Exploring Self-attention for Image Recognition

Abhinava Arasada - f20170028@pilani.bits-pilani.ac.in

Ayush Singh - f20170652@pilani.bits-pilani.ac.in

Abdul Kadir Khimani - f20170696@pilani.bits-pilani.ac.in

29/09/2020

Abstract

The paper we selected explores the usage of self-attention for image recognition models, something which has traditionally been put to use in natural language processing. The effectiveness of using two types of self attention as a base for image recognition has been assessed primarily. One is the pairwise self-attention, which generalizes the standard dot-product attention (which we have also described in this introduction) and is essentially a set operator. The other is patchwise self-attention. The performance of self-attention networks has been shown to be same or much higher than their convolutional counterparts, with the patchwise models specifically significantly outperforming the convolutional baselines.

1 Introduction

Artificial Intelligence, in the context of the 21st Century, has been fundamental in bridging the gap between humans and machines' abilities. A tremendous amount of research work has happened in related fields over the past few years. Computer Vision is one of the areas that has seen a significant improvement with increased computing power and massive data availability. Convolutional networks have tremendously improved results in the field

of Computer Vision.[7] In the late 90s, they were successfully applied for handwritten digit recognition. Building on this novel advancement, CNNs were scaled up in 2012, achieving state-of-the-art accuracy using the ImageNet dataset, beating all previously known methods by a significant margin, thus propelling the deep learning era in computer vision.

A Convolutional Neural Network, often termed as CNN or ConvNet, is a Deep Learning algorithm that can take in an input image, assign significance (learnable weights and biases) to aspects/pixels in the image, and can differentiate one from the other. The preprocessing and feature engineering required is lower compared to other algorithms. In classical methods, filters had to be chosen manually. However, with sufficient training, data, and computation power, ConvNets can 'learn' these filters independently. Although the convolution operator has served as the primary operator in modern image recognition, it has its drawbacks. For example, the convolution operator is rotation invariant. The number of parameters the model has to learn grows with the size and dimension of kernel k. CNNs have poor scaling abilities thus making it harder to capture long-range interactions for convolutions. This interaction problem has been overcome in sequence modelling through the use of attention.

Recent work has discovered that self-attention may constitute a viable alternative for building image recognition models. The self-attention operator has been very useful in natural language processing. It serves as the basis for beautiful architectures that have replaced recurrent and convolutional models across various tasks. The development of effective self-attention architectures in computer vision holds the hope of discovering models with different and perhaps similar properties to convolutional networks.

2 Related work

2.1 Convolutional Neural Networks

Before we dive head first into attention models, it is necessary to have some background in Convolutional Networks. We shall look at CNNs in brief and then proceed futher.

An image can be represented as a matrix of pixel values. One way is to flatten the image (e.g., 10x10 image matrix into a 100x1 vector) and feed it to a Multi-Level Feed Forward Network for classification purposes. This process is not without its caveats. This method gives an average precision score for class prediction on raw binary images. However, the performance is very low for complex images having pixel dependencies throughout.

On the other hand, CNN can successfully capture the Spatial and Temporal dependencies in an image using filters(kernels). Also, the parameters needed to train the model are reduced. Weights can also be reused. It is also possible to train a very deep network(e.g., ResNet). In a nutshell, the network can be made to understand the sophistication of the image better. Processing images of large dimensions can get computationally intensive. The role of the ConvNet is to tone the images down to a level so that they easier to process without losing critical features necessary for a good prediction. This is significant when designing a model architecture that is good at learning features and is scalable to massive datasets.

All ConvNets are based on the discrete convolution operator. The discrete convolution operator * is defined as follows:

$$X(F * k)(p) = \sum_{s+t=p} F(s)k(t) \tag{1}$$

Here F is a discrete function; k is a discrete filter.

An essential characteristic of the convolution is its translation invariance: the same filter k is applied across the image F. Regarding a convolutional neural network, convolution is a linear operation that performs the multiplication of weights with the input, similar to a

neural network. Since convolutions are applied to two-dimensional input, the multiplication is carried out between an array of input and a two-dimensional array of weights, called a filter or a kernel.The filter is taken to be smaller than the input data, and the multiplication between a filter-sized window of the input and the filter is a dot product. A dot product is an element-wise multiplication between the patch of the input(same as filter size) and filter, which is then added, thus yielding a single scalar value. The operation is thus referred to as the "scalar product ".

