (DeiT-Base). Each map is size of 196 × 197. Zoom in for details. Best view in color.

2.1. Image Classification Model

Fig. 1 shows average attention map of 144 heads (12 heads×12 layers), where we normalize each attention map (by scaling between 0 and 1) for better visualization. First, we notice that the attention maps are very sparse, as most of the elements are zero. Furthermore, we find several patterns

* This work is done while working at Facebook.



(a) **Average** attention map of Head **4-2** (b) Attention map of Head **4-2** for certain query points

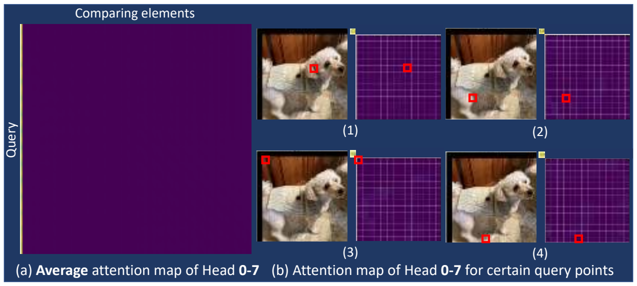
(c) **Average** attention map of Head **1-7** (d) Attention map of Head **1-7** for certain query points

Figure 2: Examples of heads which attend based on relative position to a query point. (Left) **Average** attention map of a specific head. (Right) A sample image and an attention map of each query. Red box indicates a query point, [0,0] in the attention map is a score for `cls_token`. Yellow color represents a value close to 1, which means highly activated, while purple close to 0.

in attention map distribution and divide them into three different types: 1) relative position based attention, 2) absolute position based attention, and 3) contents based attention. In the following, we zoom into several heads with distinctive behaviors.

Relative position based attention: We show two heads with relative position based attention in Fig. 2 (Head 4-2 and 1-7). Specifically, Head 4-2 is specialized for a token located one below. Interestingly, when there is no token below (Fig. 2 (b)-4), it attends to `cls_token` instead. On the other hand, the average attention map of Head 1-7 (Fig. 2 (c)) shows Convolution-like patterns which have high activation score on horizontal and vertical neighborhoods. Note that since an attention map is dynamically computed depending on query and key values, it is less structured and varies from query to query whereas Convolutions have structured and fixed filters shared over all the positions. Heads of this type typically attend to local region nearby a query and yield sparse attention map where few tokens have high activation score while others remain close to zero.

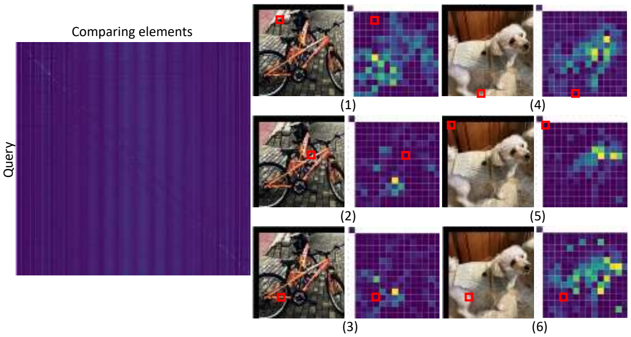
Absolute position based attention: This attention type refers to a head which attends to a certain absolute position regardless of a query. This can be identified by a vertical line with high activation score in the average attention map. Fig. 3 shows that in Head 0-7 all queries have exceptionally high activation score to the special token, *i.e.* `cls_token`. Note that in ViT models every token is added with position embeddings so it is possible to identify a token in a certain position and have a high score to a specific position.



(a) **Average** attention map of Head **0-7** (b) Attention map of Head **0-7** for certain query points

Figure 3: An example of heads attending to a certain absolute position.

Content-based attention: Fig. 4 presents one head that yields attention map based on contents. In Head 9-1, queries tend to have high activation score nearby a salient object. Some queries attend over the whole object ((b)-1, (b)-4) while other queries attend to a specific part of an object ((b)-2, (b)-5). Since a salient object can be located in any positions and can be any size, the average attention map (Fig. 4 (a)) shows that the score is spread over a wide range of tokens rather than peaky in a sparse, local region.

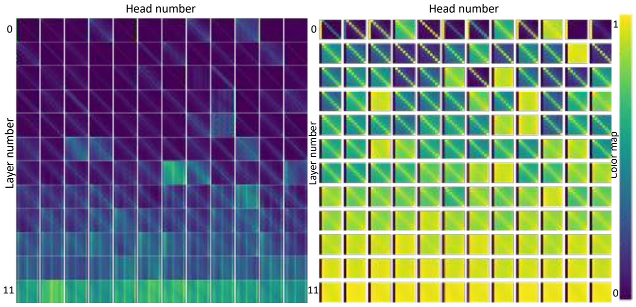


(a) **Average** attention map of Head **9-1** (b) Attention map of Head **9-1** for certain query points

Figure 4: An example of heads attending based on contents.

