On the Relationship between Self-Attention and Convolutional Layers
ML Reproducibility Challenge 2020

Anonymous Author(s)
Affiliation
Address
email

Abstract

Networks based on self-attention are now paving the way towards achieving state-of-the-art performance in image recognition. In this report, we present a detailed study on the paper titled "On the Relationship between Self-Attention and Convolutional Layers" by Cordonnier et. al. [2020] which provides a theoretical evidence that attention layers can indeed behave like convolutions. We explore different variations of self-attention from other recent works and assess their effectiveness for image recognition. Further, we propose a new variant of the attention operation - *Hierarchical Attention* which shows improved performance with significantly lesser parameters. To facilitate further study, all the code used in our experiments are publicly available here.

1 Introduction

In computer vision, convolutional architectures [1, 2, 3, 4] have dominated across various image recognition tasks like classification, segmentation, etc. However, they have some limitations like lack of rotation invariance, inability to aggregate information based on the image content. This has inspired researchers to explore a different design space and introduce models with interesting new capabilities. Self-attention based networks, in particular Transformers [5] have become the model of choice in various Natural Language Processing (NLP) tasks. The major difference between transformers and previous methods, such as recurrent neural networks and convolutional neural networks (CNN), is that the former can simultaneously learn to attend to various parts of the input sequence. To utilize this capacity to learn meaningful inter-dependencies, many recent works have tried to incorporate self-attention [6, 7], some even replacing convolutions entirely [8, 9]. The main highlights of the paper (Cordonnier et. al. [2020]) is the following:

1. Theoretical proof that self-attention layers can behave similar to convolution layers.
2. Empirically validate the performance of the self-attention model on the image classification task using the CIFAR10 dataset. Visually show that self-attention applied to images learns convolutional filters around the query pixels.

In this study, we perform a detailed analysis on the various experiments outlined in the paper. Due to the emerging hype in utilizing attention methods for various vision tasks, we compare differences between various other works which try to utilize self-attention operation as the basic building block for image recognition models. We go beyond verifying their claims and introduce a new variant of the attention mechanism called Hierarchical Attention (HA) which can be easily adapted to any existing self-attention model. Detailed experiments show that the proposed method significantly outperforms existing self-attention variants with roughly $1/5^{th}$ the number of parameters.

1.1 Outline of this report

We structure the report as follows:

1. Section. 2 formally introduces the attention operation, followed by the fundamental working principle of the transformer.
2. We validate the claims made by the paper (Cordonnier et. al. [2020]) in Section. 3.
3. In Section. 4, we introduce other recent works who have tried to completely replace convolutions with self-attention and highlight differences between them.

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

4. During our experiments, we observe a flaw in the existing attention operation and introduce a novel Hierarchical Attention operation described in Section. 5. We compare the modified operation on various methods and empirically show significant performance gains.
5. Section. 6 concludes and suggests possible improvements for future research.

2 Fundamentals

In this section, we introduce the attention operation, first in terms of its origin in NLP and how it can be extended for images. We also provide a short introduction to transformers since most methods described in this report heavily rely on it.

2.1 Attention

Attention has been first proposed for NLP, where the goal is to focus on a subset of important words. Consequently, relations between inputs are highlighted that can be used to capture context and higher-order dependencies. The attention matrix $A(.)$ indicates a score between N queries Q, and N_k keys which indicates which part of the input sequence to focus on and $\sigma(.)$ is any activation function (generally $\sigma(.) = softmax(.)$).

$$A(Q, K) = \sigma(QK^T), \tag{1}$$

To capture the relations among the input sequence, the values V are weighted by the scores from Equation 1. Therefore, we have

$$SelfAttention(Q, K, V) = A(Q, K)V, \tag{2}$$

While in NLP, each element in the sequence corresponds to a word, the same idea is applicable for a sequence of N discrete objects, e.g. pixels. A key property of the self-attention model described above is that it is equivariant to the input order, that is, it gives the same output independent of how the N input tokens are shuffled. This is quite problematic in cases where the order actually matters like in images. Hence, a positional encoding is learnt for each token in the sequence and added before the self-attention operation. Hence, the Q and K vectors are derived from the summation of input X and positional encoding P.

2.2 Transformer Attention

The Transformer network is an extension of the attention mechanism from 2 based on the Multi-Head Attention (MHA) operation. Rather than computing the attention once, the MHA operation computes it multiple times (heads). This helps the transformer jointly attend to different information derived from each head. The output from each of these heads are concatenated before projecting onto a final output dimension. A transformer layer also contains a residual connection followed by a layer normalization and the overall operation can be summarized as:

$$\begin{aligned} MHA(Q, K, V, heads) &= concat_{heads}[A(Q, K, V)], \\ Transformer &= LayerNorm(MHA + MLP(MHA)), \end{aligned} \tag{3}$$

2.3 Positional Encoding for Images

There are two types of positional encoding that has been used in transformer-based architectures: the absolute and relative encoding. With absolute encodings, a (fixed or learned) vector P_p, is assigned to each pixel p. The relative positional encoding, introduced by [10] considers only the position difference between the query pixel (pixel we compute the representation of) and the key pixel (pixel we attend) instead of the absolute position of the key pixel.