The filter is smaller than the input size because it allows the same filter (set of weights) to be multiplied by the input image multiple times at different locations on the input. Specifically, the filter is applied to each overlapping part or filter-sized window of the input data, and the window is moved each time; this metric is called 'stride'. The filter progresses to the right with a fixed stride until it goes through the complete width. Moving on, it comes down to the beginning of the image with the same stride and repeats the same process until the entire image is captured. For images with multiple channels, the filter has the same depth as that of the input image. Matrix Multiplication is performed between each window of the stack, and all the results are summed with the bias to give us a flat one-depth channel output. For the detection of a particular type of feature in the input image, the application of that filter in this fashion across the entire input image enables the filter to discover that features anywhere in the image. This property is called translation invariance, e.g., whether the feature is present instead of where.

The Convolution Operation extracts high-level features such as edges, from the input image. Multiple Convolutional Layers can be applied, thus forming a deeper network. The first layer captures low-level features such as edges, color, gradient orientation, etc. With added layers, the architecture understands high-level features specific to our task, thus giving outstanding results. Convolution operation results in two types of output — one in which the convolved feature gets reduced in dimensionality compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by Valid

Padding in the case of the former, or Same Padding in the latter's case. Padding is generally performed by appending zeroes to the image on all sides.

The other building block is the Pooling layer, responsible for reducing the size of the Convolved Feature. Pooling helps to decrease the computational power required to process the data by applying dimensionality reduction. It is also helpful for extracting dominant features that are rotational and positional invariant, thus aiding our model's effective training. There exist two types of Pooling: Max Pooling and Average Pooling. Max Pooling gives the maximum value from the portion of the image parsed by the Kernel. Average Pooling yields the average of all the values from the portion of the image parsed by the Kernel.

Max Pooling also functions as a Noise Suppressant. It performs de-noising, other than dimensionality reduction. On the other hand, Average Pooling carries out dimensionality reduction as a noise-suppressing mechanism. Hence, Max Pooling is generally used more than Average Pooling. The Convolutional Layer and the Pooling Layer, together form the n-th layer of a Convolutional Neural Network. As per the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but this increases computational cost.

Our final layers depend on the specific task at hand. Once the features are learned through convolution, the final layers can be fully connected layers for the classification task, or a 2D output, for the segmentation task.CNNs have become state of the art with the availability of specific open-sourced architectures: LeNet, AlexNet, VGGNet, GoogLeNet, ResNet[3] , ZFNet[11]

2.2 R-CNN

This paper employs a novel approach to the technique of self-attention to build an image recognition model. Hence, before we describe the methods used within the paper itself, and contrast it with the conventionally used convolutional neural networks, let us briefly look into the concept of attention followed by self-attention. To explain what attention is,

let us take an example of a translator that converts a sentence in one language to another language. Internally, this model would be a decoder that takes the sequence of words as input one by one and, after consuming the entire input, would produce a context that would be passed on to an encoder, which would then produce the output sentence. Hence one word is consumed by the decoder at each step, following which it produces a context, after which one word is produced by the encoder in each step. Also, both the encoder and decoder tend to be RNNs (recurrent neural networks) such that the output of a certain step depends on both the input passed within that step as well as the previous step's output. Hence we can think of output at each step as a sort of state representation such that the current state of the decoder/encoder determines the output for a given input (a single word in this case) in the next step. Additionally note that the final state output of the decoder is what is passed to the encoder as input, other states are (as of now) hidden i.e they are not passed from decoder to the encoder.

2.3 Attention

In such models, people soon noticed that the context passed from the decoder to the encoder turned out to be the limiting factor, as only a limited amount of information could be passed. This limitation was very apparent when trying to translate longer sentences. Hence, by necessity, this lead to the development of attention[6]. Attention allows us to, as the name suggests, pay more attention to important parts of the input. In the current example, obviously given a sentence in any language, it is intuitive that certain words/phrases would carry more information than other parts of the sentence. Attention allows us to highlight and focus upon such chunks of the input while flattening the amount of importance allocated to stop words and the like which rarely contain a lot of information. In the ongoing example of a translator, we would have to incorporate two major changes to enable attention. Firstly, the context passed from the decoder to the encoder no longer consists of only the final state, instead all the intermediate states are now passed from the decoder to the

encoder. The other change is that scores are assigned to each of the states, and these scores are multiplied with the state variables which in turn determines the importance/relevance of each state. This score calculation is done at each step.

