Explicitly Modeled Attention Maps for Image Classification

Andong Tan* [1,4], **Duc Tam Nguyen*** [2,4], **Maximilian Dax**[3,4], **Matthias Nießner**[1], **Thomas Brox**[2]

[1] Technical University of Munich
[2] University of Freiburg
[3] University of Bonn
[4] Robert Bosch GmbH

andong.tan@tum.de, nguyen@cs.uni-freiburg.de, maximiliandax@gmx.com, niessner@tum.de, brox@cs.uni-freiburg.de

Abstract

Self-attention networks have shown remarkable progress in computer vision tasks such as image classification. The main benefit of the self-attention mechanism is the ability to capture long-range feature interactions in attention-maps. However, the computation of attention-maps requires a learnable key, query, and positional encoding, whose usage is often not intuitive and computationally expensive. To mitigate this problem, we propose a novel self-attention module with explicitly modeled attention-maps using only a single learnable parameter for low computational overhead. The design of explicitly modeled attention-maps using geometric prior is based on the observation that the spatial context for a given pixel within an image is mostly dominated by its neighbors, while more distant pixels have a minor contribution. Concretely, the attention-maps are parametrized via simple functions (e.g., Gaussian kernel) with a learnable radius, which is modeled independently of the input content. Our evaluation shows that our method achieves an accuracy improvement of up to 2.2% over the ResNet-baselines in ImageNet ILSVRC and outperforms other self-attention methods such as AA-ResNet152 in accuracy by 0.9% with 6.4% fewer parameters and 6.7% fewer GFLOPs. This result empirically indicates the value of incorporating geometric prior into self-attention mechanism when applied in image classification.

Introduction

An attention mechanism allows the network to focus on the global context in each layer. Attention is an essential part of human vision, also known as foveation, which allows the vision system to focus its limited resources on a small part of the input signal. The implementation of this concept as self-attention in the transformer network (Vaswani et al. 2017) has resulted in a substantial performance increase in natural language processing (Devlin et al. 2018; Radford et al. 2018). Recent works in computer vision (Bello et al. 2019; Parmar et al. 2019) proposed self-attention for object recognition tasks. Their suggested modification of common network architectures, such as the Resnet (He et al. 2016a,b), leads to significant performance improvement over the original convolutional baseline.

One key benefit of a self-attention layer is that such a layer can incorporate the entire spatial context for the computation of features in the subsequent layers through the calculation of attention-maps. As an intuitive explanation, attention-maps express how much attention different areas of the input receive when focusing on a particular pixel position. However, the high degree of freedom of self-attention networks, such as in (Bello et al. 2019), requires learning the weights for the key and the query to calculate attention-maps. There is no obvious need for this excessive, computationally expensive parametrization in the context of computer vision tasks. By visualizing the weights, we show that the content-dependent key and query play a minor role in the final attention-maps, where an area with high attention weight is more related to the geometric position; compare Fig. 1c, 1g to Fig. 1b, 1f. Further, in vision tasks such as image classification, a common observation is that neighbor pixels are more related than distant ones. In other words, when focusing on particular pixels, neighbor areas should receive higher attention.

Based on these insights, we propose a self-attention mechanism with explicitly modeled attention-maps, which considers the global context information with positive correlation to the distance between pixels. The freedom of the attention module is reduced to a predefined form (e.g., Gaussian kernel) with a learnable parameter, as illustrated in Fig. 2. We integrate the local context prior in self-attention explicitly by restricting the learnable attention-map to a centered, yet global kernel with a learnable shape parameter. The primary motivation is to maintain global information for the feature computation while reducing the freedom caused by learning key and query for higher efficiency. Surprisingly, networks augmented with such modules not only outperform regular self-attention in parameters, memory and computation cost, but also achieve very competitive accuracy.

The contributions of this paper are summarized as follows: (1) We analyze the efficiency of AA-Net (Bello et al. 2019) and empirically show that geometric information plays an essential role in attention-maps. (2) Based on the above analysis, we propose a novel self-attention module with explicitly modeled attention-maps under the assumption that neighbor pixels are more related than distant ones. We investigate fixed attention-maps parametrized by different global, yet centered simple functions (e,g, cosine, linear) to model monotonically decreased attention paid to distant pixels w.r.t. centered pixel.

