Attention Augmented Convolutional Networks

Irwan Bello Barret Zoph Ashish Vaswani Jonathon Shlens
Quoc V. Le
Google Brain
{ibello,barretzoph,avaswani,shlens,qvl}@google.com

Abstract

Convolutional networks have been the paradigm of choice in many computer vision applications. The convolution operation however has a significant weakness in that it only operates on a local neighborhood, thus missing global information. Self-attention, on the other hand, has emerged as a recent advance to capture long range interactions, but has mostly been applied to sequence modeling and generative modeling tasks. In this paper, we consider the use of self-attention for discriminative visual tasks as an alternative to convolutions. We introduce a novel two-dimensional relative self-attention mechanism that proves competitive in replacing convolutions as a stand-alone computational primitive for image classification. We find in control experiments that the best results are obtained when combining both convolutions and self-attention. We therefore propose to augment convolutional operators with this self-attention mechanism by concatenating convolutional feature maps with a set of feature maps produced via self-attention. Extensive experiments show that Attention Augmentation leads to consistent improvements in image classification on ImageNet and object detection on COCO across many different models and scales, including ResNets and a state-of-the art mobile constrained network, while keeping the number of parameters similar. In particular, our method achieves a 1.3% *top-1 accuracy improvement on ImageNet classification over a ResNet50 baseline and outperforms other attention mechanisms for images such as Squeeze-and-Excitation [17]. It also achieves an improvement of 1.4 mAP in COCO Object Detection on top of a RetinaNet baseline.*

1. Introduction

Convolutional Neural Networks have enjoyed tremendous success in many computer vision applications, especially in image classification [24, 23]. The design of the convolutional layer imposes 1) locality via a limited receptive field and 2) translation equivariance via weight sharing.

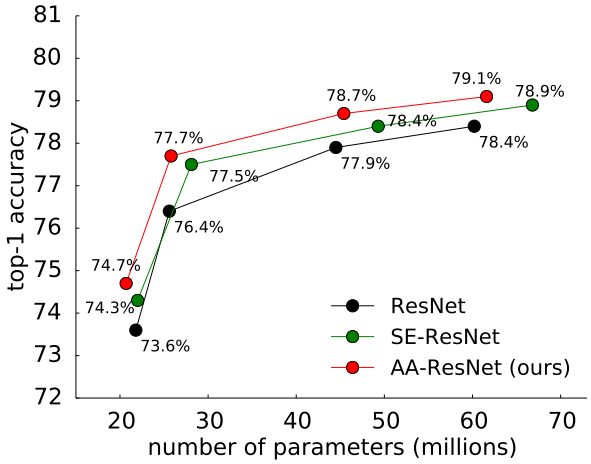

Figure 1. **Attention Augmentation systematically improves image classification across a large variety of networks of different scales.** ImageNet classification accuracy [9] versus the number of parameters for baseline models (ResNet) [14], models augmented with channel-wise attention (SE-ResNet) [17] and our proposed architecture (AA-ResNet).

Both these properties prove to be crucial inductive biases when designing models that operate over images. However, the local nature of the convolutional kernel prevents it from capturing global contexts in an image, often necessary for better recognition of objects in images [32].

Self-attention [41], on the other hand, has emerged as a recent advance to capture long range interactions, but has mostly been applied to sequence modeling and generative modeling tasks. The key idea behind self-attention is to produce a weighted average of values computed from hidden units. Unlike the pooling or the convolutional operator, the weights used in the weighted average operation are produced dynamically via a similarity function between hidden units. As a result, the interaction between input signals depends on the signals themselves rather than being predetermined by their relative location like in convolutions. In particular, this allows self-attention to capture long range

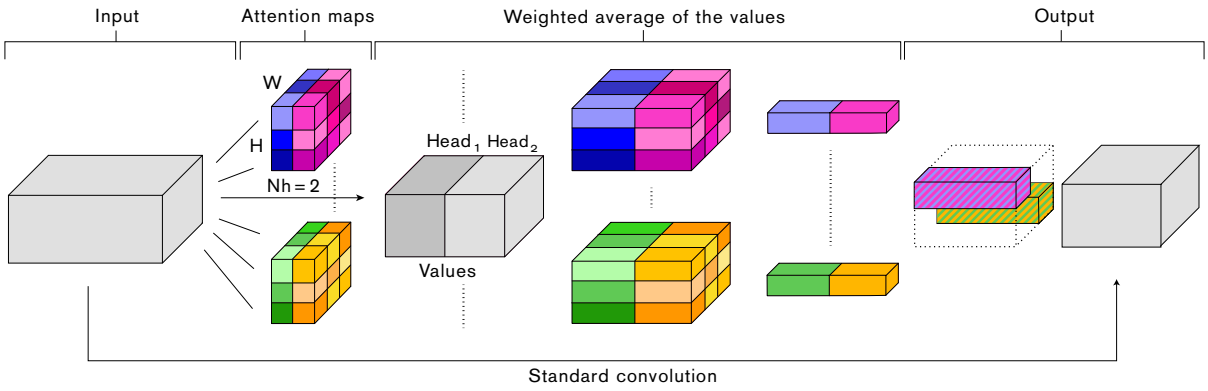

Figure 2. **Attention-augmented convolution**: For each spatial location (h, w), N_h attention maps over the image are computed from queries and keys. These attention maps are used to compute N_h weighted averages of the values V. The results are then concatenated, reshaped to match the original volume's spatial dimensions and mixed with a pointwise convolution. Multi-head attention is applied in parallel to a standard convolution operation and the outputs are concatenated.

interactions without increasing the number of parameters.

In this paper, we consider the use of self-attention for discriminative visual tasks as an alternative to convolutions. We develop a novel two-dimensional relative self-attention mechanism [35] that maintains translation equivariance while being infused with relative position information, making it well suited for images. Our self-attention formulation proves competitive for replacing convolutions entirely, however we find in control experiments that the best results are obtained when combining both. We therefore do not completely abandon the idea of convolutions, but instead propose to augment convolutions with this self-attention mechanism. This is achieved by concatenating convolutional feature maps, which enforce locality, to self-attentional feature maps capable of modeling longer range dependencies (see Figure 2).