The authors from the paper [11] have elegantly proved how an attention operation mimics a convolution. The main result is the following: *A multi-head self-attention layer with N_h heads of dimension D_h, output dimension D_{out} and a relative positional encoding of dimension $D_p >= 3$ can express any convolutional layer of kernel size $\sqrt{N_h} * \sqrt{N_h}$ and $min(D_h, D_{out})$ output channels*. We strongly urge the readers to refer to their paper to get a complete understanding.

3 Experiments for Reproducibility

The aim of this section is to validate the results claimed by the paper - to examine whether self-attention layers in practice do actually learn to operate like convolutional layers when trained on standard image classification tasks. For all our experiments mentioned in this report, we use the CIFAR10 dataset to benchmark the performance of the model for the classification task.

3.1 Dataset - CIFAR10

The CIFAR-10 dataset [12] consists of 60,000 32x32 colour images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images. The classes are completely mutually exclusive.

The results mentioned in the paper uses a fully-attentional model consisting of 6 multi-head self-attention layers each with 9 heads. In all the experiments, the input image undergoes a 2x2 down-sampling operation to reduce its size. The final image vector is derived by average-pooling the representations derived from the last layer and then passed to a linear layer for classification. Please refer to Table. 4 for a detailed list of hyper parameters used in each experiment. The paper uses three different implementations of the attention mechanism to completely validate their claims. We shall now describe them individually, followed by the results section.

3.2 Quadratic encoding

Lemma 1 from the paper [11], derives a representation of the attention matrix which can be modelled as an isotropic bivariate Gaussian distribution. Hence to validate their claims, all the attention matrices in the model are replaced with these Gaussian priors. The authors also provide figures to intuitively prove that self-attention indeed tries to learn convolutional filters around a query pixel.

3.3 Learned Relative Position Encoding

Learned Relative Position Encoding (also referred to as Learned embedding) is another experiment where the authors try to understand the relevance of positional encoding in fully-attentional models. The authors implement a 2D relative positional encoding scheme where a positional encoding vector is learned for each row and column pixel shift. In one experiment (Learned embedding w/o content), the authors completely discard the input data and compute the attention weights solely with the derived encoding. In another setting (Learned embedding w/ content), both the positional and content based attention information is used which corresponds to a fully-blown stand alone self-attention model.

We closely follow the official implementation for reproducing the three embedding schemes and attention mechanisms. The obtained performance is very similar to what has been reported in the paper. Table 1 compares the results mentioned in the paper and the results obtained using our implementation. Further, we try increasing the number of heads to 16 but there seems to be no significant impact to the model's performance (seen in Table. 2).

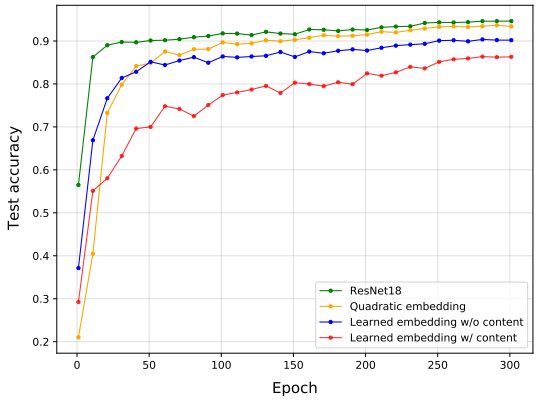
Model	Accuracy (paper)	Accuracy (ours)	# of params
ResNet18	0.938	0.946	11.2M
Quadratic embedding	0.938	0.937	12.1M
Learned embedding w/o	0.918	0.904	12.3M
Learned embedding w/ content	0.871	0.864	29.5M