*Equal contribution

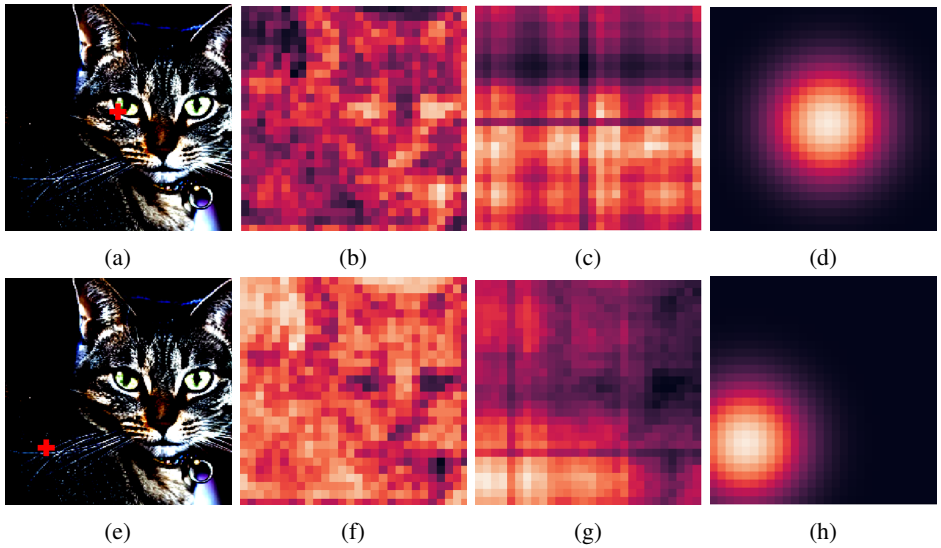


Figure 1: Attention-maps of two pixels: (a,e) are the input from ImageNet, red crosses indicate the positions of 2 pixels. (b,f) show key×query; (c,g) show key×query+positional encoding; (d,h) show explicitly modeled attention-maps using Gaussian kernel. All pictures are extracted from the first layer where attention mechanism is applied in ResNet50. Brighter color indicates higher weight. First and second row of pictures correspond to attention-maps of the center and left pixels. All values are normalized for visualization.

Further, we study the effect of automatically determining attention-maps using the Gaussian kernel with a learnable radius. (3) Experimental results in CIFAR10, CIFAR100, Tiny ImageNet, and ImageNet show that convolutional networks augmented with such modules have lower model complexity than augmented with regular self-attention while achieving very competitive accuracy.

Related Works

The most important works on attention in deep networks comprise advances for sequence-to-sequence modeling in natural language processing (NLP) tasks such as neural machine translation (Bahdanau, Cho, and Bengio 2014). More recently, multi-head self-attention (Vaswani et al. 2017; So, Liang, and Le 2019) allows effective pretraining for many NLP-tasks using language modeling as a self-supervision task ((Devlin et al. 2018; Radford et al. 2018)) and for other tasks (Shaw, Uszkoreit, and Vaswani 2018; Zhang et al. 2018; Yu et al. 2018; Zhang, Winn, and Tomioka 2016). Since self-attention is computationally expensive, there are also works exploring efficient self-attention (Shen et al. 2018, 2019; Kitaev, Kaiser, and Levskaya 2019). Further, Synthesizer (Tay et al. 2020) challenges the necessity of computationally expensive key-query self-attention, and fixed self-attention patterns are proposed for machine translation (Raganato, Scherrer, and Tiedemann 2020). However, these proposed self-attention mechanisms are designed for sequences-to-sequence tasks and do not necessarily transfer to imaging tasks with different dynamics. Contrary, the focus of our work is to study the self-attention concept from NLP for learning on computer vision tasks.

Attention methods in image recognition tasks can be roughly categorized into channel attention, spatial attention, or a combination of them. A representative work exploring channel attention is SE-Net (Hu, Shen, and Sun 2018), which calculates channel attention by using global average pooling and channel scaling. GE-Net (Hu et al. 2018) uses depth-wise convolution to calculate spatial attention. CBAM (Woo et al. 2018) extends SE-Net by additionally considering spatial attention independently. ResNeSt (Zhang et al. 2020) uses a cardinal group to generalize prior work in channel attention. Further, GSoP (Gao et al. 2019) exploits channel and spatial attention respectively from a statistical perspective. More recent work such as AA-Net (Bello et al. 2019) calculates spatial and channel attention jointly using the self-attention concept from NLP. These works mainly aim at improving performance with intricate module design. In the domain of efficient attention mechanism, ECA-Net (Wang et al. 2019) improves SE-Net (Hu, Shen, and Sun 2018) for an efficient channel attention mechanism by controlling the size of 1D convolution. Different from all the above, we try to improve AA-Net (Bello et al. 2019) and aim at offering efficient attention in spatial and channel dimensions jointly by incorporating geometric prior.

Overall, to the best of our knowledge, we are the first to propose a self-attention module with explicitly modeled attention-maps in an extremely simplified way for vision task. In this form, the attention-maps are shared across multiple heads and are parameterized by only one single learnable parameter in each layer. Compared to AA-Net (Bello et al. 2019), our module does not employ any key or query and retains only the value. Since our method is strongly motivated by AA-Net (Bello et al. 2019), we first introduce how AA-Net (Bello et al. 2019) applies the self-attention concept from

NLP in vision task.

Background on Multi-head Self-attention in Computer Vision

We first denote the notations used in this section. Following the convention, we denote H, W, C as height, width, and the number of channels of input $X' \in R^{H\times W\times C}$ (Batch dimension is omitted for simplicity). The flattened input is denoted as $X \in R^{HW\times C}$. Further, we define d as the depth of key or query, d_v^h as the depth of value in each head, and N as the total number of heads.