We test our method on the CIFAR-100 and ImageNet classification [22, 9] and the COCO object detection [27] tasks, across a wide range of architectures at different computational budgets, including a state-of-the art resource constrained architecture [40]. Attention Augmentation yields systematic improvements with minimal additional computational burden and notably outperforms the popular Squeeze-and-Excitation [17] channelwise attention approach in all experiments. In particular, Attention Augmentation achieves a 1.3% top-1 accuracy ImageNet on top of a ResNet50 baseline and 1.4 mAP increase in COCO object detection on top of a RetinaNet baseline. Suprisingly, experiments also reveal that fully self-attentional models, a special case of Attention Augmentation, only perform slightly worse than their fully convolutional counterparts on ImageNet, indicating that self-attention is a powerful standalone computational primitive for image classification.

2. Related Work

2.1. Convolutional networks

Modern computer vision has been built on powerful image featurizers learned on image classification tasks such as CIFAR-10 [22] and ImageNet [9]. These datasets have been used as benchmarks for delineating better image featurizations and network architectures across a broad range of tasks [21]. For example, improving the "backbone" network typically leads to improvements in object detection [19] and image segmentation [6]. These observations have inspired the research and design of new architectures, which are typically derived from the composition of convolution operations across an array of spatial scales and skip connections [23, 39, 37, 38, 14, 45, 13]. Indeed, automated search strategies for designing architectures based on convolutional primitives result in state-of-the-art accuracy on large-scale image classification tasks that translate across a range of tasks [52, 21].

2.2. Attention mechanisms in networks

Attention has enjoyed widespread adoption as a computational module for modeling sequences because of its ability to capture long distance interactions [2, 42, 4, 3]. Most notably, Bahdanau et al. [2] first proposed to combine attention with a Recurrent Neural Network [15] for alignment in Machine Translation. Attention was further extended by Vaswani et al. [41], where the self-attentional Transformer architecture achieved state-of-the-art results in Machine Translation. Using self-attention in cooperation with convolutions is a theme shared by recent work in the natural language processing field [47]. For example, the QANet [48] and Evolved Transformer [36] architectures al-

```
def augmented_conv2d(X, Fout, k, dk, dv, Nh, relative):
    # X has shape [B, H, W, Fin]
    conv_out = tf.layers.conv2d(X, Fout - dv, k)
    # [B, Nh, HW, dvh or dkh]
    flat_q, flat_k, flat_v = compte_flat_qkv(X, dk, dv)
    # [B, Nh, HW, HW]
    logits = tf.matmul(flat_q, flat_k, transpose_b=True)
    if relative:
        h_rel_logits, w_rel_logits = relative_logits(q)
        logits += h_rel_logits
        logits += w_rel_logits
    weights = tf.nn.softmax(logits)
    # [B, Nh, HW, dvh]
    attn_out = tf.matmul(weights, flat_v)
    attn_out = tf.reshape(v, [B, Nh, H, W, dv // Nh])
    attn_out = combine_heads_2d(v) # [B, H, W, dv]
    attn_out = tf.layers.conv2d(attn_out, dv, 1)
    return tf.concat([conv_out, attn_out], axis=3)
```

Figure 3. Python code for our attention augmented 2D convolution. Helper functions can be found in the Appendix 9.

ternate between self-attention layers and convolution layers for Question Answering applications and Machine Translation respectively. Additionally, multiple attention mechanisms have been proposed for visual tasks to address the weaknesses of convolutions [17, 16, 7, 44, 43, 50]. For instance, Squeeze-and-Excitation [17] and Gather-Excite [16] reweigh feature channels using signals aggregated from entire feature maps, while BAM [30] and CBAM [44] refine convolutional features *independently* in the channel and spatial dimensions. In non-local neural networks [43], improvements are shown in video classification and object detection via the additive use of a few non-local residual blocks that employ self-attention in convolutional architectures. However, non-local blocks are only added to the architecture after ImageNet pretraining and are initialized in such a way that they do not break pretraining.

In contrast, our attention augmented networks do not rely on pretraining of their fully convolutional counterparts and employ self-attention along the entire architecture. The use of multi-head attention allows the model to attend *jointly* to both spatial and feature subspaces. Additionally, we enhance the representational power of self-attention over images by extending relative self-attention [35, 18] to two dimensional inputs allowing us to model translation equivariance in a principled way. Finally our method produces additional feature maps, rather than recalibrating convolutional features via addition [43, 50] or gating [17, 16, 30, 44]. This property allows us to flexibly adjust the fraction of attentional channels and consider a spectrum of architectures, ranging from fully convolutional to fully attentional models.

3. Methods

We now formally describe our proposed Attention Augmentation method. We use the following naming conventions: H, W and F_{in} refer to the height, width and number of input filters of an activation map. N_h, d_v and d_k respectively refer the number of heads, the depth of values and the depth of queries and keys in multihead-attention (MHA). We further assume that N_h divides d_v and d_k evenly and denote d_v^h and d_k^h the depth of values and queries/keys per attention head.

3.1. Self-attention over images

Given an input tensor of shape (H, W, F_{in}),[1] we flatten it to a matrix $X \in \mathbb{R}^{HW \times F_{in}}$ and perform multihead attention as proposed in the Transformer architecture [41]. The output of the self-attention mechanism for a single head h can be formulated as:

$$O_h = \text{Softmax}\left(\frac{(XW_q)(XW_k)^T}{\sqrt{d_k^h}}\right)(XW_v) \quad (1)$$

where $W_q, W_k \in \mathbb{R}^{F_{in} \times d_k^h}$ and $W_v \in \mathbb{R}^{F_{in} \times d_v^h}$ are learned linear transformations that map the input X to queries Q, keys K and values V.

The outputs of all heads are then concatenated and projected again as follows:

$$\text{MHA}(X) = \text{Concat}\left[O_1, \ldots, O_{Nh}\right] W^O \quad (2)$$

where $W^O \in \mathbb{R}^{d_v \times d_v}$ is a learned linear transformation. $\text{MHA}(X)$ is then reshaped into a tensor of shape (H, W, d_v) to match the original spatial dimensions. We note that multi-head attention incurs a complexity of $O((HW)^2 d_k)$ and a memory cost of $O((HW)^2 N_h)$ as it requires to store attention maps for each head.

3.1.1 Two-dimensional Position Encodings

Without explicit information about positions, self-attention is *permutation equivariant*:

$$\text{MHA}(\pi(X)) = \pi(\text{MHA}(X))$$

for any permutation π of the pixel locations, making it ineffective for modeling highly structured data such as images. Multiple positional encodings that augment activation maps with explicit spatial information have been proposed to alleviate related issues. In particular, the Image Transformer [31] extends the sinusoidal waves first introduced in the original Transformer [41] to 2 dimensional inputs and CoordConv [28] concatenates positional channels to an activation map.