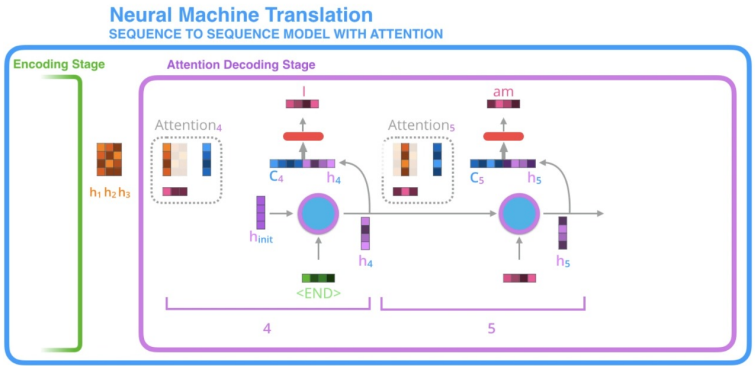

Figure 1: Attention

2.4 Self attention

Now, where does self - attention come into the picture? In the context of translation consider the sentence, “Abdul thinks he is very tall since Ayush is shorter than him.“. In this sentence is the term “he” stand alone, i.e does it carry any information by itself, or is it associated with/ refer to some other term ? Similarly, who does “him” refer to ? How would we encode this information that the terms “he” and “him” refer to, or have a very strong connection with, people mentioned previously ? Self attention allows us to associate the pronouns “he” and “him” with Abdul. Similar to attention, self attention allows us to accentuate the important parts of the sentence and drown out the less important parts, however one difference is that this process now involves the interaction of different parts of the input with each other. In order to do this, we require three vectors for each input[2][1]. These are called the key, value and query vector. We store a set of weight vectors (randomly initialized) for each of key, value and query such that each input (which in this case, is a vectorized representation of each word in some form such as an embedding) is simply multiplied with these weights in order to generate the actual key, value and query vector for each input. Now that we have the required vectors for each input term, we must obtain attention values for each of the

terms when compared to a single word. Let us suppose that in the aforementioned example we are now calculating the attention scores of each term of the sentence with respect to the term “Abdul”. In order to do this, we take the query vector of “Abdul” and find the product of this vector with all of the terms key vectors. This result is then normalized (by using softmax, for example). Now the values are recalculated for each term by multiplying it with the attention result obtained. This is referred to as the weighted value. Now all you must do is simply add all of these weighted values together. This gives us the self-attention output for the word “Abdul” when compared to all the terms in the input, including “Abdul” itself. It is very intuitive to see that an important term that has a higher attention score would give larger values in the output, regardless of the actual values of each value vector. Additionally, those terms which would have the highest attention scores with important terms i.e., terms with high values in the value vector, would end up having the highest score. While the previous example has been from the point of view of language processing because that is much easier to visualize, its application is mathematically as well as conceptually almost identical in the field of image recognition where input tokens arent words or phrases but rather pixels or segments of grayscale images.

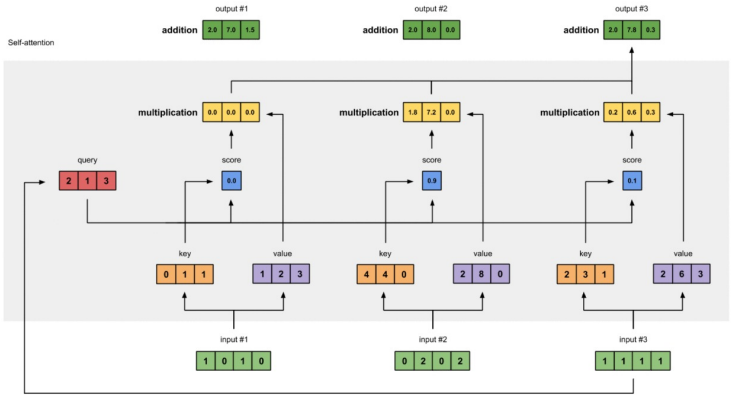

Figure 2: Self attention

2.5 Previous work on self - attention

Self attention as a concept is in a very nascent stage, and is undergoing rapid advancements in image recognition and related fields day by day. However, due to performance issues and other constraints self attention concepts were only used in a supporting role to i.e in combination with generic convolution neural networks. Only recently have people experimented with combining convolutional and self-attention processing streams, but the global self-attention they used was not sufficiently powerful to replace convolutions entirely. Additionally related work was also done in the form of dynamic filter networks[5], where filters do not stay constant after training as is the case conventionally, but rather adapt to the input.However this too was very intensive in terms of memory and computational costs especially in the case of high definition images and large datasets.

However, as was recently discovered, restricting the scope of self attention to a small neighbourhood[4] instead of applying it globally, has resulted in a sharp fall in the amount of computing power as well as memory required. This breakthrough enables this paper among others[9] where self attention is used throughout the network.

