Simplified Replication of Attention Augmented Convolutional Networks

Matteo Demartis
demartis@kth.se

Flavia García Vázquez
flaviagv@kth.se

Fernando García Sanz
fegs@kth.se

Abstract

In this project we are revisiting the publication Attention Augmented Convolutional Networks [1] with the purpose of studying and replicating the main topics and features discussed in the article on CIFAR-10 [2] dataset. In order to do so, we have implemented both a ResNet-34 [3] and a Wide-ResNet-28-10 [4] architectures, as well as data augmentation and self-attention mechanisms. Experiments have shown that both networks perform very good for our image classification task, but the Wide-ResNet-28-10 outperformed the ResNet-34 by 2.11%. Data augmentation gave a positive result, improving the accuracy of the Wide-ResNet and the ResNet by $\sim 2\%$ and $\sim 5\%$, respectively. Due to memory constraints, self-attention was applied to a few number of layers with lower dimensionality. The accuracy of the attention augmented networks was not better than the already high accuracy of the non augmented networks: 89.10% in the Wide-ResNet-28-10 and 86.99% in the ResNet-34. Comparing these results with the reference paper, we can conclude that in order to have an improvement by augmenting convolutional layers with self-attention, we should augment more convolutional layers. To be able do this we should have more than one GPU with the proper memory amount.

1 Introduction

Since Convolutional Neural Networks (CNNs) were invented, these have been the most popular algorithm for computer vision tasks, mainly in image classification [5]. The convolutional layer is designed so that the main features considered by the algorithm are locality and translation equivariance. The locality feature represents the main weakness of the convolutional networks, making the network miss global information of the image.

On the other hand, Google Brain published in 2017 a paper [6] that used self-attention for machine translation, which provides state-of-the-art results. Self-attention is an algorithm that captures long range interactions of the input. This algorithm has been mainly applied to sequential input data, as text, for example.

The main idea we tested is if the accuracy of a convolutional network can be improved by augmenting convolutional layers with self-attention. This augmentation consists of concatenating the feature maps of the convolution and the self-attention. The Attention Augmented Convolutional Neural Network (AA-CNN) will identify local and non-local interactions and will preserve translation equivariance. For the sake of performing these experiments, the TensorFlow framework in version 2.0 has been used[1].

We test the different methods on the CIFAR-10 dataset by augmenting a ResNet-34 and a Wide-ResNet-28-10. Both networks returned good test accuracy when classifying CIFAR-10 data, 86.99% and 89.10%, respectively. The obtained results have proven that the applied data augmentation is an effective mechanism to enhance the performance of the model classifying unseen data. Because

[1]The project code can be found in https://github.com/fernando2393/DD2424_Final_Project

of resources limitation, we applied self-attention to less convolution layers than the reference paper. This resulted in a non-improvement of the performance of the network. The results which have led to these conclusions can be found in Section 5.

2 Related Work

It exists a lot of work related with CNNs, focused on improving the classification and object detection accuracy on benchmark datasets (CIFAR-10, ImageNet [7] and COCO [8]). The main goal is to find the most adequate network architecture. State-of-the-art networks have shown that using convolutional layers with skipping connections works better than other architectures [9] [10] [11].

Attention has found an important place into the algorithms that model sequences because of its ability to capture long distance interactions. It was first introduced by Bahdanau et al. [12] that combined attention with a Recurrent Neural Network [13] for alignment in Machine Translation. Attention was further extended by Vaswani et al. [14], where the self-attentional Transformer architecture achieved state-of-the-art results in Machine Translation. After this success, many researchers started to focus on improving convolutional layers with self-attention. This became a hot topic in different fields of machine learning, as Language Processing [15], Reinforcement Learning [16] and visual tasks [17] [18]. From this last category, Google Brain showed that attention augmented convolutions get better results than other combinations of attention and convolution, by proposing an algorithm that produces additional feature maps, rather than recalibrating convolutional features. Our work has a direct relation with the previously mentioned Google Brain work.