Table 1: Test accuracy (paper vs ours) on CIFAR-10 and model sizes; 9 heads

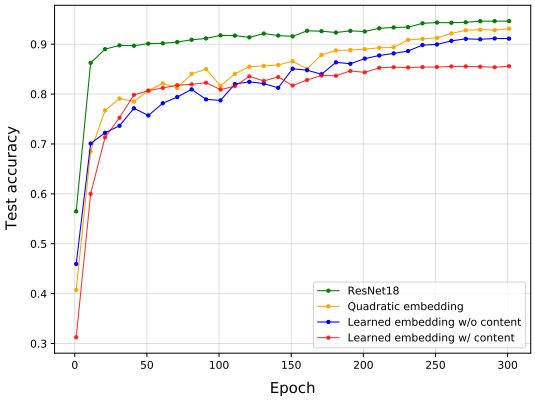
Model	Accuracy (paper)	Accuracy (ours)	# of params
ResNet18	0.938	0.946	11.2M
Quadratic embedding	0.938	0.931	12.1M
Learned embedding w/o content	0.918	0.912	12.3M
Learned embedding w/ content	0.871	0.856	29.5M

Table 2: Test accuracy (paper vs ours) on CIFAR-10 and model sizes; 16 heads

The performance of the fully-blown self attention model (Learned embedding w/ content) is still quite far away from standard convolutional neural networks like ResNets. Quadratic encoding, Learned embedding w/o content perform better as information is explicitly given, avoiding the need for the model to learn extra features.We discuss on how further improvements can be improved in the later sections of this report (Section. 5). As seen from Fig. 1, the ResNet model tends to converge faster than the attention-based methods. Increasing the number of heads seems to slow down the convergence even more as the model needs to learn more number of parameters.

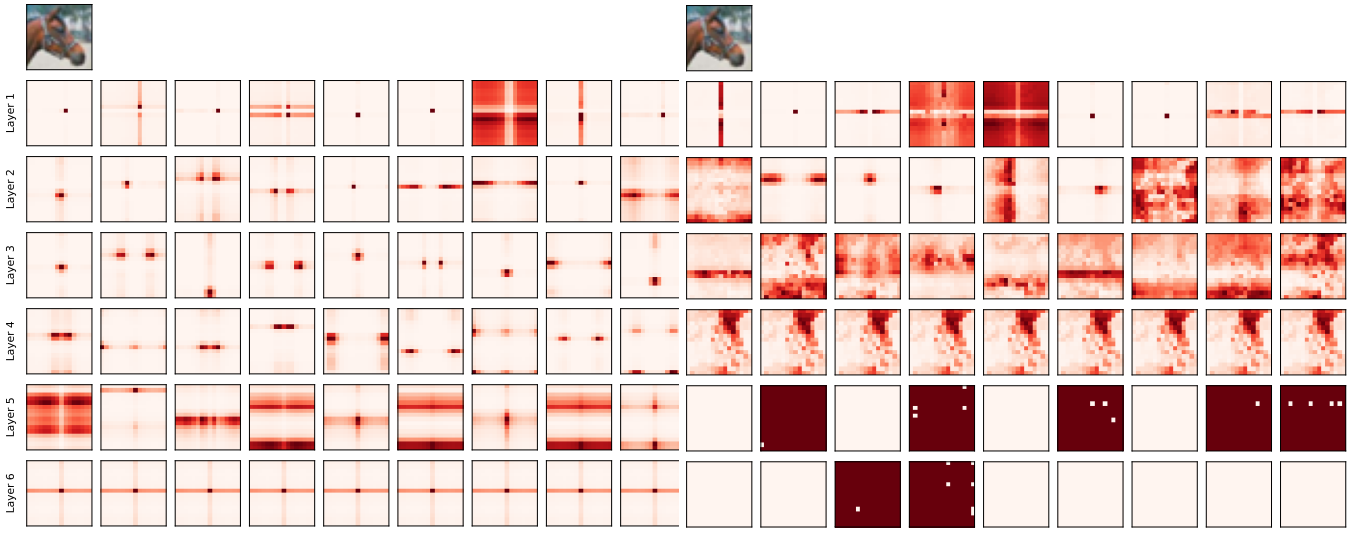


(a) Models with 9 heads



(b) Models with 16 heads

Figure 1: Test performance of models on CIFAR10



(a) Learned embedding w/o content attention

(b) Learned embedding w/ content attention

Figure 2: Attention probabilities for a query pixel at the center of the given image

Fig. 2 visualises the attention probabilities for a given query pixel (in fig: center of the given image). We can see that in both the cases, the attention layers try act like convolutional filters. The first few layers (1,2) tend to focus on local patterns, while deeper layers shift the attention further away from the query pixel. One important thing to note is that in the fully blown attention model (Fig. 2b), the last two layers seem to not contribute much to overall features learnt. We shall come back to this in Section 5.

4 Other Attention-based networks for Vision

Due to the recent hype in replacing convolutions with attention layers, there is a strong requirement to highlight differences among these recent methods. Among these works, we choose two popular and unique papers - "Exploring Self-attention for Image Recognition" by Zhao et. al. [2020], "An Image is worth 16x16 words: Transformers for Image Recognition at scale" by Dosovitskiy et. al. [2020].