However these encodings did not help in our experiments on image classification and object detection (see Section 4.5). We hypothesize that this is because such positional encodings, while not permutation equivariant, do not

[1] We omit the batch dimension for simplicity.

satisfy *translation equivariance*, which is a desirable property when dealing with images. As a solution, we propose to extend the use of relative position encodings [35] to two dimensions and present a memory efficient implementation based on the Music Transformer [18].

Relative positional encodings: Introduced in [35] for the purpose of language modeling, relative self-attention augments self-attention with relative position encodings and enables translation equivariance while preventing permutation equivariance. We implement two-dimensional relative self-attention by independently adding relative height information and relative width information to the original the attention logits in Equation 1 as:

$$O_h = \mathsf{Softmax}\left(\frac{QK^T}{\sqrt{d_k^h}} + S_H^{rel} + S_W^{rel}\right)V \qquad (3)$$

where $S_H^{rel}, S_W^{rel} \in \mathbb{R}^{HW \times HW}$ are matrices of relative position logits along height and width dimensions respectively.

The entries in S_H^{rel} satisfy $S_H^{rel}[i,j] = q_i(r_{ij}^H)^T$, where q_i is the i-th row of Q (the d_k^h dimensional query corresponding to position i in the flattened matrix Q) and r_{ij}^H is a learned d_k^h dimensional embedding of the relative height distance, in the original volume, between positions i and j in X. Similarly $S_W^{rel}[i,j] = q_i(r_{ij}^W)^T$, where r_{ij}^W is a learned d_k^h dimensional embedding of the relative width distance between the same pair of positions. This allows relative distances in height and width to modulate the attention distribution, affecting how information is combined, in a similar fashion to convolutions. As we consider relative height and width information separately, S_H^{rel} and S_W^{rel} also satisfy the properties $S_W^{rel}[i,j] = S_W^{rel}[i,j+W]$ and $S_H^{rel}[i,j] = S_H^{rel}[i+H,j]$, which prevents from having to compute the logits for all (i,j) pairs.

The relative attention algorithm in [35] explicitly stores all relative embeddings r_{ij} in a tensor of shape (HW, HW, d_k^h), thus incurring an additional memory cost of $O((HW)^2 d_k^h)$. This compares to $O((HW)^2 N_h)$ for the position-unaware version self-attention that does not use position encodings. As we typically have $N_h < d_k^h$, such an implementation can prove extremely prohibitive and restrict the number of images that can fit in a minibatch. Instead, we extend the memory efficient relative masked attention algorithm presented in [18] to unmasked relative self-attention over 2 dimensional inputs. Our implementation has a memory cost of $O(HWd_k^h)$. We leave the Tensorflow code of the algorithm in the Appendix.

The relative position representations are learned and shared across heads but not layers. For each layer, we add $(2(H+W)-2)d_k^h$ parameters to model relative distances along height and width.

3.2. Attention Augmented Convolution

Multiple previously proposed attention mechanisms over images [17, 16, 30, 44] suggest that the convolution operator is limited by its locality and lack of understanding of global contexts. These methods capture long-range dependencies by recalibrating convolutional feature maps. In particular, Squeeze-and-Excitation (SE) [17] and Gather-Excite (GE) [16] perform channelwise reweighing while BAM [30] and CBAM [44] reweigh both channels and spatial positions *independently*. In contrast to these approaches, we 1) use an attention mechanism that can attend *jointly* to spatial and feature subspaces (each head corresponding to a feature subspace) and 2) introduce additional feature maps rather than refining them. Figure 2 summarizes our proposed augmented convolution.

Concatenating convolutional and attentional feature maps: Formally, consider an original convolution operator with kernel size k, F_{in} input filters and F_{out} output filters. The corresponding attention augmented convolution can be written as

$$\mathsf{AAConv}(X) = \mathsf{Concat}\Big[\mathsf{Conv}(X), \mathsf{MHA}(X)\Big].$$

We denote $\upsilon = \frac{d_v}{F_{out}}$ the ratio of attentional channels to number of original output filters and $\kappa = \frac{d_k}{F_{out}}$ the ratio of key depth to number of original output filters. Similarly to the convolution, the proposed attention augmented convolution 1) is equivariant to translation and 2) can readily operate on inputs of different spatial dimensions. We include Tensorflow code for the proposed attention augmented convolution in Figure 3 (see Figure 9 in the Appendix for the helper functions.).

Effect on number of parameters: Multihead attention introduces a 1x1 convolution with F_{in} input filters and $(2d_k+d_v) = F_{out}(2\kappa+\upsilon)$ output filters to compute queries, keys and values and an additional 1x1 convolution with $d_v = F_{out}\upsilon$ input and output filters to mix the contribution of different heads. Considering the decrease in filters in the convolutional part, this leads to the following change in parameters:

$$\Delta_{params} \sim F_{in}F_{out}(2\kappa + (1-k^2)\upsilon + \frac{F_{out}}{F_{in}}\upsilon^2), \qquad (4)$$

where we ignore the parameters introduced by relative position embeddings for simplicity as these are negligible. In practice, this causes a slight decrease in parameters when replacing 3x3 convolutions and a slight increase in parameters when replacing 1x1 convolutions. Interestingly, we find in experiments that attention augmented networks still significantly outperform their fully convolutional counterparts while using less parameters.

Attention Augmented Convolutional Architectures: In all our experiments, the augmented convolution is followed by a batch normalization [20] layer which can learn to scale the contribution of the convolution feature maps and the attention feature maps. We apply our augmented convolution once per residual block similarly to other visual attention mechanisms [17, 16, 30, 44] and along the entire architecture as memory permits (see Section 4 for more details).

Since the memory cost $O((N_h(HW)^2)$ can be prohibitive for large spatial dimensions, we augment convolutions with attention starting from the last layer (with smallest spatial dimension) until we hit memory constraints. To reduce the memory footprint of augmented networks, we typically resort to a smaller batch size and sometimes additionally downsample the inputs to self-attention in the layers with the largest spatial dimensions where it is applied. Downsampling is performed by applying 3x3 average pooling with stride 2 while the following upsampling (required for the concatenation) is obtained via bilinear interpolation.

4. Experiments

In the subsequent experiments, we test Attention Augmentation on standard computer vision architectures such as ResNets [14, 45, 13], and MnasNet [40] on the CIFAR-100 [22], ImageNet [9] and COCO [25] datasets. Our experiments show that Attention Augmentation leads to systematic improvements on both image classification and object detection tasks across a broad array of architectures and computational demands. We validate the utility of the proposed two-dimensional relative attention mechanism in ablation experiments. In all experiments, we substitute convolutional feature maps with self-attention feature maps as it makes for an easier comparison against the baseline models. Unless specified otherwise, all results correspond to our two-dimensional relative self-attention mechanism.