2.2. Video Classification Model

We present the attention map distribution of a video model (TimesFormer ST) in Fig. 5. In the video model, due to the additional temporal dimension, it has a larger number of tokens (1+14×14×8=1,569 tokens in our case) than the image model. Due to space limit, we present separate attention map for spatial dimension and temporal dimension. In order to reduce the dimensions, we sum up the comparing elements along the dimension of the interest, and average over the query points. For example, the original attention map excluding `cls_token` is size of (1568, 1568). It can be reshaped into (8,14,14, 8,14,14) where the first three dimension is for query tokens while the last three represents comparing tokens. In order to have spatial map, we apply mean over the first dimension and sum over the fourth dimension, *i.e.*, (mean, 14, 14, sum, 14, 14), resulting in (14, 14, 14, 14) which can be reshaped into (14×14, 14×14).



(a) Average attention map of spatial dimension. Its size is (14*14, 1+14*14).

(b) Average attention map of temporal dimension. Its size is (8, 1+8).

Figure 5: Marginalized average attention map of the video model. (a) spatial attention map (b) temporal attention map.

2.3. Observations

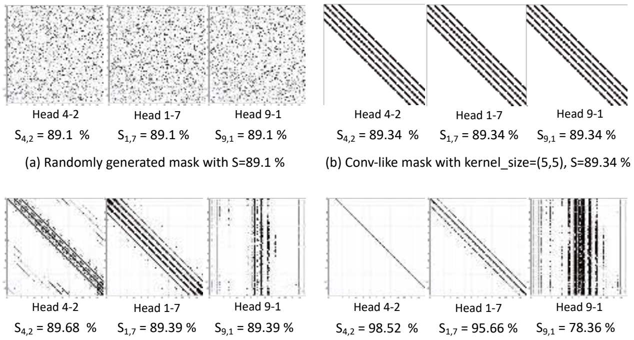
We have several interesting observations from attention map distribution of two models.

- Most of attention maps are sparse and only a few of them are dense. Heads in earlier layers tend to attend to local and sparse regions while attending globally as layers go deeper.
- We notice a few attention patterns: a) relative position-based attention, b) absolute position-based attention, and c) dense attention, which we hypothesize to be content-based attention.

Specifically, heads in earlier layers have either Convolution-like patterns or high activation score at certain absolute positions. On the other hand, heads in deeper layers mostly attend based on contents and its attending region is scattered across the image. This can be seen in Fig. 5 where the dominant color of the average map gradually changes from purple to yellow as layers go deeper. Purple color indicates that there are few regions which have high activation scores. Yellow color indicates that most regions have very similar average score so that after normalization they all become close to one.

3. Mask Mechanism on Attention Map

One core limitation of Transformer is the huge number of FLOPs it involves. Its full attention mechanism requires pairwise comparison among tokens, resulting in quadratic dependency on the number of tokens. When the number of tokens is large (e.g., for a high resolution image or a long video), the computation cost can be extraordinarily high.



(a) Randomly generated mask with S=89.1 %

(b) Conv-like mask with kernel_size=(5,5), S=89.34 %

(c) [Data-driven] Ratio-based mask with S=89.77 %

(d) [Data-driven] Information-based mask with I=0.3, S=90.48 %

Figure 6: Mask patterns generated using different mask generation methods. Elements filled with black (value 0) will be elements of interest whereas elements of white color (value 1) will be excluded. These are mask maps of an image model whose size is 197x197. $S_{l,h}$ denotes the sparsity ratio of head h in layer l.

In the previous section, however, we observed that many attention maps have high sparsity – especially in early layers. This means that pairwise comparison is not necessary in most cases and it can be viewed as a waste of computation resource. Based on this observation, we aim to explicitly put sparsity constraint into attention map by using masking mechanism. This involves two design challenges: 1) It's non-trivial to generate the optimal mask pattern, and 2) we need to finetune the sparsity ratio for the mask pattern to achieve a desired accuracy-efficiency trade-off.

In the following, we introduce six different mask pattern generation methods. Note that each mask is the same size to the attention map, *i.e.* (num_tokens, num_tokens), and `cls_token` relevant column and row, *i.e.* [0,:] and [:,0] is always activated. Fig. 6 presents mask patterns generated using different methods.

Random mask generation. Randomly generate a mask map with a certain sparsity ratio S. Here, all heads will have different mask maps.

Structured mask generation. Generate a 1/2/3D-Conv like mask map. Here, all heads share the same mask map.

In the following, we provide four mask generation methods which exploit data-driven average attention map. To generate mask, we first sort elements by activation score in a descending order and select subset along that order (sorting and selection happens query by query).

[Data-Driven] Ratio-based mask generation. Generate a mask with cut-off sparsity ratio S. For example, if S=1 and num_tokens=1,569, each query will select top 15 comparing elements based on their average similarity score. Here, each query will select the same number of elements.