3 Data

The dataset used was CIFAR-10, a standard benchmark for low resolution imagery (32x32 RGB images). Six random samples from this dataset are shown in Figure 1 . For training, validation, and testing tasks, we divided the dataset in three subsets with 45,000, 5,000 and 10,000 samples each. This data was mean-normalized by extracting the training mean and dividing it by 255, maximum numerical value for RGB. Furthermore, data augmentation was employed by applying a horizontal flipping with a 0.5 probability, a zero-padding to have images of 40x40 pixels and a random cropping of 32x32 pixels. The results obtained with and without data augmentation are shown in Table 1.

As CIFAR-10 is a benchmark dataset, many researchers have focused on improving its classification accuracy by designing new algorithms. Currently, the state-of-the-art performing methods on this dataset obtain an accuracy around 99%. The best algorithm is BiT-L [19] that achieved an accuracy of 99.3%. This algorithm focuses on using transfer learning for visual tasks, which revisits the paradigm of pre-training on large supervised datasets and fine-tuning the model on a target task.

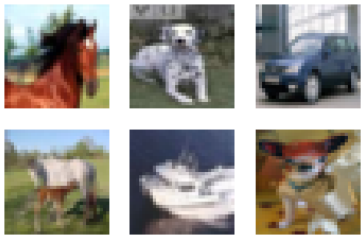

Figure 1: Random samples from CIFAR-10.

4 Methods

As mentioned before, attention is originally used to capture long-distance interaction in sequences. The attention augmented network presented in *Attention Augmented Convolutional Networks* employs self-attention throughout the network in order to augment convolutional layers. Moreover, the attention module employed is modified so that it can be adapted to images. The resulting module is called attention augmented convolution, being able to also capture global contexts in pictures. This

attention augmented module is what is added by us to both the ResNet-34 and the Wide-ResNet-28-10. The outcomes we obtain will be compared with the results of the not augmented architecture, with the purpose of determining if this mechanism actually enhances the performance of these networks and, therefore, is worth it.

4.1 Convolutional network architectures

As mentioned before, two different convolutional neural network architectures have been implemented: ResNet-34 and Wide-ResNet-28-10.

4.1.1 ResNet-34

The ResNet-34 is a network composed by 34 layers. The new main idea behind the ResNet-34 architecture is the use of a residual block. The residual block is composed of two connected layers and a skipping connection between the input and the output. The connection skipping is used to avoid the vanishing gradient problem. The following figure shows the structure of the residual block.

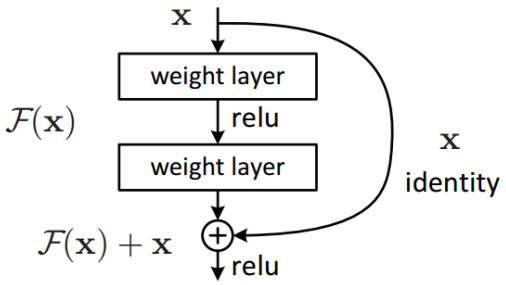

Figure 2: ResNet residual block.

The ResNet-34 consists of four consecutive main stacks of 3, 4, 6, 3 residual blocks respectively. In each stack the filter size is doubled each time, starting from 64. All the layers inside the stacks use a kernel of size 3x3 and at the beginning of each stack the input is down-sampled by a factor of two. At the beginning of the network a convolution using a 7x7 kernel followed by a max-pooling operation with pool size 3x3 is used. Moreover, right after the last stack an average-pooling operation is used.

4.1.2 Wide-ResNet-28-10

The Wide-ResNet-28-10 is a variation in the implementation of a common ResNet. Its difference lies in the width of the layers which compose it. In a nutshell, by increasing the width of the layers, more parameters can be trained at the same time, increasing the learning rate. This makes the network shallower, speeds up the training process, and reduces the errors due to gradient propagation through the layers. This type of network has proven to be quite efficient for this kind of computations, converging to good results quite fast compared to other explored methods.