4.1. CIFAR-100 image classification

We first investigate how Attention Augmentation performs on CIFAR-100 [22], a standard benchmark for low-resolution imagery, using a Wide ResNet architecture [49]. The Wide-ResNet-28-10 architecture is comprised of 3 stages of 4 residual blocks each using two 3×3 convolutions. We augment the Wide-ResNet-28-10 by augmenting the first convolution of all residual blocks with relative attention using N_h=8 heads and κ=2υ=0.2 and a minimum of 20 dimensions per head for the keys. Given the low resolution of CIFAR-100 images, we do not downsample feature maps before the attention operation and instead resort to a smaller batch size. We train all networks for 500 epochs using synchronous SGD with momentum 0.9 distributed across 8 TESLA V100 GPUs. The learning rate is linearly scaled from 0 to $0.2B/256$, where B is the total batch size, for the first 5% training epochs and then annealed with cosine decay [29]. We use standard CIFAR preprocessing: mean normalizing, random flipping and cropping [52, 10, 46]. For the non-augmented architectures, we use a batch size of 1024 and a weight decay of 2e-4. When using Attention Augmentation, the batch size is set to 256 and the weight decay is set to 5e-4.

We compare Attention Augmentation (AA) against other forms of attention including Squeeze-and-Excitation (SE) [17] and the parameter-free formulation of Gather-Excite (GE) [16]. Table 1 shows that Attention Augmentation improves performance both over the baseline network and Squeeze-and-Excitation at a similar parameter and complexity cost.

Architecture	Params	GFlops	top-1	top-5
Wide-ResNet [49]	36.3M	10.4	80.3	95.0
GE-Wide-ResNet [16]	36.3M	10.4	79.8	95.0
SE-Wide-ResNet [17]	36.5M	10.4	81.0	95.3
AA-Wide-ResNet (ours)	36.2M	10.9	81.6	95.2

Table 1. Image classification on the CIFAR-100 dataset [22] using the Wide-ResNet 28-10 architecture [49]. We measure top-1 and top-5 accuracy on a baseline model against the parameter-less formulation of Gather-Excite (GE) [16], Squeeze-Excite (SE) [17] and the proposed Attention Augmentation (AA).

4.2. ImageNet image classification with ResNet

We next examine how Attention Augmentation performs on ImageNet[9, 21], a standard large-scale dataset for high resolution imagery, across an array of architectures. We start with the ResNet architecture [14, 45, 13] because of its widespread use and its ability to easily scale across several computational budgets. The building block in ResNet-34 comprises two 3x3 convolutions with the same number of output filters. ResNet-50 and its larger counterparts use a bottleneck block comprising of 1x1, 3x3, 1x1 convolutions where the last pointwise convolution expands the number of filters and the first one contracts the number of filters. We modify all ResNets by augmenting the 3x3 convolutions as this decreases number of parameters.[2] We apply Attention Augmentation in each residual block of the last 3 stages of the architecture – when the spatial dimensions of the activation maps are 28x28, 14x14 and 7x7 – and downsample only during the first stage. All attention augmented networks use κ=2υ=0.2, except for ResNet-34 which uses κ=υ=0.25. The number of attention heads is fixed to N_h=8. We train all ResNet architectures for 100 epochs using synchronous SGD with momentum 0.9 across 8 TESLA V100 GPUs and weight decay of 1e-4. We use the largest batch size per worker $B \in \{32, 64, 128, 256\}$ that fits in a minibatch. The initial learning rate is scaled linearly according

[2] We found that augmenting the pointwise expansions works just as well but does not save parameters or computations.

to the total batch size using a base learning rate of 0.128 for total batch size of 256. During training, we linearly scale the learning rate from 0 to this value for the first 5% of training epochs and divide it by 10 at epochs 30, 60, 80 and 90. We use standard Inception data augmentation as described in [39].

In all experiments, Attention Augmentation significantly increases performance over the non-augmented baseline and notably outperforms Squeeze-and-Excitation (SE) [17] while being more parameter efficient (see Table 2 and Figure Figure 1). Remarkably, our AA-ResNet-50 performs comparably to the baseline ResNet-101 and our AA-ResNet-101 outperforms the baseline ResNet-152. These results suggest that attention augmentation is preferable to simply making networks deeper. We include and discuss attention maps visualizations from different pixel positions in the appendix.

Architecture	GFlops	Params	top-1	top-5
ResNet-34 [14]	7.4	21.8M	73.6	91.5
SE-ResNet-34 [17]	7.4	22.0M	74.3	91.8
AA-ResNet-34 (ours)	7.1	20.7M	74.7	92.0
ResNet-50 [14]	8.2	25.6M	76.4	93.1
SE-ResNet-50 [17]	8.2	28.1M	77.5	93.7
AA-ResNet-50 (ours)	8.3	25.8M	77.7	93.8
ResNet-101 [14]	15.6	44.5M	77.9	94.0
SE-ResNet-101 [17]	15.6	49.3M	78.4	94.2
AA-ResNet-101 (ours)	16.1	45.4M	78.7	94.4
ResNet-152 [14]	23.0	60.2M	78.4	94.2
SE-ResNet-152 [17]	23.1	66.8M	78.9	94.5
AA-ResNet-152 (ours)	23.8	61.6M	79.1	94.6

Table 2. Image classification on the ImageNet dataset [9] across a range of ResNet architectures: ResNet-34, ResNet-50, Resnet-101, and ResNet-152 [14, 45, 13].

4.3. ImageNet classification with MnasNet

In this section, we inspect the use of Attention Augmentation in a resource constrained setting by conducting ImageNet experiments with the MnasNet architecture [40], which is an extremely parameter-efficient architecture. In particular, the MnasNet was found by neural architecture search [51], using only the highly optimized mobile inverted bottleneck block [34] and the Squeeze-and-Excitation operation [17] as the primitives in its search space. We follow the training setup described in [40] and train all networks for 350 epochs with the RMSProp optimizer using exponential learning rate decay. When training our augmented MnasNets, we divided the learning by 2 and adjusted the learning rate decay so that the final learning rate stays the same. We apply Attention Augmentation to the mobile inverted bottleneck by replacing convolutional channels in the expansion pointwise convolution using κ=2υ=0.1 and N_h=4. Our augmented MnasNets use

Architecture	GFlops	Params	top-1	top-5
MnasNet-0.75	0.45	2.91M	73.3	91.3
AA-MnasNet-0.75	0.51	3.02M	73.9	91.6
MnasNet-1.0	0.63	3.89M	75.2	92.4
AA-MnasNet-1.0	0.70	4.06M	75.7	92.6
MnasNet-1.25	1.01	5.26M	76.7	93.2
AA-MnasNet-1.25	1.11	5.53M	77.2	93.6
MnasNet-1.4	1.17	6.10M	77.2	93.5
AA-MnasNet-1.4	1.29	6.44M	77.7	93.8

Table 3. Baseline and attention augmented MnasNet [40] accuracies with width multipliers 0.75, 1.0, 1.25 and 1.4.

