EXPANDING VISUAL ATTENTION

Can Küçüksözen
Department of Computer Science and Engineering
Koç University, İstanbul
ckucuksozen19@ku.edu.tr

ABSTRACT

Convolutional Neural Networks (CNNs) are considered to be the core building module of almost any Computer Vision related Neural Network architecture nowadays, after its successful premiere in 2012 on ImageNet Challenge, Krizhevsky et al. (2012). After a considerable amount of time inspecting the architectures and primary building blocks that are involved in Deep Learning applications of core Computer Vision tasks, one can see that Convolutional Layers and Networks paired with sliding window fashioned processing pipelines are the most widely adopted. Despite their fundamental advantages on spatially correlated data granted by the weight sharing and sparse connectivity properties, convolutional layers suffer from their poor channel-wise information projecting capabilities. Stemmed from this fundamental disadvantage of convolutions, researchers tried to go beyond convolutions and capture *long-range dependencies* between intermediate activation volumes. Most of these efforts focus on augmenting convolutional models with *content-based* interactions such as *attention* based methods. In light of aforementioned observations, this work aims to show that *attention* based methods can indeed be used on top of convolutional layers in order to effectively search the long range interactions inside the huge feature volumes created by convolutional layers. While doing so, this work shows that making use of basic natural intuitions about how humans perceive the world, benefits the accuracy and decreases the computational workload of the proposed Image Classification model.

1 INTRODUCTION

The construction of large datasets containing tens of thousands of labeled training samples and recent advances in computing resources, made CNNs the backbone for many computer vision applications. The field of deep learning has in turn largely shifted toward the design of architectures of CNNs for improving the performance on tasks such as image classification, object detection and image segmentation. In their work, Ramachandran et al. (2019), authors claim that the translation equivariance property of convolutions has provided a strong motivation for adopting them as a building block for operating on images. However, capturing long range interactions for convolutions is challenging because of their poor scaling properties with respect to large receptive fields.

To illustrate a numerical example, imagine a convolutional layer of shape $[128, 256, 5, 5]$ operating on an input activation volume of size $[batch\_size, 128, 32, 32]$. This layer learns 256 different weight volumes of shape $[128, 5, 5]$ and tries to find a *match* for these 256 different 3D templates via element-wise dot-product across multiple spatial positions of the input activation volume. After training, the aforementioned convolutional layer is capable of expressing an input distribution in $5 \times 5 \times 128$ dimensional space in close proximity to 256 different points. In other words, convolutional layers that are placed in the deeper layers of the network become more and more limited in their expressivity when translating input spatial information into output messages in channel-wise dimension. Hence, it can be argued that most of the state-of-the-art convolutional architectures that are proposed during the last decade generate a huge amount of important spatial "clues" in the earlier layers of CNNs, then fail to propagate these important clues onto downstream inference tasks because of poor scaling properties of deeper convolutional layers.

After a brief discussion on pros/cons of the convolutional layers, one might claim that the limited expressivity of convolutional layers while transferring information is due to their lack of fully capturing long range interactions between the channel-wise nodes of an input activation volume. To address this specific problem, many researchers made use of "Attention" based methodologies in various deep learning applications. Furthermore, "Attention" based networks are also used in conjunction with convolutional layers in vision based tasks, such as the stand-alone self-attention module that is proposed by Ramachandran et al. (2019). However most of these attention based vision related deep learning architectures operate on single kernel size in a given layer and then try to propagate and downsample the outputs of such a layer with "noisy" operations such average pooling. One might argue that conducting attention based operations for every pixel in the input volume and then combining these results with averaging operations causes unnecessary computational overhead and poor/noisy information passing between layers.

This work firstly experiments with a novel attentional layer that is capable of learning patterns at expanding kernel sizes and summarizes the information contained in this fairly bigger kernel sized area into a 1d vector with a considerably low computational overhead. While doing so, this layer is intended to capture underlying visual symmetries in the aforementioned fairly large kernel sized input patch. This is carried out with the layers' sequential processing of symmetric 3x3, 5x5 and 7x7 patch memory blocks.