4.2 Multihead-attention (MHA) over images

MHA is defined by the number of *heads* (N_h), the depth of *values* (d_v) and the depth of *keys* and *queries* (d_k). N_h divides d_v and d_k evenly and denote d_h^v and d_h^k the depth of *values* and *queries/keys* per attention head. Each attention head is a self-attention block, each one capturing different spatial and feature subspaces.

Self-attention blocks are formed by *keys* (K), *queries* (Q) and *values*(V). In order to get these matrices a linear point-wise convolution is applied to the input (X) of shape (H,W,3). The learned linear transformations that map the input with each element are $W_q, W_k \in \mathbb{R}^{HW \mathrm{x} d_k^h}$ and $W_v \in \mathbb{R}^{HW \mathrm{x} d_v^h}$. Figure 3 represents the operations done by a self-attention block in order to compute its output. These operations are summarized in the following formula that shows how to calculate the output of the self-attention mechanism for a single head h:

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

In order to get the MHA output, all heads are concatenated and projected again (point-wise convolution) as follows:

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

where $W_0 \in \mathbb{R}^{d_v \text{x} d_v}$ is a learned linear transformation. MHA(X) is then reshaped into a tensor of shape (H,W,d_v) to match the original spatial dimensions.

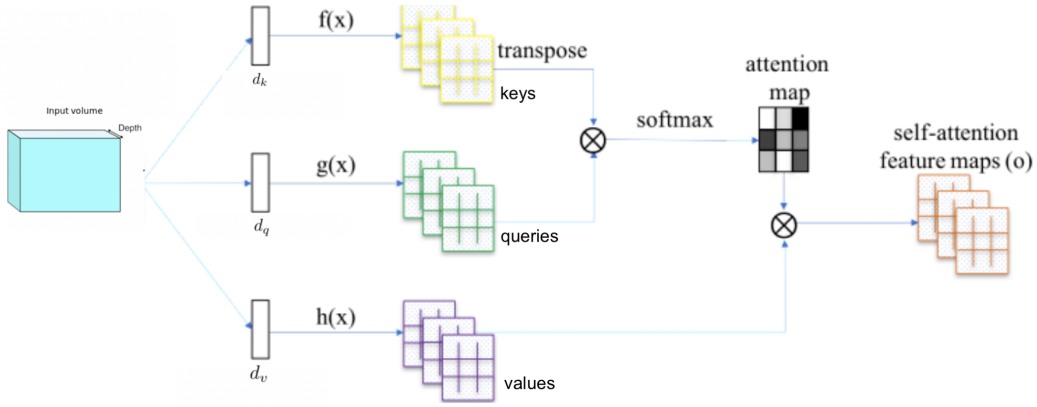

Figure 3: Self-attention block diagram.

4.3 Attention Augmented Convolution

The attention augmented convolution is obtained by concatenating the convolution with the multi-head attention as follows:

$$\text{AAConv}(X) = \text{Concat}[\text{Conv}(X), \text{MHA}(X)]$$

The resulting attention augmented convolution has the properties of being equivariant to translation and being able to operate on different spatial dimensions.

It is important that every augmented convolution layer is followed by a batch normalization [20] layer. This is done to scale the contribution of the convolution feature maps and the attention feature maps.

The augmented convolution block would be added just to some parts of the network since the memory cost $O((N_h(HW)^2)$ can be prohibitive for large spatial dimensions. Therefore, we will add attention to the smallest spatial dimensions until we hit memory constraints (see Section 5 for more details). To reduce memory problems it is also advised to reduce the batch size.

4.4 Relative Positional Embeddings

As explained in [21], relative self-attention enables translation equivariance preserving permutation equivariance. In the main paper, to better fit images, it is implemented a two-dimensional relative self-attention that adds relative height and width information. With this modification the output of a certain head h is now:

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

where S_H^{rel} is the matrix of relative position logits along the height and S_W^{rel} is the matrix of relative position logits along the width.

5 Experiments

As aforementioned, two complex convolutional networks architectures have been employed: ResNet-34 and Wide-ResNet-28-10. The first one is composed by 34 different layers, while the second one is composed by 28, but applying a widen factor of 10, which increases the width of the blocks that compose the network by 10.