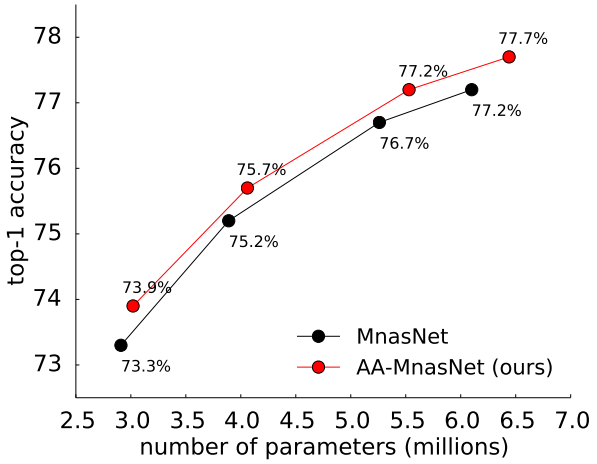

Figure 4. ImageNet top-1 accuracy as a function of number of parameters for MnasNet (black) and Attention-Augmented-MnasNet (red) with depth multipliers 0.75, 1.0, 1.25 and 1.4.

augmented inverted bottlenecks in the the last 13 blocks out of 18 in the MnasNet architecture, starting when the spatial dimension is 28x28. We downsample only in the first stage where Attention Augmentation is applied. We leave the final pointwise convolution, also referred to as the "head", unchanged.

In Table 3, we report ImageNet accuracies for the baseline MnasNet and its attention augmented variants at different width multipliers. Our experiments show that Attention Augmentation yields accuracy improvements across all width multipliers. Augmenting MnasNets with relative self-attention incurs a slight parameter increase, however we verify in Figure 4 that the accuracy improvements are not just explained by the parameter increase. Additionally, we note that the MnasNet architecture employs Squeeze-and-Excitation at multiple locations that were optimally selected via architecture search, further suggesting the benefits of our method.

Backbone architecture	GFlops	Params	mAP_{COCO}	mAP_{50}	mAP_{75}
ResNet-50 [26]	182	33.4M	36.8	54.5	39.5
SE-ResNet-50 [17]	183	35.9M	36.5	54.0	39.1
AA-ResNet-50 (ours)	182	33.1M	38.2	56.5	40.7
ResNet-101 [26]	243	52.4M	38.5	56.4	41.2
SE-ResNet-101 [17]	243	57.2M	37.4	55.0	39.9
AA-ResNet-101 (ours)	245	51.7M	39.2	57.8	41.9

Table 4. Object detection on the COCO dataset [27] using the RetinaNet architecture [26] with different backbone architectures. We report mean Average Precision at three different IoU values.

4.4. Object Detection with COCO dataset

We next investigate the use of Attention Augmentation on the task of object detection on the COCO dataset [27]. We employ the RetinaNet architecture with a ResNet-50 and ResNet-101 backbone as done in [26], using the open-sourced RetinaNet codebase.[3] We follow the setup described in [26, 11] and train the RetinaNet from scratch for 150 epochs without using ImageNet pretraining for the ResNet backbone. We use the preprocessing pipeline described in [26]. We apply multiscale jitter, randomly resize images from [512, 768] and crop to a max dimension of 640 during training. All images are horizontally flipped with a 50% probability. We apply Attention Augmentation uniquely on the ResNet backbone, modifying them similarly as in our ImageNet classification experiments. Our relative self-attention mechanism improves the performance of the RetinaNet on both ResNet-50 and ResNet-101 as shown in Table 4. Most notably, Attention Augmentation yields a 1.4% mAP improvement over a strong RetinaNet baseline from [26].

In contrast to the success of Squeeze-and-Excitation in image classification with ImageNet, our experiments show that adding Squeeze-and-Excitation operators in the backbone network of the RetinaNet significantly hurts performance, in spite of grid searching over the squeeze ratio $\sigma \in \{4, 8, 16\}$. We hypothesize that localization requires precise spatial information which SE discards during the spatial pooling operation, thereby negatively affecting performance. Self-attention on the other hand maintains spatial information and is likely to be able to identify object boundaries successfully. Visualizations of attention maps (See Figures 10 and 11 in the Appendix) reveal that some heads are indeed delineating objects from their background which might be important for localization.

4.5. Ablation Study

Fully-attentional vision models: In this section, we investigate the performance of Attention Augmentation as a function of the fraction of attentional channels. As we increase this fraction to 100%, we begin to replace a ConvNet with a fully attentional model, only leaving pointwise convolutions and the stem unchanged. Table 5 presents the performance of Attention Augmentation on the ResNet-50 architecture for varying ratios κ=$\upsilon \in \{0.25, 0.5, 0.75, 1.0\}$. Performance slightly degrades as the ratio of attentional channels increases, which we hypothesize is partly explained by the average pooling operation for downsampling at the first stage where Attention Augmentation is applied. Attention Augmentation proves however quite robust to the fraction of attentional channels. For instance, AA-ResNet-50 with κ=υ=0.75 outperforms its ResNet-50 counterpart, while being more parameter and flops efficient, indicating that mostly employing attentional channels is readily competitive.

Perhaps surprisingly, these experiments also reveal that our proposed self-attention mechanism is a powerful standalone computational primitive for image classification and that fully attentional models may be viable for discriminative visual tasks. In particular, AA-ResNet-50 with κ=υ=1, which uses exclusively attentional channels, is only 2.5% worse in accuracy than its fully convolutional counterpart, in spite of downsampling with average pooling and having 25% less parameters. Notably, this fully attentional architecture[4] also outperforms ResNet-34 while being more parameter and flops efficient (see Table 5).