The multi-head self-attention is calculated as in the Transformer architecture (Vaswani et al. 2017). The three steps of the process are defined as (Bello et al. 2019):

$$Att(X) = Softmax\left(\frac{KQ^T + PosEncoding}{\sqrt{d}}\right)V \quad (1)$$

$$V = XW_v^h; K = XW_k^h; Q = XW_q^h \quad (2)$$

$$Multihead(X) = Concat[Att_1, ..., Att_N]W^o \quad (3)$$

where $W_v^h \in R^{C\times d_v^h}$, $W_k^h \in R^{C\times d}$, $W_q^h \in R^{C\times d}$, and $W^o \in R^{Nd_v^h\times Nd_v^h}$ are 4 learnable matrices to calculate value V, key K, query Q and final output respectively. The positional encoding term refers to a learnable relative positional encoding (**?**), which is translational invariant. The division of $\sqrt{d}$ is designed for better training. The calculations in Eq. 2 are repeated multiple times with different learnable matrices to get multiple $Att(X)$ (also named as one head). In the last step, the results of N heads are concatenated along the depth dimension and linearly projected to achieve final multi-head self-attention. An overview of this method is offered in the left part of Fig. 2

What are attention-maps? The attention-map describes how much attention every pixel in the input is paid to when the model is focusing on one specific pixel. Every pixel has one attention-map, so there are in total HW attention-maps for the input of height H and width W, and every attention-map has spatial shape $H \times W$. Therefore the softmax part in Eq. 1 with the shape $HW \times HW$ indicates the attention paid to all $H \times W$ pixels when focusing on every pixel, respectively. Figure 1 shows a visualization of attention-maps of 2 pixels.

Content-dependent attention-maps. The attention-map of the above mechanism is constructed by key K, query Q, and a relative positional encoding term. Both key and query are linear projections of input, while the relative positional encoding term also depends on the query (Bello et al. 2019). Therefore, attention-maps are content-dependent. The multiplication between $K \in R^{HW\times d}$ and $Q^T \in R^{d\times HW}$ in Eq. 1 calculates the similarity between extracted features K and Q. Its output KQ^T with shape $HW \times HW$ indicates how similar each pixel's extracted feature is related to every other pixel. Finally, after adding a positional encoding and scaling, the final attention-maps are built.

However, as shown above, constructing attention-maps in such a way requires many learnable parameters and multiplication operations, and why the above mechanism is beneficial in the computer vision context is not obvious.

Explicitly Modeled Attention-maps

Intuitively, from the visualization of Figure 1, it could be observed that the key $\times$ query highly depends on the input (Figure 1b, 1f), while the weights consisting of key, query and positional encoding (Figure 1c, 1g) does not depend much on the input content. The fact that the latter form is proved to increase the performance in image classification (Bello et al. 2019) indicates the importance of geometric information. This observation inspires us to design input independent attention-maps for vision tasks. Therefore, in comparison to previously described multi-head self-attention in the last section, we made the following modification: replace content-dependent attention-maps with content-independent explicitly modeled attention-maps using the assumption that neighbor pixels are more related than distant ones.

General Form

Generally, the spatial context for a given pixel within an image is mostly dominated by its neighbors, while more distant pixels have a minor contribution. Motivated by this observation and noting that attention-maps indicate the importance of all input pixels when focusing on each pixel, we explicitly design the weight distribution in attention-maps. In each attention-map of one specific pixel i, the weight assigned to any pixel j decreases monotonically as the spatial distance between two pixels (i and j) in input increases. Our proposed design for attention-maps is translational invariant and incorporates relative positional information by assigning spatial distance-dependent weights in a less costly manner than regular self-attention as introduced in previous section.

We define three steps of our attention mechanism as follows:

$$V = XW_v \quad (4)$$

$$P = Norm(G + 1)V \quad (5)$$

$$ExpAtt(X) = PW^o \quad (6)$$

In addition to existing notations, we define d_v as the total number of channels of all attention heads, where $d_v = N \times d_v^h$. $W_v \in R^{C\times d_v}$ is a learned linear transformation, which can be easily realized by d_v $1D$ convolutions to create value V in N heads together. We introduce the matrix $G \in R^{HW\times HW}$ in Eq. 5 to model weight distribution in attention-maps explicitly and constrain $G_{ij} \in (0, 1]$. The concrete forms of G are discussed in the subsequent section. We add an element-wise offset of 1 to G to impose global feature consideration which ensures that information from the relatively distant area also has a reasonable magnitude of weight. This makes the whole spatial input considered. Following the normalization design of $Softmax$ in Eq. 1, $Norm$ denotes a row-wise normalization on input, and $Norm(G + 1)$ indicates the attention-maps. $Norm$ normalizes values by directly dividing sum of values in each row of its input instead