It should be noted that, this work measures its proposed architectures performance on the Image Classification task by making use of a smaller dataset called CIFAR-10. CIFAR-10 is an Image Classification dataset that is made up of 60000 32x32 pixel images containing instances of 10 different object classes, Krizhevsky et al. (2008). In order to objectively evaluate the performance of aforementioned models on the CIFAR-10 Dataset, baseline versions of ResNet and SelfAttnNet(SANet) by Ramachandran et al. (2019) models, are implemented and evaluated.

The following sections include a background on convolution and self-attention operations, their implementations in vision based tasks, a detailed explanation of the implemented models in this work, display of the conducted experiments, their results, conclusion and discussion.

2 BACKGROUND

This section introduces background knowledge on aforementioned convolution and self-attention operations and conveys a detailed discussion about how they operate, the similarities and differences between them.

2.1 CONVOLUTION

Convolutional Neural Networks generally operate on small neighborhoods (i.e. kernel sizes) of pixels to encourage the network to learn local correlation structures within a particular layer. Given an input $x \in \Re^{h \times w \times d}$ in with height h, width w, and input channels d_{in}, a local neighborhood N_k around a pixel x_{ij} is extracted with spatial extent k, resulting in a region with shape $k \times k \times d_{in}$ (see Figure 1).

Given a learned weight matrix of a convolutional layer, $W \in \Re^{k \times k \times d_{out} \times d_{in}}$, the output $y_{ij} \in \Re^{d_{out}}$ for position ij is defined by spatially summing the product of depthwise matrix multiplications of the input values:

$$y_{ij} = \sum_{a,b \in N_k(i,j)} W_{i-a,j-b}\, x_{ab} \tag{1}$$

where $N_k(i,j) = \{a, b \mid |a-i| \leq k/2\ ,\ |b-j| \leq k/2\}$ (see Figure 2). One of the most important characteristics of a CNN layer is that it employs weight sharing, where a single weight matrix W is reused for generating the output for all pixel positions of ij in a single channel dimension. In addition, weight sharing enforces translation equivariance in the learned representation and consequently decouples the parameter count of the convolution from the input size.

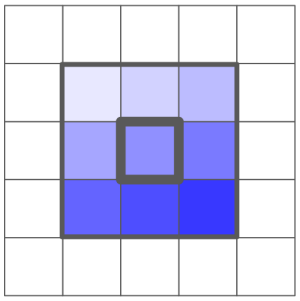

Figure 1: An example of a local window around $i = 3, j = 3$ (one-indexed) with spatial extent $k = 3$.

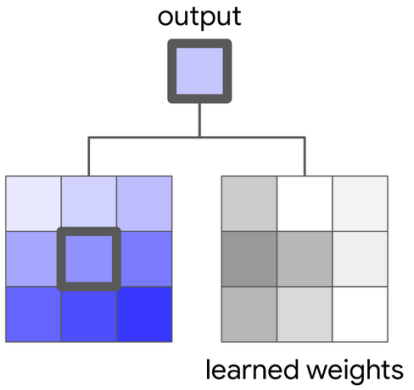

Figure 2: An example of a 3×3 convolution operation. The output is the inner product between the local window and the learned weights.

2.2 SELF-ATTENTION

The concept of *Attention* in Deep Learning applications was introduced by Bahdanau et al. (2014). "It was used by an encoder-decoder neural machine translation model to allow for content-based summarization of information from a variable length source sentence. The ability of attention to learn to focus on important regions within a context has made it a critical component in neural transduction models for several modalities" (Ramachandran et al, 2019).

With the introduction of *Transformer* models by Vaswani et al. (2017), attention based layers became the primary mechanism for representation learning, entirely replaced recurrence with self-attention. Self-attention is defined as attention applied to a single context instead of across multiple contexts (in other words, the query, keys and values, as defined later in this section, are all extracted from the same context). The ability of self-attention to directly model long-distance interactions and its parallelizability, which leverages the strengths of modern hardware, has led to state-of-the-art models for various tasks (Ramachandran et al, 2019).