3 Ongoing work - What this paper is about

For almost two decades, convolutional networks have been dominant in the image recognition field of Machine Learning. There have been several attempts to incorporate self-attention ideas along with the convolutional models but, all attempts use self-attention as a 'supporter' module for the 'main' convolutional network. In this paper, the aim is to use only self-attention for image recognition, specifically in the form of two new types of self-attention operators.

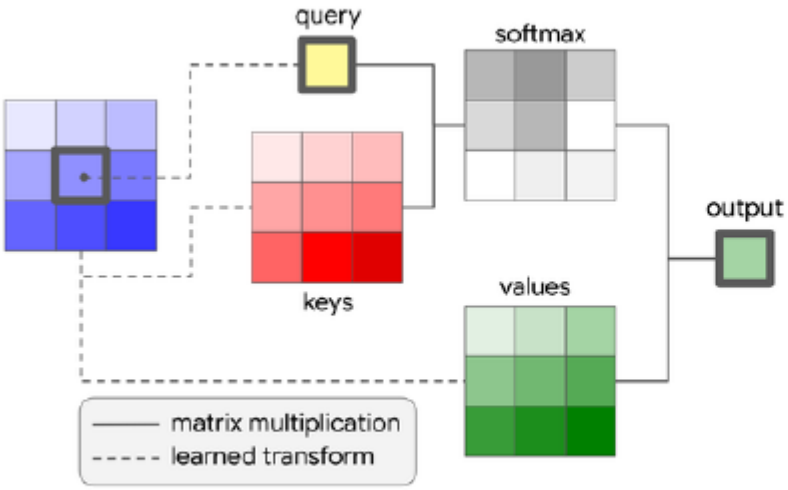

Figure 3: An example of a local attention layer over spatial extent of $k = 3$.

-1, -1	-1, 0	-1, 1	-1, 2
0, -1	0, 0	0, 1	0, 2
1, -1	1, 0	1, 1	1, 2
2, -1	2, 0	2, 1	2, 2

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*.

Figure 3: Self atention illustrated

3.1 Pairwise self attention

A generalized version of the standard dot product attention often seen in natural language processing. The convolution operator is a sequence operator, in comparison to that, pairwise attention is a set operator. Therefore, it does not add external parameters (which will have to be trained) making it permutation and cardinality invariant. This allows us to change the patch size without any change in the amount of parameters. Compared to the convolution operator, the advantage is remarkable.

3.2 Patchwise self attention

Like convolution, the patchwise self-attention operator can be used to identify specific patterns, but they are much more powerful than convolution. Unlike the pairwise operator, the patchwise operator is not a set operator neither is it permutation or cardinality invariant.

Basic form,

$$\mathbf{y}_i = \sum_{j \in \mathcal{R}(i)} \alpha(\mathbf{x}_i, \mathbf{x}_j) \odot \beta(\mathbf{x}_j),$$

Where alpha can be decomposed as follows

$$\alpha(\mathbf{x}_i, \mathbf{x}_j) = \gamma(\delta(\mathbf{x}_i, \mathbf{x}_j)).$$

And delta is the relation function

Summation: $\delta(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) + \psi(\mathbf{x}_j)$

Figure 4: Pairwise self attention

Patchwise Self-attention

Basic form,

$$\mathbf{y}_i = \sum_{j \in \mathcal{R}(i)} \alpha(\mathbf{x}_{\mathcal{R}(i)})_j \odot \beta(\mathbf{x}_j),$$

Where alpha can be decomposed as follows

$$\alpha(\mathbf{x}_{\mathcal{R}(i)}) = \gamma(\delta(\mathbf{x}_{\mathcal{R}(i)})).$$

And delta is the relation function

Concatenation: $\delta(\mathbf{x}_{\mathcal{R}(i)}) = [\varphi(\mathbf{x}_i), [\psi(\mathbf{x}_j)]_{\forall j \in \mathcal{R}(i)}]$

Figure 5: Patchwise self attention

4 Implementation

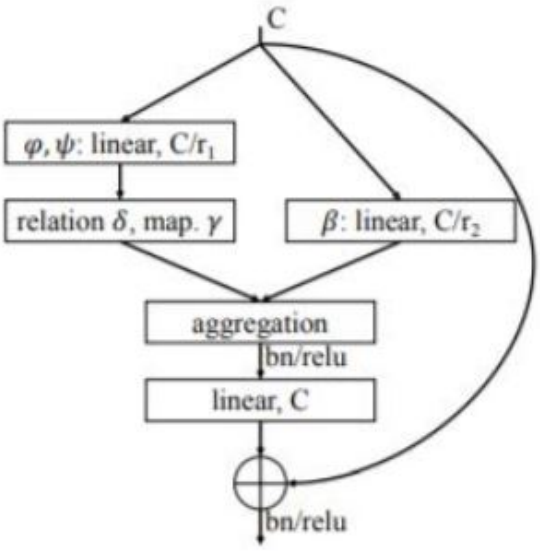

Figure 6: Histogram equalization

4.1 Code and Data

The implementation of this paper can be found at https://github.com/hszhao/SAN[12], which is the official github repository of the creators. The data on which this model was trained and tested is the ImageNet dataset (ILSVRC2012)[10].