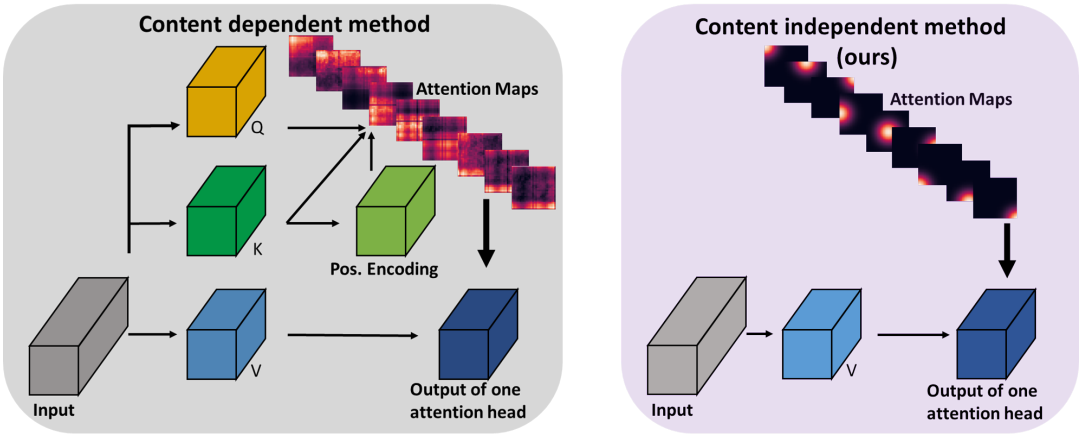


Figure 2: Comparison between self-attention mechanism using content-dependent and independent attention-maps: content-dependent attention-maps are constructed using linearly projected input key (K) and query (Q), while our explicitly modeled attention-maps simply incorporate geometric prior. In the end, the attention-maps are used to calculate the weighted average of the value (V) for the output of one attention head. Attention-maps are restructured for better visualization.

.

of additionally using an exponential function as in $Softmax$. This retains the effect of the designed G.

Additionally, we share attention-maps across heads. This helps to avoid splitting value V into multiple heads and concatenate them together in the end as Eq. 3. Finally, in Eq. 6, the output is linearly projected to achieve final self-attention output, where the weights are expressed in $W^o \in R^{d_v \times d_v}$.

Fully Fixed Attention-maps

A natural choice of modeling attention-maps explicitly is directly fixing all weights in attention-maps using exact functions. We design different simple functions for G to model the weight decrease in attention-maps with increasing relative spatial distance. As a special case, we include the option of using a constant function. G_{ij} indicates how much attention is paid to pixel j when focusing on pixel i. We denote i_x, i_y, j_x, j_y as the corresponding spatial coordinate of pixel i and j in the input X'. For a fair comparison, all alternative formulations of G are designed such that when $i = j$, $G_{ij} = G_{max} = 1$ (one pixel should receive the highest attention when focusing on itself).

Constant In this case, the weights are distributed uniformly. Using different constants does not make a difference since the result after normalization will be the same.

$$G_{ij} = 1 \tag{7}$$

Linear decrease For linear function, the weight decreases with an increasing Euclidean distance at a constant speed.

$$G_{ij} = 1 - \frac{\sqrt{(i_x - j_x)^2 + (i_y - j_y)^2}}{\sqrt{H^2 + W^2}} \tag{8}$$

Cosine decrease In cosine decrease, weights are decreasing slower in the neighborhood and quicker in the middle distance and decreasing slower again in the very distant area.

$$G_{ij} = 0.5 \cdot \left(1 + \cos\left(\frac{\sqrt{(i_x - j_x)^2 + (i_y - j_y)^2}}{\sqrt{H^2 + W^2}}\pi\right)\right) \tag{9}$$

The main advantage of fully fixed attention-maps is they can be fully pre-calculated before training and simply loaded into the model without introducing any learnable parameter or online computation. The implementation is very simple.

Learnable Attention-maps

Since our mechanism models weight distribution in attention-maps explicitly, how exactly the weights are distributed needs to be optimized. The optimized form of attention-maps can be tuned manually with different kinds of functions. Some examples include constant, linear, and cosine functions. However, manual tuning will cost lots of computing resources. Therefore, we further show some possible options to automatically determine the weight distribution in attention-maps.

Gaussian First of all, G is parametrized via the Gaussian kernel function defined as:

$$G_{ij} = \exp\left(\frac{(\frac{i_x - j_x}{W})^2 + (\frac{i_y - j_y}{H})^2}{-2\sigma^2}\right) \tag{10}$$

σ is a learnable scalar, which indicates how flat the weights are distributed in attention-maps. Since the Gaussian kernel is also a radial basis function, we name σ as radius, which is the shape parameter of this function. Visually, it also indicates the "radius" of the "circle-shaped" attention-maps, as shown in Figure 1d.

Exponential decrease We form the exponential decrease in two different ways, where the weight decreases with Euclidian and Manhatten distances, respectively as follows:

$$G_{ij} = \exp\left(\frac{\sqrt{(\frac{i_x - j_x}{W})^2 + (\frac{i_y - j_y}{H})^2}}{-\sigma}\right) \quad (11)$$

$$G_{ij} = \exp\left(\frac{(\frac{|i_x - j_x|}{W}) + (\frac{|i_y - j_y|}{H})}{-\sigma}\right) \quad (12)$$

Both options are similar to the Gaussian kernel function and mainly differ in the form of the numerator. In the choice of Gaussian kernel function, the weight also decreases exponentially. However, we keep them separate for convenience in later discussion.