In their work, Ramachandran et al, describes the stand-alone self-attention layer that can be used to replace spatial convolutions and build a fully attentional model as follows. Similar to a convolution, the first step of a stand-alone self-attention operation is to, given a pixel x_{ij}, extract a local neighborhood of pixels in positions $ab \in N_k(i, j)$ with spatial extent k centered around x_{ij}. Authors stress out that this form of local attention differs from prior work exploring attention in vision which have performed global (i.e., all-to-all) attention between all pixels.

In contrast to convolution operation, single-headed attention for computing the pixel output $y_{ij} \in \Re^{d_{out}}$, is then computed as follows (see Figure 3):

$$y_{ij} = \sum_{a,b \in N_k(i,j)} softmax_{ab} \left(q_{ij}^{\mathsf{T}} \, k_{ab}\right) v_{ab} \tag{2}$$

where pixel in position ij and its neighbors in positions ab are linearly transformed to obtain: *queries* $q_{ij} = W_Q x_{ij}$, *keys* $k_{ab} = W_K x_{ab}$ and *values* $v_{ab} = W_V x_{ab}$. $softmax_{ab}$ denotes a softmax applied to all logits computed in the neighborhood of ij. $W_Q, W_K, W_V \in \Re^{d_{out} \times d_{in}}$ are all learned transforms. From above description, it can be stated that local self-attention does not only aggregates spatial information over neighborhoods similar to convolutions (Equation 1) but also the aggregation is carried out with a convex combination of value vectors with mixing weights (softmax) parametrized by content interactions. In other words, spatial and channel-wise information is transferred to output by first calculating a weights matrix by content-based interactions between *querys* and *keys*. Next, a softmax non-linearity is applied to the weights matrix. Then, the output of softmax function and *values* are linearly combined to produce the output at pixel location ij. Above described processing pipeline is repeated for every input pixel ij.

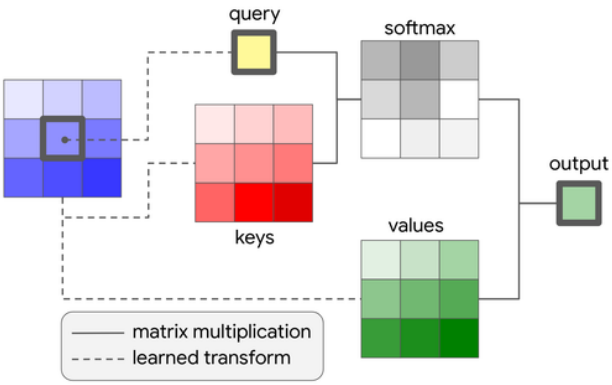

Figure 3: An example of a local attention layer over spatial extent of $k = 3$.

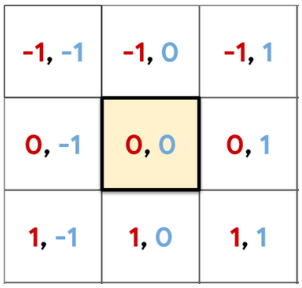

Figure 4: An example of relative distance computation. The relative distances are computed with respect to the position of the highlighted pixel. The format of distances is *row offset, column offset*.

Authors of the paper, "Stand-Alone Self-Attention in Vision Models" note that in practice, they make use of multiple attention *heads* which are used to learn multiple distinct representations of the input. It is applied by partitioning the pixel features x_{ij} in channel dimension into N groups $x_{ij}^n \in \Re^{d_{in}/N}$, computing single-headed attention on each head as described above with different transforms $W_Q^n, W_K^n, W_V^n \in \Re^{d_{out}/N \times d_{in}/N}$ per head, and finally concatenating the output representations into the final output $y_{ij} \in \Re^{d_{out}}$.