[Data-Driven] Magnitude-based mask generation. The number of elements to select is determined by top-1 attention score. Let's say top-1 attention score is A. Then we select 1/A' elements, where A'=clip((A+τ), min=1/(1+THW), max=1) and τ is a temperature factor. For example, when $\tau = 0$, if one query's top-1 similarity score is 1, then it will select 1 comparing element and others unchosen. If another query's top-1 similarity score is 1/(1+THW), then it will select 1+THW comparing elements and nothing will be masked.

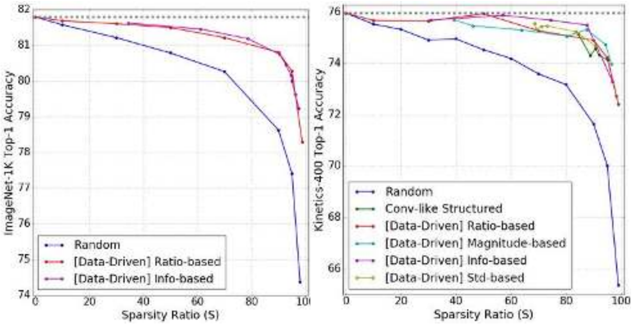
[Data-Driven] Information-based mask generation. Cutoff when the cumulative sum of the sorted similarity score is equal or greater than I where I is the quantity of information. Each query will select different number of elements.

[Data-Driven] Std-based mask generation. Compute mean (m) and std (σ) of each query's similarity vector. Then select comparing elements i, whose value is equal or larger than m+Cσ where C is sigma_coefficient.

Different from ratio-based mask generation, the last three methods will select different number of comparing elements for each query (Fig. 6 (c) and (d)). That is, sparsity degree would be taken into account in deciding the number of comparing elements to remain.

4. Experiments

We present performance of masked ViTs with different mask generation methods. For training a masked image model, we start from ImageNet-1K pre-trained weights and fine-tune weights with a mask. For a video model (including the non-masking model), we start from ImageNet-21K pre-trained weights and fine-tune for 30 epochs following [1].



(a) Performance of an **image** model with different mask generation methods

(b) Performance of a **video** model with different mask generation methods

Figure 7: Compare the performance of a masked ViT model with different mask generation methods. (a) Results on ImageNet-1K (b) Results on Kinetics-400.

We have three observations. First, manually-designed or data-driven masks outperform randomly generated masks. This indicates that self-attention actually happens with some patterns so random masking would lead to performance degradation. Second, there is little performance difference among different data-driven masks. Even though the gap is marginal, information-based mask seems to hold a slight edge over others. Third, ViT models with mask can maintain the performance loss less than 2 points even with sparsity ratio of 95%. This means that when num_tokens is 197, 9 tokens are used on average for attention mechanism. This shows that sparse attention mechanisms can preserve the expressivity of the original model to some extents.

5. Limitation

The current model with mask has one limitation – there is an upper bound of FLOPs reduction a model can achieve and it is not too large. Tab. 1a presents a submodule-wise FLOPs computation in a single Transformer. In terms of FLOPs, submodules that can benefit from mask mechanism are 'attention map computing' and 'weighted sum computing' parts. However, when instantiated with the actual model configurations (Tab. 1b), those parts take 4% and 25% of the total computation in DeiT-base and TimeSFormer ST, respectively. In other words, the maximum reduction in FLOPs a model can achieve is 4% and 25% in each model.

	MSA (Multi-head Self-Attention)				FFN (MLP)	Total
	QKV projection	Attention map computing	Weighted sum computing	Output projection	2-layer FC	
# FLOPs	3NCC	(1-S)·NNC	(1-S)·NNC	NCC	2NCM	4NCC + 2NNC + 2NCM

(a) Flop computation of a single Transformer module. N: num_tokens, C: dim_latent, M: dim_MLP, L: num_layers, S: sparsity ratio

	QKV projection	Attention map computing	Weighted sum computing	Output projection	2-layer FC	Total
Instantiation when N=197, C=768, M=3072, S=0 (DeiT-Base)						
# GFLOPs	0.3486	0.0298	0.0298	0.1162	0.9296	1.4540
Ratio	24.0%	2.0%	2.0%	8.0%	63.9%	100%
Instantiation when N=1569, C=768, M=3072, S=0 (TimeSFormer ST)						
# GFLOPs	2.78	1.89	1.89	0.93	7.40	14.89
Ratio	18.7%	12.7%	12.7%	6.2%	49.7%	100%

(b)

Table 1: (a) FLOP computation of a single Transformer module. (b) Actual GFLOPs instantiated with a specific model configuration.

This is mainly because the number of tokens in both models is much smaller than the channel dimension in FFN, where num_tokens is 197 or 1569 and dim_MLP is 3072. Therefore, in ViT models where num_tokens $<$ dim_MLP, an effective way to control the total FLOPs is to adjust FFN parts such as dim_MLP. On the flip side, this also means that masking mechanism can be more effective when num_tokens gets larger.

References

[1] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, 2020.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 2009.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[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. In *International Conference on Learning Representations*, 2021.

[6] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[7] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

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

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