4.2 Installation

The code has been tested with 8x Quadro RTX 6000, with PyTorch 1.4.0, Python3.7, CUDA 10.1, CuPy 10.1, tensorboardX. We cloned the github repository on Google Colab.

4.3 Challenges

ImageNet Dataset is not publicly available. We had to create an account through BITS mail to access it. The dataset is too huge to carry out training on our local PCs. The dataset

has a total of 1,281,167 images for training set. There are 50,000 validation images, with 50 images per synset. There are 100,000 test image, all in JPEG format. Training set is of 138 GBs, Validation set of 6GBs and Testing Set of 16GBs. It was not possible for us to do training due to the lack of resources. So instead, we used the validation set and pretrained weights(available in the repository) for cross checking the working of the model. We tried using the local PC for this, installed CUDA as well after lots of issues but downloading 16GBs wasn't possible due to internet issues. We then used Google Colab for this purpose. The validation set was downloaded and preprocessed using an online script. We had to create 1000 folders, each folder contains the same class images.

4.4 Observations

We were able to match the results given in the paper exactly after setting up the config files.

5 Innovation

As per the instructions given by Professor, we decided to train this model from scratch on some different dataset. We downloaded the Intel Image Classification Dataset from Kaggle[8] Website which was used to host this competition. Intel Data Consists of Natural Scenes around the world. This Data contains around 25k images of size 150x150 distributed under 6 categories : buildings, forest, glacier, mountain, sea, street. The Train, Test and Prediction data is separated in each zip files. There are around 14k images in Train, 3k in Test and 7k in Prediction. This image data was initially published on `https://datahack.analyticsvidhya.com` by Intel to host a Image classification Challenge.

The model was trained on Google Colab. We used a batch size of 16, 32, 64(best) and a learning rate of 0.0001. We werent able to go beyond the batch size of 64 due to GPU memory limitations. The model was trained for 45 epochs with a training accuracy of 88% and test accuracy of 87% using the SAN15 patchwise network. The model accuracy could

further be improved by trying an array of hyperparameters which we couldn't experiment much because of memory and time constraints. We also tried changing the intermediate layer to Subtraction but that did not yeild comparable results.

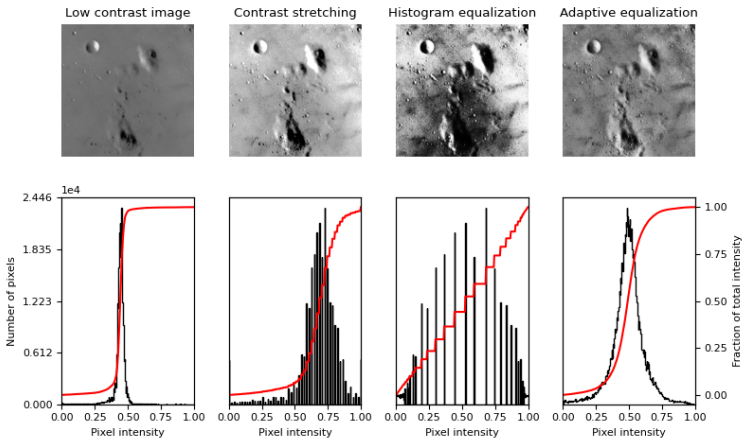

Figure 7: Patchwise self attention

References

[1] Jay Alammar. Illustrated: Self-Attention. `https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a`.

[2] Jay Alammar. The Illustrated Transformer. `http://jalammar.github.io/illustrated-transformer/`.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015. cite arxiv:1512.03385Comment: Tech report.

[4] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition, 2019.

[5] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors,

Advances in Neural Information Processing Systems 29, pages 667–675. Curran Associates, Inc., 2016.

[6] Raimi Karim. Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention). https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

[7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[8] Puneet. Intel image classification. https://www.kaggle.com/puneet6060/intel-image-classification, 2019.

[9] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 68–80. Curran Associates, Inc., 2019.

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[11] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 2019.

[12] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020.