Architecture	GFlops	Params	top-1	top-5
ResNet-34 [14]	7.4	21.8M	73.6	91.5
ResNet-50 [14]	8.2	25.6M	76.4	93.1
$\kappa = \upsilon = 0.25$	7.9	24.3M	77.7	93.8
$\kappa = \upsilon = 0.5$	7.3	22.3M	77.3	93.6
$\kappa = \upsilon = 0.75$	6.8	20.7M	76.7	93.2
$\kappa = \upsilon = 1.0$	6.3	19.4M	73.9	91.5

Table 5. Attention Augmented ResNet-50 with varying ratios of attentional channels.

[3] https://github.com/tensorflow/tpu/tree/master/models/official/retinanet

[4] We consider pointwise convolutions as dense layers. This architecture employs 4 non-pointwise convolutions in the stem and the first stage of the architecture, but we believe such operations can be replaced by attention too.

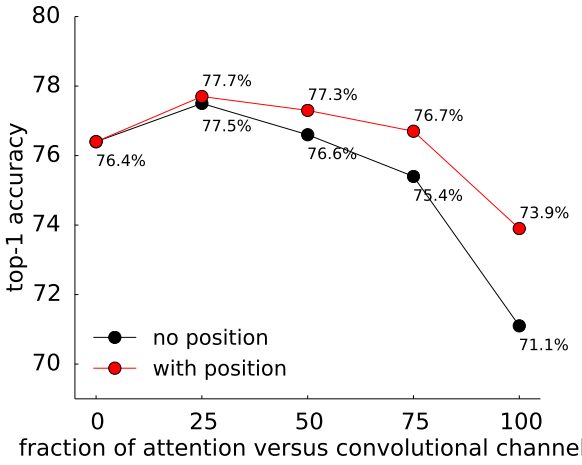

Figure 5. Effect of relative position embeddings as the ratio of attentional channels increases on our Attention-Augmented ResNet50.

Importance of position encodings: In Figure 5, we show the effect of our proposed two-dimensional relative position encodings as a function of the fraction of attentional channels. As expected, experiments demonstrate that our relative position encodings become increasingly more important as the architecture employs more attentional channels. In particular, the fully self-attentional ResNet-50 gains 2.8% top-1 ImageNet accuracy when using relative position encodings, which indicates the necessity of maintaining position information for fully self-attentional vision models.

We additionally compare our proposed two-dimensional relative position encodings to other position encoding schemes. We apply Attention Augmentation using the same hyperparameters as 4.2 with the following different position encoding schemes: 1) The position-unaware version of self-attention (referred to as *None*), 2) a two-dimensional implementation of the sinusoidal positional waves (referred to as *2d Sine*) as used in [31], 3) CoordConv [28] for which we concatenate (x,y,r) coordinate channels to the inputs of the attention function, and 4) our proposed two-dimensional relative position encodings (referred to as *Relative*).

In Table 6 and 7, we present the results on ImageNet classification and the COCO object detection task respectively. On both tasks, Attention Augmentation without position encodings already yields improvements over the fully convolutional non-augmented variants. Our experiments also reveal that the sinusoidal encodings and the coordinate convolution do not provide improvements over the position-unaware version of Attention Augmentation. We obtain additional improvements when using our two-dimensional relative attention, demonstrating the utility of preserving translation equivariance while preventing permutation equivariance.

Architecture	Position Encodings	top-1	top-5
AA-ResNet-34	None	74.4	91.9
AA-ResNet-34	2d Sine	74.4	92.0
AA-ResNet-34	CoordConv	74.4	92.0
AA-ResNet-34	Relative (ours)	74.7	92.0
AA-ResNet-50	None	77.5	93.7
AA-ResNet-50	2d Sine	77.5	93.7
AA-ResNet-50	CoordConv	77.5	93.8
AA-ResNet-50	Relative (ours)	77.7	93.8

Table 6. Effects of different position encodings in Attention Augmentation on ImageNet classification.

Position Encodings	mAP_{COCO}	mAP_{50}	mAP_{75}
None	37.7	56.0	40.2
CoordConv [28]	37.4	55.5	40.1
Relative (ours)	38.2	56.5	40.7

Table 7. Effects of different position encodings in Attention Augmentation on the COCO object detection task using a RetinaNet AA-ResNet-50 backbone.

5. Discussion and future work

In this work, we consider the use of self-attention for vision models as an alternative to convolutions. We introduce a novel two-dimensional relative self-attention mechanism for images that enables training of competitive fully self-attentional vision models on image classification for the first time. We propose to augment convolutional operators with this self-attention mechanism and validate the superiority of this approach over other attention schemes. Extensive experiments show that Attention Augmentation leads to systematic improvements on both image classification and object detection tasks across a wide range of architectures and computational settings.

Several open questions from this work remain. In future work, we will focus on the fully attentional regime and explore how different attention mechanisms trade off computational efficiency versus representational power. For instance, identifying a *local* attention mechanism may result in an efficient and scalable computational mechanism that could prevent the need for downsampling with average pooling. Additionally, it is plausible that architectural design choices that are well suited when exclusively relying on convolutions are suboptimal when using self-attention mechanisms. As such, it would be interesting to see if using Attention Augmentation as a primitive in automated architecture search procedures proves useful to find even better models than those previously found in image classification [52], object detection [12], image segmentation [6] and other domains[5, 1, 33, 8]. Finally, one can ask to which degree fully attentional models can replace convolutional networks for visual tasks.

Acknowledgements

The authors would like to thank Tsung-Yi Lin, Prajit Ramachandran, Mingxing Tan, Yanping Huang and the Google Brain team for insighful comments and discussions.

References

[1] M. Alber, I. Bello, B. Zoph, P. Kindermans, P. Ramachandran, and Q. V. Le. Backprop evolution. *CoRR*, abs/1808.02822, 2018. 8

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015. 2

[3] I. Bello, S. Kulkarni, S. Jain, C. Boutilier, E. H. Chi, E. Eban, X. Luo, A. Mackey, and O. Meshi. Seq2slate: Re-ranking and slate optimization with rnns. *CoRR*, abs/1810.02019, 2018. 2

[4] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. 2016. 2

[5] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le. Neural optimizer search with reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 459–468. JMLR.org, 2017. 8

[6] L.-C. Chen, M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *Advances in Neural Information Processing Systems*, pages 8713–8724, 2018. 2, 8

[7] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng. A^2-nets: Double attention networks. *CoRR*, abs/1810.11579, 2018. 3

[8] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018. 8

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009. 1, 2, 5, 6

[10] X. Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017. 5

[11] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pages 10750–10760, 2018. 7