Analysis on Efficiency

Since our method mainly improves the attention-maps construction, we focus the theoretical analysis on this part. Given input with height H, width W, and C channels, the calculation of key and query in regular self-attention projects the input from C channels to d channels, respectively. Therefore they introduce $N \times 2Cd$ learnable parameters for N heads in one layer. In each head, different attention-maps with shape $HW \times HW$ are saved, which has $O(N(HW)^2)$ memory cost. The projection of key and query costs $O(2HWC^2d)$ computation, and the multiplication between key and query costs $O(d^2(HW)^2)$ per head. The same computation is repeated in N heads. Relative positional encoding introduces $2(H+W-1)d$ parameters with memory cost $O(HWd)$ and computation cost $O((HW)^2)$ per layer according to Bello et al. (2019).

However, using our explicitly modeled attention-maps will introduce only one learnable parameter (radius) if they are learnable and will not introduce any parameter if they are fixed. Further, we reduce the memory cost from $O(N(HW)^2)$ to $O((HW)^2)$ thanks to the sharing of attention-maps across heads. From the perspective of computation, fully fixed attention-maps can be completely precalculated and require no computation during training. Even our learnable variant only needs some scaling operations since the numerator of G can be pre-calculated. Both variants of our method drastically increase the efficiency. The comparison is summarized in Table 1.

Experiments

In this section, we test our *ExpAtt* module in widely used architectures such as ResNets (He et al. 2016b,a) and representative lightweight architecture such as MobileNetV2 (Sandler et al. 2018) on small scale and large scale image classification datasets including CIFAR10, CIFAR100 (Krizhevsky 2009), Tiny ImageNet (Yao and Miller 2015) and ImageNet (Deng et al. 2009). We report average accuracy for all experiments. The experiments show that our module leads to improvement in different architectures in multiple aspects. Since self-attention networks strongly motivate the proposed *ExpAtt* module, we follow exactly the same network settings of Bello et al. (2019) to integrate our *ExpAtt* module into networks for comparability. Experiments in the same dataset use same data preprocessing.

Integration by Feature Concatenation

For an original convolution with stride 1 and output channels C_{out} , we first split C_{out} to standard convolution features C_{conv} and *ExpAtt* features C_{expatt}. In other words, $C_{conv} + C_{expatt} = C_{out}$. Subsequently, the convolution output has shape $H \times W \times C_{conv}$ with H and W being the input height and width, respectively. The *ExpAtt* output has shape $H \times W \times C_{expatt}$. From the perspective of multi-head self-attention, C_{expatt} is equivalent to $N \times d_v^h$, where N is the number of attention heads, and d_v^h is the depth of value in each head. Finally, the $ExpAtt$ output is concatenated with convolution output along channel dimension to receive the augmented convolutional features in shape $H \times W \times C_{out}$. For convolutions with stride 2, an additional 3×3 average pooling with stride 2 is applied to the *ExpAtt* output to keep the spatial shape matching. The number of heads is fixed to 8 in ResNets and 4 in MobileNetV2. The ratio of C_{expatt}/C_{out} is set to 0.1 for ResNets and 0.05 for MobileNetV2. When C_{expatt} is not evenly dividable by 8 or 4, the closest value that is evenly dividable is taken. Self-attention mechanism (including AA-Net and our ExpAtt-Net) is incorporated into 3×3 convolutions of all 4 residual stages of ResNets in CIFAR and only last 3 stages of ResNets in ImageNet experiments. The integration into MobileNetV2 starts when channel number is 24×6 through concatenation with 1×1 convolutions. More details are offered in the Appendix.

Training

Models are trained from scratch. All experiments (including AA-Net and ExpAtt-Net) are based on the respective baselines from PyTorch (Paszke et al. 2019), use synchronous SGD with momentum 0.9, and cosine learning rate with restarts (Loshchilov and Hutter 2016) for in total 450 epochs, 164 epochs, and 324 epochs in CIFAR, ImageNet, and Tiny ImageNet experiments respectively. Concretely, in the first 15 epochs, learning rate is linearly increased to 0.05, than a cosine learning rate with restarts at 25,45,85,165,325 epochs is applied where appliable. Additionally, CIFAR experiments use learning rate 0.0002 between epoch 325 and 450. Batch size of all experiments are chosen to fit the GPU memory. The radius σ of Gaussian kernel is initialized to 0.75.

ResNet50 in CIFAR-10,100

Tab. 2 shows the performance of Resnet50 when the attention-maps of our *ExpAtt* module are parametrized differently with various simple functions. To fit the resolution of CIFAR, we remove the first average pooling and change the stride of the first convolution to one in ResNet50. All considered attention modules with different parametric attention-maps outperform the ResNet50 baseline and the plain self-attention in AA-ResNet50. All functions perform similarly with the Gaussian-kernel being slightly better than others. This may occur because different functions have similar gaussian-like patterns. Surprisingly, even using uniform distribution in attention-maps achieves competitive performance compared to AA-ResNet50. This may mean regularization is helpful in the attention module.