At this point, Ramachandran et al reiterates that as currently framed, no positional information is encoded in visual self-attention, which makes it permutation equivariant, limiting expressivity for vision tasks. In order to remedy this, the authors suggest using attention with 2D relative position embeddings, *relative attention*. Relative attention that Ramachandran et al proposes starts off by defining the relative distance of ij to each neighborhood position $ab \in N_k(i,j)$. The relative distance is factorized across dimensions, so each element in the neighborhood $ab \in N_k(i,j)$ receives two distances: row offset $a-i$ and column offset $b-j$ (see Figure 4). The row and column offsets are associated with an embedding r_{a-i} and r_{b-j} respectively each with dimension $\frac{1}{2}d_{out}$. The row and column offset embeddings are then concatenated to form $r_{a-i,b-j}$. This spatial-relative attention is now defined as:

$$y_{ij} = \sum_{a,b \in N_k(i,j)} softmax_{ab} \left(q_{ij}^\intercal \, k_{ab} + q_{ij}^\intercal \, r_{a-i,b-j}\right) v_{ab} \tag{3}$$

Notice that both row and column offsets can only take on integer values in the range $[-k/2, k/2]$ where k denotes the kernel size of the layer. Therefore, in practice, this self-attention layer learns 2 different offset embedding matrices, $R_{height}, R_{width} \in \Re^{(2(k-1)+1) \times \frac{1}{2}d_{out}}$ that hold all possible row and column offsets. Authors of the work claim that, with this form of relative attention, the logit measuring the similarity between the query and an element in $N_k(i,j)$ is modulated by the content of the element and the relative distance of the element from the query. They add that by infusing relative position information, proposed layer also gains translation equivariance, similar to convolutions.

Now that the background knowledge on convolution and visual self-attention operations are established, it is in order to describe how these layers inspired the architecture proposed in this work, the Expanding Visual Attention module, in image classification models.

3 Methodology

This section starts off with a detailed explanation of how the proposed Expanding Visual Attention layer is implemented for the experiments of this work, with the abbreviation EVA. Next, it introduces a thorough guideline on how the EVA module operates by touching on all of the operations involved and finally establishes a common ground for the upcoming Dataset & Experiments section.

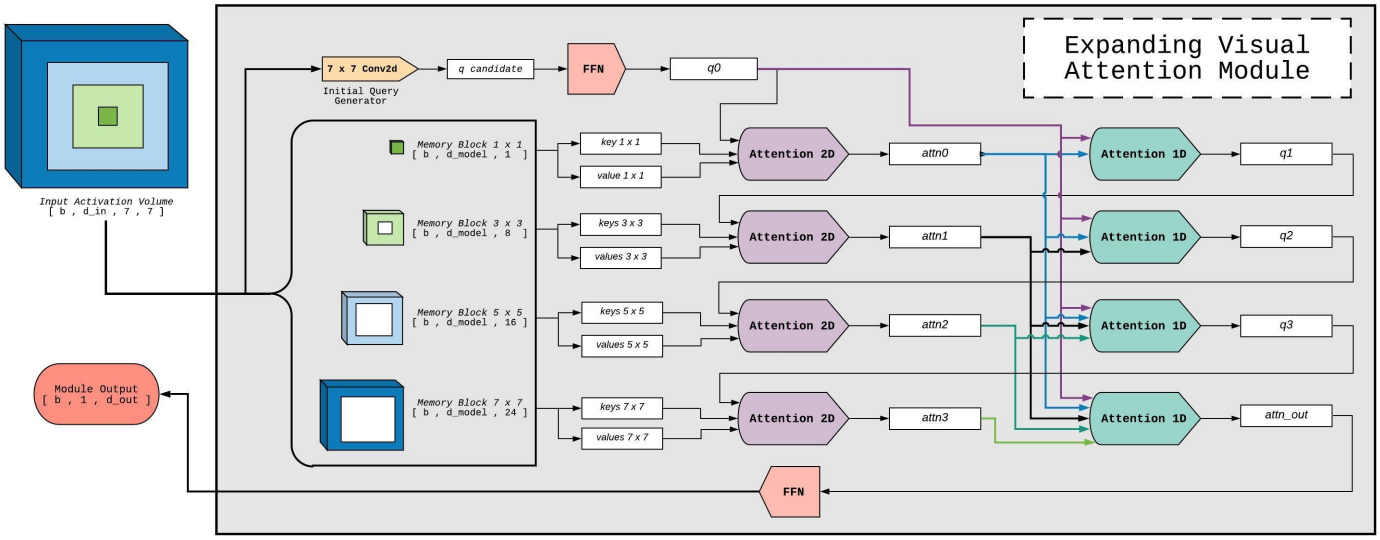

Figure 5: Computation graphs of plain convolutional and self-attentional versions of the *BasicBlock* and *Bottleneck* modules.