These networks have been employed first with their originally proposed configuration over the original data and the augmented data (Section 5.1), and then their architecture has been modified in order to apply the self-attention mechanism (Section 5.2).

The loss function applied to both networks is categorical cross-entropy, which seems to be the best choice since the purpose of the network is classifying a picture among a total of 10 different categories.

As suggested in the main paper, we initially trained the ResNet-34 using stochastic gradient descent with momentum 0.9 and using a linearly increasing learning rate from 0 to 0.2B/256 (where B is the batch size) for the first 5% of the epochs and then annealed with cosine decay for a total of 500 epochs. After several experiments, we found out that Adam optimizer with a step increasing learning rate outperformed the version of stochastic gradient descent presented in the main paper by increasing the accuracy of about 10%. For this reason, we decided to use Adam optimizer instead.

The original configuration of the Wide-ResNet-28-10 employs stochastic gradient descent as optimization method, together with a Nesterov momentum of 0.9. It also makes use of a custom learning rate policy in which its the value decreases according to the current epoch of training. We have decided to keep this configuration since when we tried to replicate the one used in the ResNet-34 the results we obtained were worse.

For the sake of implementing these modifications, it has been necessary to develop a custom mechanism for training which varies from the default one implemented by Keras, TensorFlow. The data need to be specifically manipulated during training to be able to handle the developed data augmentation mechanism.

5.1 Data augmentation ablation study

In order to analyse the effect of data augmentation, we do an ablation study over the two networks aforementioned. In Section 3 we give a more detailed description of how the augmentation was performed. The ResNet-34 was trained for 220 epochs, while the Wide-ResNet-28-10 was trained for only sixty since this kind of network converges in quite less epochs. As it can be seen in the following table (Table 1), the results of applying data augmentation improved the accuracy obtained in validation and testing. Therefore, all the experiments performed Section 5.2 in were executed together with data augmentation in order to maximize the accuracy obtained in the results.

Architecture	Data Augmentation	Train Acc.	Val. Acc.	Test Acc.
ResNet-34	No	95.87%	81.74%	81.23%
ResNet-34	Yes	93.37%	87.55%	86.99%
Wide-ResNet-28-10	No	98.26%	87.82%	87.26%
Wide-ResNet-28-10	Yes	94.99%	89.74%	89.10%

Table 1: Performance of different network architectures with and without data augmentation.

5.2 Self-attention ablation study

In order to prove the enhancement attention augmentation mechanisms produce in the training process, we have performed several experiments using the two types of architectures named before. Because of memory constraints it was not possible to augment the networks as it was done in the reference paper [1].

We tried to implement the attention augmented module to the ResNet-34. Because of memory problems and limited resources we have been able to substitute just the last layer of the network.

With the purpose of getting the attention augmented version of the Wide-ResNet (AA-Wide-ResNet), we replaced the first convolution performed in the third wide block by the augmented version. In the reference paper they augmented the convolution of the first layer of the three wide layers. Because memory limitations this was not possible to be performed.

The following results have been obtained training the ResNet-34 during 220 epochs, while the Wide-ResNet has been trained for 60 epochs.

Architecture	With Data Augmentation			
	Train Acc.	Val. Acc.	Test Acc.	Training time (hours)
ResNet-34	93.37%	87.55%	86.99%	~ 4
AA-ResNet-34	92.05%	86.79%	86.76%	~ 5
Wide-ResNet-28-10	94.99%	89.74%	89.10%	~ 3
AA-Wide-ResNet-28-10	93.49%	89.84%	88.40%	~ 6
AA-NRE-Wide-ResNet-28-10	93.28%	87.58%	88.19%	~ 6

Table 2: Performance of different network architectures with data augmentation.

From the previous table we can see that the augmented version of ResNet-34 does not perform better than the original. We concluded that this behaviour is triggered by the fact that self-attention is applied just to the last layer of the network. Therefore, the network is trying to find attention feature maps of a 2x2 image, consequently these feature maps would not help for the classification task. A similar behaviour is observed with the Wide-ResNet-28-10: just the last wide layer is augmented with self-attention therefore the attention feature maps do not have enough information to improve the classification accuracy.