	PARAMETERS	MEMORY	COMPUTATION
KEY AND QUERY	$N \times 2Cd$	$O(N(HW)^2)$	$O(2NHWC^2d + Nd^2(HW)^2)$
POSITIONAL ENCODING	$2(H+W-1)d$	$O(HWd)$	$O((HW)^2)$
LEARNABLE ATT.-MAPS	1	$O((HW)^2)$	$O((HW)^2)$
FULLY FIXED ATT.-MAPS	0	$O((HW)^2)$	0

Table 1: Comparison on parameter, memory and computation cost between explicitly modeled attention-maps and attention-maps calculated by key, query and relative positional encoding per layer.

DECAY FUNC.	PARA.	FLOPS	ACC.
CIFAR-10			
RESNET50	23.7M	1.31G	90.20
AA-RESNET50	23.9M	1.45G	90.78
UNIFORM	22.7M	1.25G	90.77
COSINE	22.7M	1.25G	90.96
LINEAR	22.7M	1.25G	90.91
EXP. EUCLID.	22.7M	1.25G	90.94
EXP. MAN.	22.7M	1.25G	**91.02**
GAUSSIAN	22.7M	1.25G	**90.99**
CIFAR-100			
RESNET50	23.7M	1.31G	79.46
AA-RESNET50	23.9M	1.45G	80.32
COSINE	22.7M	1.25G	80.74
UNIFORM	22.7M	1.25G	80.76
LINEAR	22.7M	1.25G	80.82
EXP. EUCLID.	22.7M	1.25G	80.90
EXP. MAN.	22.7M	1.25G	80.91
GAUSSIAN	22.7M	1.25G	**81.02**

Table 2: Performance of modified ResNet50 using different parametric attention-maps on CIFAR-10/100.

TYPE	PARA.	FLOPS	TOP1	TOP5
RESNET34	21.8M	3.6G	73.30	91.42
SE-RESNET34	22.0M	3.6G	74.30	91.80
CBAM-RESNET34	22.0M	3.7G	74.01	91.76
AA-RESNET34	20.7M	3.6G	**74.70**	**92.00**
Gaussian-ExpAtt	17.3M	3.1G	74.24	91.81
RESNET50	25.6M	3.8G	76.15	92.87
SE-RESNET50	28.1M	3.9G	77.50	93.70
CBAM-RESNET50	28.1M	3.9G	77.34	93.69
ECA-RESNET50	24.4M	3.9G	77.48	93.68
AA-RESNET50	25.8M	4.2G	77.70	93.80
Gaussian-ExpAtt	24.5M	4.0G	**78.13**	**94.07**
RESNET101	44.5M	7.6G	77.37	93.56
SE-RESNET101	49.3M	7.6G	78.40	94.20
CBAM-RESNET101	49.3M	7.6G	78.49	94.31
ECA-RESNET101	42.5M	7.4G	78.65	94.34
AA-RESNET101	45.4M	8.1G	78.70	94.40
Gaussian-ExpAtt	42.7M	7.6G	**79.56**	**94.78**
RESNET152	60.2M	11.3G	78.31	94.06
SE-RESNET152	66.8M	11.3G	78.90	94.50
ECA-RESNET152	57.4M	10.8G	78.92	94.55
AA-RESNET152	61.6M	11.9G	79.10	94.60
Gaussian-ExpAtt	57.6M	11.1G	**80.02**	**94.85**

Table 3: Performance of ResNets utilizing different attention modules in ImageNet. Our methods are cursive.

ResNets in ImageNet

In table 3, we compare representative networks exploring channel attention (SE-Net(Hu, Shen, and Sun 2018)), efficient channel attention (ECA-Net(Wang et al. 2019)), channel and spatial attention independently (CBAM (Woo et al. 2018)) and jointly (AA-Net (Bello et al. 2019)) with our *ExpAtt-Net*, which explores efficient joint channel and spatial attention. The methods are compared by integration into multiple Resnet architectures. The Gaussian kernel parametrized *ExpAtt* improves its counterpart using key and query (AA-ResNet) by 0.47%, 0.87%, and 0.92% on ResNet50, ResNet101, and ResNet152, respectively in Top1 accuracy, which indicates that increasing architecture depth is beneficial for explicit modeling of the attention-maps. This may due to the regulation introduced by *ExpAtt* model, which helps to decrease training difficulty when the model becomes deeper. Compared to ECA-ResNet, which is designed to execute efficient channel attention, we use a similar number of parameters and GFLOPs while achieving a higher Top1 accuracy. This further proves the benefit of our method.

However, the *ExpAtt* augmented ResNet34 underperforms the AA-Resnet34, though still outperforming ResNet34 baseline. Since ResNet50/101/152 uses a different type of residual block (bottleneck) than ResNet34, this may indicate that our method is more suitable to model attention weight distribution in architectures with bottleneck residual block which consists of a 1×1, 3×3, and 1×1 convolution instead of the residual blocks consisting of two 3×3 convolutions (e.g., ResNet34).