3.1 Implementing the Expanding Visual Attention Layer, EVA

The proposed Expanding Visual Attention (EVA) Module, whose computation graph is displayed above, is designed in order to effectively summarize spatially correlated input tensors with high spatial extent into 1 x 1 vectors. Version that is shared above, operates on input tensors of spatial extent 7 x 7 and produces an output tensor of 1 x 1. EVA module completes its processing schedule in 4 consecutive stages which are summarized below:

- ***1st Stage:*** During the first stage, the input tensor of shape $[batch\_size,\ input\_dims,\ 7,\ 7]$ is first fed through a Convolutional Layer of spatial extent 7×7 in order to generate the first query candidate of shape $[batch\_size,\ model\_dims,\ 1,\ 1]$. Next, freshly obtained query candidate is processed by a 2 layered Feed Forward Neural Network to obtain the first query named **q0**. Then, **q0** is used with the first memory block of shape 1×1 in a 2D visual attention module to generate the first attentional output labeled **attn0**. Latter 2D visual attention module therefore receives a single query vector, **q0**, a single key vector and a single value vector generated from the 1×1 memory block. The first stage is completed with a 1D attention module which receives **attn0** as the query vector and the concatenation of **q0** and **attn0** as the keys and values vectors. It should be noted that the aforementioned 1D attention module hence receives 1 query, 2 key and 2 value vectors. The output of the latter 1D attention module is named **q1** and has the shape $[batch\_size,\ model\_dims,\ 1,\ 1]$.
- ***2nd Stage:*** After the first stage is completed, the remaining 3 stages are almost identical in structure and follow similar computations. The output of the first stage, **q1**, is used as the query vector for the 2D visual attention module of the 2nd stage. In this stage, the key and value vectors for the 2D visual attention module are obtained from the second memory block of shape 3×3 that is also extracted from the original input tensor. Reader should bear in mind that, since the center pixel content of the original input tensor is previously used in the first stage as the 1×1 memory block, the second memory block of shape 3×3 is obtained by masking the center pixel and taking the remaining 8 pixels into account. Therefore, the 2D visual attention module of the 2nd stage receives a single query vector **q1**, 8 key and 8 value vectors that are generated from the 3×3 memory block. The output of the latter 2D visual attention module is named **attn1** and has the shape $[batch\_size,\ model\_dims,\ 1,\ 1]$. The second stage is also terminated with a 1D attention module which receives **attn1** as the query vector and the concatenation of **q0**, **attn0** and **attn1** as the keys and values vectors. Hence, the 1D attention module of the second stage

receives 1 query, 3 key and 3 value vectors. The output of the latter 1D attention module is named **q2** and has the shape $[batch\_size,\ model\_dims,\ 1,\ 1]$.

- ***3rd Stage:*** The third stage is constructed almost exactly like the second one except that the 2D visual attention module of the 3rd stage operates on a visual memory block with the spatial extent of 5×5. The output of the second stage, **q2**, is used as the query vector for the 2D visual attention module of the 3rd stage. In this stage, the key and value vectors for the 2D visual attention module are obtained from the third memory block of shape 5×5 that is also extracted from the original input tensor. As it was the case in the 2nd Stage, the center 3×3 block of the original input tensor is masked while obtaining the memory block of shape 5×5. Therefore, the 2D visual attention module of the 3rd stage receives a single query vector **q2**, 16 key and 16 value vectors that are generated from the masked 5×5 memory block. The output of the latter 2D visual attention module is named **attn2** and has the shape $[batch\_size,\ model\_dims,\ 1,\ 1]$. The third stage is also terminated with a 1D attention module which receives **attn2** as the query vector and the concatenation of **q0**, **attn0**, **attn1** and **attn2** as the keys and values vectors. Thus, the 1D attention module of the third stage receives 1 query, 4 key and 4 value vectors. The output of the third stage is named **q3** and has the shape $[batch\_size,\ model\_dims,\ 1,\ 1]$.