Moreover, we also trained the same architecture but without the relative embedding mechanism (AA-NRE-Wide-ResNet-28-10). As expected, the the network with relative embedding has a better accuracy showing that the relative embedding is indeed an effective improvement.

The architecture of the Wide-ResNet increases the number of trainable parameters per layer, and this implies an increment in the consumption of computational resources, from which memory has proven to be a crucial part. Due to the usage of the batches without any kind of dimensionality reduction during the first augmented block, we have run into several memory problems since most GPUs were unable to properly handle the amount of data when applying self-attention, since this architecture generates one attention map per pixel. This has implied the need of reducing the magnitude of the experiment in order to gather relevant data which explains the performance of this architecture. There exists the need of using more, and more powerful GPUs to properly deal with images if we want to apply this architecture in the Wide-ResNet.

In the case of the performed experiments, the GPU employed has been a Nvidia Tesla P100 [22]. The original authors of the method counted on 8 different Nvidia Tesla V100 [23], a more powerful graphics processing unit, which can perform more floating point operations per second. Furthermore, having several of them allows parallelism, being able to split operations and perform them simultaneously, being faster and more efficient than using them sequentially. This lack of resources has impeded us to properly perform the attention augmentation, as was our intention.

Because of the memory problems and the computational limitations we faced, it was not possible to perform the tuning of the hyperparameters N_h, d_k and d_v.

6 Conclusion

Convolutional neural networks have proven to be a really effective mechanism to perform image classification tasks. Both the ResNet-34 and the Wide-ResNet-28-10 have shown a really good performance just by applying their original settings.

Data augmentation has been a really useful mechanism in order to enhance the generalization properties of these networks, providing an extra boost to the already good accuracy obtained in our experiments.

Reached this point, trying to increase even more the performance of already good classifiers lies on complex techniques. The use of self-attention can be one of them, but sadly, we did not have enough resources to put it on practice. The memory consumption it has shown has made unable to get results that outperform the ones obtained the aforementioned methods. With an accuracy around the 90% obtained with the Wide-ResNet-28-10, we do not think that employing self-attention is worth it regarding the tradeoff between improvement and time and resources consumption, even more if we take into account that there are different methods which have outperformed this mechanism.

In the case that we would have enough resources to perform this task, the next step should be looking for the optimal values for the parameters N_h, d_k and d_v, variables which define the way the augmented convolution works.

Summing up, the two architectures tested have demonstrated to be reliable networks for image classification exercises. The Wide-ResNet deserves a special mention since it has succeeded in its claim of being a network that is faster to train and which provides accurate results. Data augmentation mechanisms have also proven to be a useful and 'cheap' way of enhancing the generalization properties of our models, but on the other hand, self-attention has been a truly heavy technique whose results do not provide a real improvement over the good outcomes already achieved.

References

[1] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3286–3295, 2019.

[2] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[4] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[11] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[15] Baosong Yang, Longyue Wang, Derek Wong, Lidia S Chao, and Zhaopeng Tu. Convolutional self-attention networks. *arXiv preprint arXiv:1904.03107*, 2019.

[16] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Deep reinforcement learning with relational inductive biases. 2018.

[17] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[18] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[19] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning, 2019.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[21] Ashish Vaswani Peter Shaw, Jakob Uszkoreit. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[22] Nvidia tesla p100. https://www.nvidia.com/en-us/data-center/tesla-p100. Accessed: 2020-05-17.

[23] Nvidia tesla v100. https://www.nvidia.com/en-us/data-center/tesla-v100. Accessed: 2020-05-17.

[24] Wide-resnet-28-10 tensorflow implementation. https://github.com/akshaymehra24/WideResnet. Accessed: 2020-05-08.

[25] Resnet tensorflow implementation. https://github.com/jeffheaton/t81_558_deep_learning/blob/master/t81_558_class_06_3_resnet.ipynb. Accessed: 2020-05-03.