MobileNetV2 in Tiny ImageNet

In this section, we use a parameter efficient architecture MobileNetV2 as the backbone and compare *ExpAtt* with AA-Net (Bello et al. 2019) using precisely the same training and network setting. Following Bello et al. (2019), we apply the Gaussian parametrized *ExpAtt* module in an inverted bottleneck by replacing part of expansion point-wise convolution channels. Table 4 shows that we achieve an accuracy improvement with lower model complexity compared to AA-MobileNetV2. Since the inverted bottleneck mainly consists of two point-wise and one depth-wise convolution, this also suggests a way to let our method complement depth-wise convolution.

Ablation Study

Sharing attention-maps across heads From the perspective of multi-head self-attention, each head can have a different set of attention-maps. Therefore, we study the effect of sharing and not sharing attention-maps (parametrized by

ARCHITECTURE	PARA.	FLOPS	ACC.
MOBILENETV2	3.50M	0.32G	64.72
AA-MOBILENETV2	3.55M	0.33G	65.89
EXPATT-MOBILENETV2	3.51M	0.32G	**66.14**

Table 4: Performance of MobileNetV2 and its variants augmented with self-attention modules.

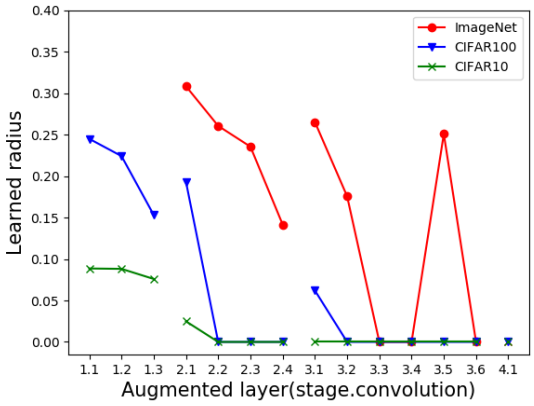
Gaussian kernel) across heads by experiments in ResNet50 on CIFAR100. The result shows that sharing attention-maps achieves 81.02% Top1 accuracy while using different attention-maps across heads achieves only 80.78%. One possible explanation is that the tied radius parameter might reduce the difficulty of joint optimization in training neural networks.

Interplay with content-based method In this study, we try to understand whether our content-independent method is orthogonal to the content-dependent one using ResNet50 on CIFAR100. In all augmented layers, we combine the two methods by concatenating the output of *ExpAtt* and the content-based method's output (Bello et al. 2019). Unfortunately, the accuracy (78.19%) is worse than plain ResNet50 (79.46%). The result suggests that both methods are not complementary, though they individually have an obvious improvement over vanilla networks.

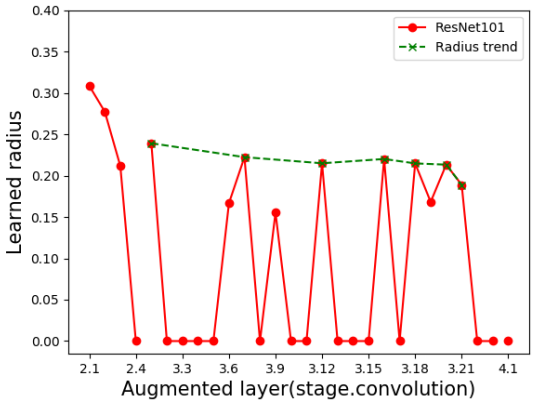
Importance of global features Though our method is motivated by focusing local features, we explicitly constrain the attention-maps to consider global features by using element-wise plus one to G. Without using element-wise plus one in Eq. 5, attention-maps parametrized by Gaussian kernel achieves only 80.42% in CIFAR100 while additionally using element-wise plus one achieves 81.02%. This indicates the importance of considering global features.

Results of Learned Radius

Since radius is the only learnable parameter of Gaussian kernel parametrized *ExpAtt*, we show how it varies between layers. The augmented layers are denoted as "stage.convolution" on the x-axis, because ResNet50 consists of several stages, and each stage has several 3×3 convolution layers. Fig. 3a shows the final learned value of the radius in different layers of ResNet50 on different datasets. There is an apparent decreasing trend in the learned radius as the layer depth increases in the first augmented stage. This suggests that the global context is more important in early layers than later ones. Further, the radius learned in stage one on CIFAR100 is higher than the learned radius on CIFAR10. This may mean that the global context is more important as the prediction task becomes more difficult. In the third stage, although the radius is not continuously decreasing, the general trend of "peak radius" in neighbors is still decreasing as the layer goes deeper. This trend can be easier observed in deeper network such as ResNet101 (Fig. 3b) . In the fourth stage, only the first augmented convolution is parametrized by the Gaussian kernel, and the final radius is close to zero in all datasets. Radius



(a)



(b)

Figure 3: The final results of learned radius in different augmented layers of Resnet50 on CIFAR and ImageNet (a) and ResNet101 on ImageNet (b). Results in different stages are split for clarity.

close to zero means that attention-map of any pixel F assigns the highest weight to pixel F and considers information from all other surrounding pixels equally.