- ***4th Stage:*** The fourth and final stage is also similar to the ones described earlier. The output of the third stage, **q3**, is used as the query vector for the 2D visual attention module of the fourth and final stage. In this stage, the key and value vectors for the 2D visual attention module are obtained from the fourth memory block of shape 7×7 that is also extracted from the original input tensor. Again, the center 5×5 block is masked since it was processed during the earlier stages. Therefore, the 2D visual attention module of the final stage receives a single query vector **q3**, 24 key and 24 value vectors that are generated from the masked 7×7 memory block. The output of the latter 2D visual attention module is named attn3 and has the shape $[batch\_size,\ model\_dims,\ 1,\ 1]$. The fourth stage is also completed with a 1D attention module which receives **attn3** as the query vector and the concatenation of **q0**, **attn0**, **attn1**, **attn2** and **attn3** as the keys and values vectors. Hence, the 1D attention module of the final stage receives 1 query, 5 key and 5 value vectors. The output of the latter 1D attention module is named **attn_out** and has the shape [$[batch\_size,\ model\_dims,\ 1,\ 1]$. As the final step of the EVA module, the **attn_out** tensor is fed through a 2 layered Feed Forward Neural Network and the output is used as the final module output. This final output of the EVA module has the shape $[batch\_size,\ output\_dims,\ 1,\ 1]$

Above bullet points and Figure 5 summarize the basic working principles of the Expanding Visual Attention layer that is proposed in this work. Reader should note that all of the above described operations are implemented in PyTorch Deep Learning Framework for the experimenting purposes of this work. All coding implementations can be accessed in the open source repository via the following link:

TO BE UPDATED:!!!https://github.com/cankucuksozen/COMP541—Deep-Learning.

3.2 Intuition Behind the Expanding Visual Attention Module

Human beings can obtain visual information in parallel through the retina, but they cannot pay attention to all the information at the same time, Mizuhara et al. (2000). At any given time instant, the human visual system attends to a small group of points which lie at the center of our receptive field. Humans can make sense of the surrounding environment in vicinity to the point-like region however their ability to correctly recognize shapes and objects deteriorates as the distance to the point of attention increases. In a naive attempt to model this phenomenon, Expanding Visual Attention Module starts off by constructing a "rough and noisy" message about the class of the object with the help of a full sized convolutional layer and continues to reinvestigate the sample by first attending to the single center pixel content and recurrently updates its decision about the input patch by attending to pixels inside larger kernel sized windows growing around the center pixel. While doing so, the EVA module is designed to share information between the various kernel sized windows around the center of attention and recognize patterns and/or symmetries around a single focal point.

Remembering the discussion on shortcomings of convolutions placed in deeper layers of vision based networks, the EVA module is designed to effectively search the large feature space created by earlier convolutional layers. It tries to do so by partitioning its processing pipeline in hierarchical recurrent steps where it updates its notion about the input patch at every step. While EVA recurrently searches for meaningful connections inside the large input activation volume alongside the channel dimension, it also tries to learn spatial patterns that can be explained by symmetries evolving around a focal point. With this purpose, EVA divides its computation hierarchy and the spatial structure of the input activation volume into 4 stages and 4 differently sized frames. These frames occur as expanding around a center focal point, hence giving the name Expanding Visual Attention Module. During those 4 computation stages, EVA extracts aforementioned 4 differently sized and masked frames from the input patch at each computation step and projects keys and values from these masked frames. The query that is involved in every stage of EVA is constructed from the first convolutional layer as a "rough and noisy" message, and is asked to attend to different keys and values from differently sized frames at each computation stage to update its contents. Hence, in overall, EVA learns a convolutional layer to construct the first rough message query, 4 different 2D visual attention layers to investigate the input at different kernel sized windows and 4 different 1D multi-headed key-value attention layers to facilitate communication between the message at time step t with earlier findings at steps t-1, t-2 and etc.