[12] G. Ghiasi, T.-Y. Lin, R. Pang, and Q. V. Le. NAS-FPN: Learning scalable feature pyramid architecture for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 8

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2, 5, 6

[14] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016. 1, 2, 5, 6, 7, 11

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997. 2

[16] J. Hu, L. Shen, S. Albanie, G. Sun, and A. Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 9423–9433, 2018. 3, 4, 5

[17] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2, 3, 4, 5, 6, 7

[18] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck. Music transformer. In *Advances in Neural Processing Systems*, 2018. 3, 4

[19] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Learning Representations*, 2015. 5

[21] S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2, 5

[22] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 2, 5

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing System*, 2012. 1, 2

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 1

[25] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 5

[26] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 7

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 2, 6, 7

[28] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, pages 9628–9639, 2018. 3, 8

[29] I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 5

[30] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon. Bam: bottleneck attention module. In *British Machine Vision Conference*, 2018. 3, 4, 5

[31] N. Parmar, A. Vaswani, J. Uszkoreit, Ł. Kaiser, N. Shazeer, A. Ku, and D. Tran. Image transformer. In *International Conference on Machine Learning*, 2018. 3, 8

[32] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. 2007. 1

[33] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. 8

[34] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 6

[35] P. Shaw, J. Uszkoreit, and A. Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018. 2, 3, 4

[36] D. R. So, C. Liang, and Q. V. Le. The evolved transformer. *CoRR*, abs/1901.11117, 2019. 2

[37] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, Inception-Resnet and the impact of residual connections on learning. In *International Conference on Learning Representations Workshop Track*, 2016. 2

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2

[39] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2, 6

[40] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2, 5, 6

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 1, 2, 3

[42] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In *NIPS*, pages 2692–2700, 2015. 2

[43] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018. 3

[44] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 3, 4, 5

[45] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 5, 6

[46] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise. Shakedrop regularization for deep residual learning. *arXiv preprint arXiv:1802.02375*, 2018. 5

[47] B. Yang, L. Wang, D. F. Wong, L. S. Chao, and Z. Tu. Convolutional self-attention network. In *CoRR*, volume abs/1810.13320, 2018. 2

[48] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le. QAnet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations*, 2018. 2

[49] S. Zagoruyko and N. Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016. 5

[50] H. Zhang, I. J. Goodfellow, D. N. Metaxas, and A. Odena. Self-attention generative adversarial networks. *arXiv:1805.08318*, 2018. 3

[51] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. 6

[52] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 2, 5, 8

A. Appendix

A.1. Accuracy as a function of floating point operations.

In the paper, we show the accuracies of our method across parameter counts. Parameter count is one of many metrics to measure to complexity of a model. The other metric that researchers often use is FLOPS, which correlates with running times. Figures 6 and 7 show the accuracies of our attention augmented networks across FLOPS counts.

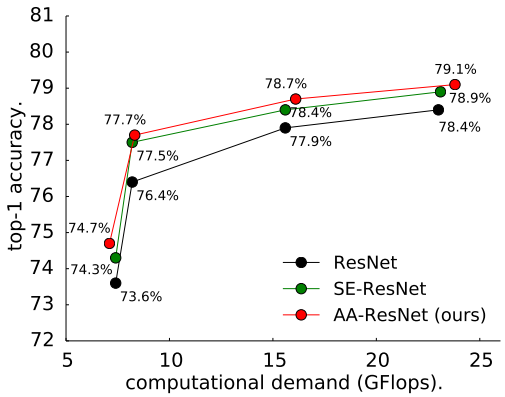

Figure 6. ImageNet top-1 accuracy as a function of computational demand for variety of ResNet architectures [14]. From left to right: ResNet-34, ResNet-50, ResNet-101 and ResNet-152.

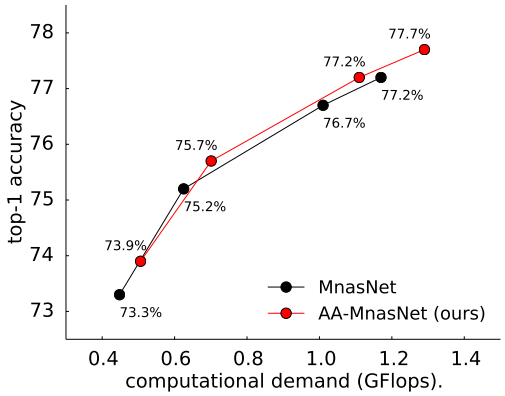

Figure 7. ImageNet top-1 accuracy as a function of computational demand for MnasNet (black) and Attention-Augmented-MnasNet (red) with depth multipliers 0.75, 1.0, 1.25 and 1.4.

A.2. Helper functions for two-dimensional multi-head attention

```
def rel_to_abs(x):
  """Converts tensor from relative to absolute indexing."""
  # [B, Nh, L, 2L−1]
  B, Nh, L, _ = x.shape

  # Pad to shift from relative to absolute indexing.
  col_pad = tf.zeros((B, Nh, L, 1))
  x = tf.concat([x, col_pad], axis=3)
  flat_x = tf.reshape(x, [B, Nh, L * 2 * L])
  flat_pad = tf.zeros((B, Nh, L−1))
  flat_x_padded = tf.concat([flat_x, flat_pad], axis=2)

  # Reshape and slice out the padded elements.
  final_x = tf.reshape(flat_x_padded, [B, Nh, L+1, 2*L−1])
  final_x = final_x[:, :, :L, L−1:]
  return final_x

def relative_logits_1d(q, rel_k, H, W, Nh, transpose_mask):
  """Compute relative logits along one dimension."""
  # [B, Nh, H, W, 2*W−1]
  rel_logits = tf.einsum('bhxyd,md−>bhxym', q, rel_k)
  # Collapse height and heads
  rel_logits = tf.reshape(rel_logits, [−1, Nh*H, W, 2*W−1])
  rel_logits = rel_to_abs(rel_logits)
  # Shape it back and tile height times
  rel_logits = tf.reshape(rel_logits, [−1, Nh, H, W, W])
  rel_logits = tf.expand_dims(rel_logits, axis=3)
  rel_logits = tf.tile(rel_logits, [1, 1, 1, H, 1, 1])
  # Reshape for adding to the attention logits.
  rel_logits = tf.transpose(rel_logits, transpose_mask)
  rel_logits = tf.reshape(rel_logits, [−1, Nh, H*W, H*W])
  return rel_logits

def relative_logits(q):
  """Compute relative position logits."""
  # [B, Nh, H, W, dk]
  dk = q.shape[−1]

  # Relative logits in width dimension.
  key_rel_w = tf.get_variable(
      'key_rel_w', shape=(2*W−1, dk),
      initializer=tf.random_normal_initializer(dk**−0.5))
  rel_logits_w = relative_logits_1d(
      q, key_rel_w, H, W, [0, 1, 2, 4, 3, 5])

  # Relative logits in height dimension.
  # For ease, we transpose height and width and repeat the
  # above steps, and transpose to eventually put the logits
  # in the correct positions.
  key_rel_h = tf.get_variable(
      'key_rel_h', shape=(2*H−1, dk),
      initializer=tf.random_normal_initializer(dk**−0.5))
  rel_logits_h = relative_logits_1d(
      tf.transpose(q, [0, 1, 3, 2, 4]),
      key_rel_h, W, H, [0, 1, 4, 2, 5, 3])

  return rel_logits_h, rel_logits_w
```