Conclusion

We aim at offering an efficient self-attention mechanism for vision task. To this end, we propose an *ExpAtt* module to explicitly model weight distribution in attention-maps by incorporating a geometric prior. Despite the simplicity of this module compared to self-attention, experimental results show that it improves the performance of multiple architectures, including widely used ResNets and lightweight MobileNetV2. Surprisingly, it outperforms the regular self-attention design not only in efficiency but also in accuracy when integrated into the bottleneck residual block. Although experiments focus on image classification, we expect *ExpAtt* to be applicable to other vision tasks, because the assumption that nearby pixels are more related than distant ones is a general principle in images and that the Resnet baseline for image classification is widely used as backbone for many other tasks.

Acknowledgements

The authors would like to thank Zhongyu Lou from Robert Bosch GmbH for insightful comments and discussions.

Ethical Impact

Our content-independent attention-maps can help to decrease bias introduced by datasets against minorities. For example, in a face image recognition task, more training images of majorities may cause the system to perform worse in people of minorities. Self-attention with content-dependent attention-maps might learn such bias due to its large number of parameters learned from datasets. In contrast, both variants of our method try to avoid learning such bias. Concretely, the fully fixed attention-maps will not be influenced by any bias of datasets because it learns nothing from datasets. Our learnable attention-maps would also be much less influenced by such bias compared to content-based methods, since it only learns a single shape parameter thanks to our very general assumption that neighbor pixels are more related than distant ones. However, we still note that the system might be misused to cause negative ethical impact.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Bello, I.; Zoph, B.; Vaswani, A.; Shlens, J.; and Le, Q. V. 2019. Attention augmented convolutional networks. *arXiv preprint arXiv:1904.09925* .

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Gao, Z.; Xie, J.; Wang, Q.; and Li, P. 2019. Global second-order pooling convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3024–3033.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645. Springer.

Hu, J.; Shen, L.; Albanie, S.; Sun, G.; and Vedaldi, A. 2018. Gather-excite: Exploiting feature context in convolutional neural networks. In *Advances in Neural Information Processing Systems*, 9401–9411.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132–7141.

Kitaev, N.; Kaiser, L.; and Levskaya, A. 2019. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto.

Loshchilov, I.; and Hutter, F. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* .

Parmar, N.; Ramachandran, P.; Vaswani, A.; Bello, I.; Levskaya, A.; and Shlens, J. 2019. Stand-Alone Self-Attention in Vision Models. In *Advances in Neural Information Processing Systems*, 68–80.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Raganato, A.; Scherrer, Y.; and Tiedemann, J. 2020. Fixed Encoder Self-Attention Patterns in Transformer-Based Machine Translation. *ArXiv* abs/2002.10260.

Sandler, M.; Howard, A. G.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* 4510–4520.

Shaw, P.; Uszkoreit, J.; and Vaswani, A. 2018. Self-Attention with Relative Position Representations. In *NAACL-HLT*.

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; and Zhang, C. 2018. Bi-Directional Block Self-Attention for Fast and Memory-Efficient Sequence Modeling. *ArXiv* abs/1804.00857.

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; and Zhang, C. 2019. Tensorized Self-Attention: Efficiently Modeling Pairwise and Global Dependencies Together. In *NAACL-HLT*.

So, D. R.; Liang, C.; and Le, Q. V. 2019. The evolved transformer. *arXiv preprint arXiv:1901.11117* .

Tay, Y.; Bahri, D.; Metzler, D.; Juan, D.; Zhao, Z.; and Zheng, C. 2020. Synthesizer: Rethinking Self-Attention in Transformer Models. *ArXiv* abs/2005.00743.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, Q.; Wu, B.; Zhu, P.; Li, P.; Zuo, W.; and Hu, Q. 2019. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks. *ArXiv* abs/1910.03151.

Woo, S.; Park, J.; Lee, J.-Y.; and So Kweon, I. 2018. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 3–19.

Yao, L.; and Miller, J. 2015. Tiny imagenet classification with convolutional neural networks. *CS 231N* 2(5): 8.

Yu, A. W.; Dohan, D.; Luong, M.-T.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541* .

Zhang, H.; Goodfellow, I.; Metaxas, D.; and Odena, A. 2018. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318* .

Zhang, H.; Wu, C.; Zhang, Z.; Zhu, Y.; Zhang, Z.-L.; Lin, H.; e Sun, Y.; He, T.; Mueller, J.; Manmatha, R.; Li, M.; and Smola, A. 2020. ResNeSt: Split-Attention Networks. *ArXiv* abs/2004.08955.

Zhang, L.; Winn, J.; and Tomioka, R. 2016. Gaussian attention model and its application to knowledge base embedding and question answering. *arXiv preprint arXiv:1611.02266* .