With this unique architecture, the EVA is able to model 4 different categories for the downsampled pixel contents received from the input activation volume, thanks to 4 different query, key and value projections that are learned. If some particular combination of features are detected in the center 3×3 region, these features can be made sense by the 3×3 2D visual attention layer and the necessary message can be constructed by the corresponding 1D attention layer. Next, if the aforementioned combination of features in the center 3×3 region resembles a particular object category, then the 5×5 2D visual attention layer tries to look for more evidence in the 5×5 region if the contents of the input really matches that particular object category. This recurrent and spatially expanding processing scheme gives the EVA module better generalization and inference abilities compared to vanilla ResNets.

3.3 Creating an End-to-End Vision Model with EVA, EVANets

After the EVA layer is introduced as a *classifier module* in the previous subsections, now the question becomes how to construct a fully functional Image Classification architecture out of this decision making layer. This work *explores the straightforward strategy of attaching the proposed layer on top of an existing convolutional architecture*. To be specific, the experiments conducted in this work uses ResNet convolutional backbone to downsample the input CIFAR-10 image sample of size 32×32 pixels to 7×7 spatial extent and then feed the resulting downsampled feature map through the aforementioned EVA module.

In scope of this work, the schema explained in the previous paragraph is adapted using varying versions of the ResNet family of architectures originally proposed in He et al. (2015). Recall that, according to the original implementation of a ResNet by He et al. (2015), its core building pieces are the *BasicBlock* and the *Bottleneck* blocks. The *BasicBlock* modules are generally used in shallower networks such as the ResNet-20 and ResNet-38 whereas the *Bottleneck* blocks are made of use in deeper networks such as the ResNet-50, ResNet-101 etc. The BasicBlock of a ResNet contains two consecutive 3×3 spatial convolutions, followed by a residual connection between the input of the block and the output of the last convolution in the block.

Bear in mind that, since this work experiments on fairly small images of size 32×32 pixels supplied from CIFAR-10 dataset, it uses shallower configurations of the original ResNet architectures both as a backbone module for the proposed EVA layer and a Image Classification baseline. To be exact, this work makes use of ResNet-20 with $[16,\ 16,\ 32,\ 64]$ channel dimensions in its stem layer and following 3 stages respectively.

4 Experiments

As it has been reiterated previously, this work conducts its experiments on the CIFAR-10 dataset. It is an Image Classification dataset that is made up of 60000 32x32 pixel images containing instances

of 10 different object classes, Krizhevsky et al. (2008). This work, uses CIFAR-10 as the training dataset of the Image Classification task, recreates ResNet-20 as described in He et al. (2015) as the one of the baseline models, implements SelfAttnNet-20 as it is proposed by Ramachandran et al. (2019) in order to establish a comparable baseline between attentional vision models and proposes EVANet-20 as it has been developed throughout this work to conduct experiments on.

Following subsections share the obtained results from the experiments conducted on CIFAR-10 Image Classification task.

4.1 CIFAR-10 Image Classification

All of the experiments laid out in this subsection is executed on the CIFAR-10 dataset. The CIFAR-10 dataset, Krizhevsky et al. (2008), consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. All models tested in the following subsections, baseline ResNets, baseline SelfAttnNets and proposed EVANets are generated using the conventions laid out in earlier sections in Python/PyTorch.

In terms of architecture hyperparameter settings, dozens of varying configurations have been evaluated using a validation split (5k images) of the training set. According to the findings on the validation set, following conventions are adapted in all of the experiments listed from this point on:

- EVA – # of attention heads = 4
- Optimizer – Adam, Kingma & Ba (2014)
- Learning Rate – 0.001

In all 6 experiments covered in this subsection, the minibatch size is set to 150 and *Data Augmentation is used* as it was proposed by originally by He et al. (2015). Despite the training efficiency and computational demand of an attention based architecture is favorable to a traditional convolution, the resulting network is slower in wall-clock time (Ramacahandran et al, 2019). All models are trained using the same Adam Optimizer, Kingma & Ba (2014), with the same initial learning rate 0.001 and similar learning rate schedules.