Figure 8. Tensorflow code for computing the relative logits. The attention calculation is augmented with learned representations for the relative position between each element in q and each element in k and v in height and width dimensions. For query index (i,j) and key index (l, m), the logit is $q_{ij}k_{lm}^T + q_{ij}(r_{l-i}^H)^T + q_{ij}(r_{m-j}^W)^T$, where r^H and r^W are the set of relative embeddings in height and width spatial dimensions.

```python
def split_heads_2d(inputs, Nh):
    """Split channels into multiple heads."""
    s = inputs_shape[:-1]
    ret_shape = s + [Nh, s // Nh]
    split = tf.reshape(inputs, ret_shape)
    return tf.transpose(split, [0, 3, 1, 2, 4])

def combine_heads_2d(inputs):
    """Combine heads (inverse of split_heads_2d)"""
    transposed = tf.transpose(inputs, [0, 2, 3, 1, 4])
    a, b = transposed.shape[-2:]
    ret_shape = transposed.shape[:-2] + [a * b]
    return tf.reshape(transposed, ret_shape)

def compute_flat_qkv(inputs, dk, dv, Nh):
    """Computes flattened queries, keys and values."""
    N, H, W, _ = inputs.shape
    qkv = tf.layers.conv2d(inputs, 2*dk + dv, 1)
    q, k, v = tf.split(qkv, [dk, dk, dv], axis=3)
    q = split_heads_2d(q, Nh)
    k = split_heads_2d(k, Nh)
    v = split_heads_2d(v, Nh)
    dkh = dk // Nh
    q *= dkh ** -0.5
   flat_q = tf.reshape(q, [N, Nh, H * W, dk])
   flat_k = tf.reshape(k, [N, Nh, H * W, dk])
   flat_v = tf.reshape(v, [N, Nh, H * W, dv])
   return flat_q, flat_k, flat_v
```

Figure 9. Helper functions in Tensorflow for 2d Multi-Head-Attention.

A.3. Attention visualizations.

In Figure 11, we present attention maps visualizations for the input image shown in Figure 10. We see that attention heads learn to specialize to different content and notably can delineate object boundaries.

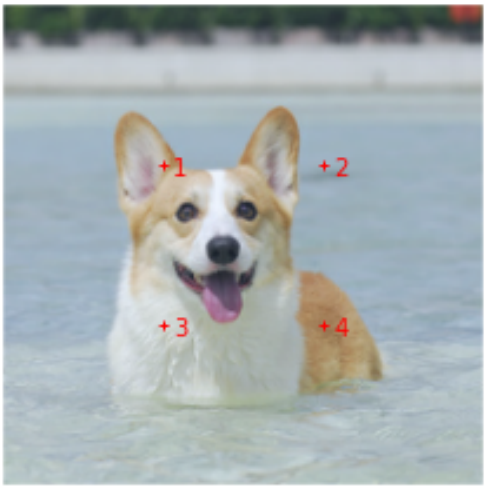

Figure 10. An input image. The red crosses indexed 1 to 4 represent the pixel locations for which we show the attention maps in Figure 11.

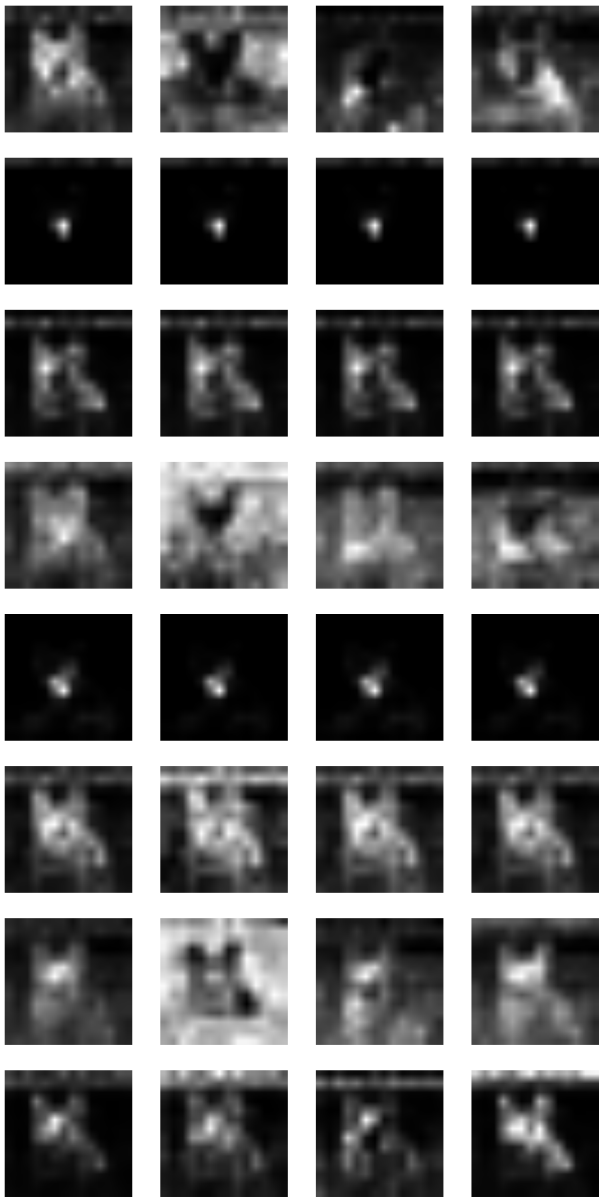

Figure 11. Visualization of attention maps for an augmented convolution in the Attention-Augmented-ResNet50. Rows correspond to the 8 different heads and columns correspond to the 4 pixel locations depicted in the input image (See Figure 10).