Model	Architecture	Output Feature Map Size	EVA	Self-Attn.	Test Acc.(%)
ResNet-20 small	[2 , 2 , 2] x BasicBlock	[16, 32, 64]	X	X	91.25
SANet-20 small	[2 , 2 , 2] x BasicBlock	[16, 32, 64]	X	2xSA layer	92.45
EVANet-20 small	[2 , 2 , 2] x BasicBlock	[16, 32, 64]	1xEVA layer	X	**93.00**
ResNet-20 large	[2 , 2 , 2] x BasicBlock	[64, 128, 256]	X	X	94.53
SANet-20 large	[2 , 2 , 2] x BasicBlock	[64, 128, 256]	X	2xSA layer	94.88
EVANet-20 large	[2 , 2 , 2] x BasicBlock	[64, 128, 256]	1xEVA layer	X	**95.33**

Table 1: CIFAR-10 Image Classification results. Note that every SANet and EVANet architecture displayed in the table contains convolutional modules in first two stages and self-attentional and expanding visual attentional modules in the rest.

4.1.1 RESULTS

Obtained *top-1 Test Accuracy(%)* results for the aforementioned 6 experiments on CIFAR-10 Dataset is shared in Table 1. One can clearly observe that both of the proposed models of this work, EVANets, whether it be a *small* or *large* version, *outperformed the traditional baseline ResNets and attentional baseline SANets*. Since CIFAR-10 is a small dataset (only containing 50k training and 10k test images), models tend to quickly memorize the training samples and diminished their capacity to generalize to unseen samples. The essential conclusion that can be drawn from Table 1 is that proposed EVANet models, constructed out of Expanding Visual Attention layers, are more than capable compared their baseline convolutional and attentional counterpats, ResNets and SANets, when it comes to the Image Classification task.

5 CONCLUSION AND DISCUSSION

Altough Convolutional Neural Networks are considered to be the core building module of almost any Computer Vision related Neural Network architecture nowadays, this work proved that Attentional models that aim to capture *long-range dependencies* within a input activation volume and can learn to focus on important regions within a context can indeed serve as *decisive end layers* in vision based deep learning applications. This work started off as a naive implementation of human visual system, which can cannot attend to all the information at the same time, Mizuhara et al. (2000), but at any given time instant it can only attend to a small group of points which lie at the center of its receptive field. The deep learning methodology that is chosen to model the aforementioned phenomenon was the visual self-attention technique that was first proposed by Ramachandran et al. (2019) in their paper "Stand-Alone Self-Attention in Vision Models". After a detailed discussion about the background information and methodology, this work developed, described, implemented and tested a novel visual attentional layer, the Expanding Visual Attention Module and showed that the attentional architectures can be easily modified, enhanced and used in vision tasks. Although the number of parameters and the number of FLOPs needed by a visual attentional layers are less than a traditional convolutional layer, resulting wall-clock time of training these models is much higher. Ramachandran et al. (2019) explain that the reason for this discrepancy is the lack of optimized kernels available on various hardware accelerators. They also note that, in principle, depending on the degree to which the field deems that attention provides a viable path, it may be possible to significantly speed up the wall-clock time for training and inference accordingly. Among other details, this work carefully explained the baseline models, limitations and experimental settings as promised. As Future Work, first order of business is to plan to start working on new forms of attentional layers and possibly test their capabilties in harder tasks such as Scene Graph Generation or Visual Question Answering.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 2008. URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

Hiroaki Mizuhara, Jing-Long Wu, and Yoshikazu Nishikawa. The degree of human visual attention in the visual search. *Artificial Life and Robotics*, 4(2):57–61, June 2000. doi: 10.1007/bf02480857. URL `https://doi.org/10.1007/bf02480857`.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.