4.1 Pairwise and Patchwise self attention

The pairwise self-attention, is essentially a general representation of Eqn. 2. It is fundamentally a set operation, does not attach stationary weights to specific locations and is invariant to permutation and cardinality. The paper presents a number of variants of the pairwise attention that have greater expressive power than dot-product attention. Specifically, the weight computation does not collapse the channel dimension and allows the feature aggregation to adapt to each channel. It can be mathematically formulated as follows:

$$\begin{aligned} y_i &= \sum_{j \epsilon R_i} \alpha(x_i, x_j) \odot \beta x_j \\ \alpha(x_i, x_j) &= \gamma(\delta(x_i, x_j)) \end{aligned} \tag{4}$$

Here, i is the spatial index of feature vector x_i, $\delta(.)$ is the relation function mapped onto another vector by $\gamma(.)$ to create adaptive weight vectors $\alpha(x_i, x_j)$ and then aggregate feature vectors obtained from $\beta(.)$. An important point to note here is that δ can produce vectors of different dimensions when compared to β, allowing for a more expressive weight construction.

The patch-wise self attention is a variant of the pairwise operation, where x_i is replaced by a patch of feature vectors $x_{R(i)}$ which allows the weight vector to incorporate information from all the feature vectors in the patch. The equations are hence rewritten as:

$$\begin{aligned} y_i &= \sum_{j \epsilon R_i} \alpha(x_{R(i)})_j \odot \beta x_j \\ \alpha(x_{R(i)}) &= \gamma(\delta(x_{R(i)})) \end{aligned} \tag{5}$$

However, this operation is no longer permutation or cardinality invariant. Both the above methods have shown strong performance in image classification specifically in large datasets (explained more in Section 4.3). One major bottlenecks in this method is the size of the attention matrices leading to huge memory requirements.

4.2 Vision Transformer

A recent work Vision Transformer, has successfully shown that reliance on convolutions is no longer necessary and a pure transformer outperforms all convolution based techniques significantly when pre-trained on large amounts of data. An image is split into patches which is then projected onto a another representation using a trainable layer and then passed through a standard set of transformer operations as described in Eqn. 3.

4.3 Inductive biases of Transformers

The above two methods tend to perform really well with sufficient training and yield modest accuracies in small datasets. As mentioned in the Vision Transformer paper, transformers lack inherent biases inherent to CNNs like translation equivariance, two-dimensional neighborhood structure, and hence do not generalize well when trained on insufficient amounts of data. Only the Multi-Layer Perceptron (MLP) layers used in these methods are local and translation equivariant, while the self attention layers are global. Hence its almost equivalent to learning these spatial relationships from scratch which requires sufficient training. This could also be the reason for the improved performance in the case of Learned embedding w/o content, Quadratic encoding (Table. 2), as they are directly replacing the attention matrices with positional information. However, this changes completely when provided sufficient training.

5 Hierarchical Attention

We now introduce a novel attention operation Hierarchical Attention (HA) which tries to overcome some problems in the normal attention operation. We try to build an intuition of the problem, introduce the alternate attention method and perform detailed experiments to showcase its superiority.

5.1 The problem

The most important step of the self-attention operation is the generation of the attention matrix of size $N * N$. In NLP, the value of N tends to be small (<100) in most cases as we are dealing with words in a sentence. On the contrary, images when broken down into a sequence of pixels are very large in number, hence creating a large attention matrix. This has also been talked about in the case of applying transformers in point clouds where the number of points are

very large (>1000). Section 4.1 in the paper [11] also mentions that Learned embedding with content is harder to train which was also noticed in our experiments. We believe that this is mainly due to the increased expressive power of the attention matrix when N is a very large number. Although not mentioned in the paper [13], we would like to highlight that splitting an image into patches makes the value of N smaller which could possibly validate our intuition. A simple and naive way to overcome this problem is to reduce the number of heads or layers but the this is not an effective method as the model loses its capacity to learn strong features. As an alternative we propose a Hierarchical Attention (HA) operation which helps the model iteratively refine its features.

5.2 Methodology

In all the above mentioned methods, independent self-attention layers are stacked sequentially and the output from one is passed onto the next to derive the Q, K and V vectors. However, in the HA operation, the weights are shared across the different layers and only the Q, V vectors are updated while K remains the same. In this case, the query pixel tries to refine its representation in every layer by focusing on the corresponding portions from the K vector, and aggregates the required information from the V vector. This helps the model hierarchically learn complex features. Figs. 3, 4 visualize the normal attention and the hierarchical attention operations. For the sake of simplicity, we do not visualize the entire inner workings of the transformer like residual connections, layer normalization, etc. Please note that the HA operation can be easily adapted to any existing attention-based network as described in the next section.

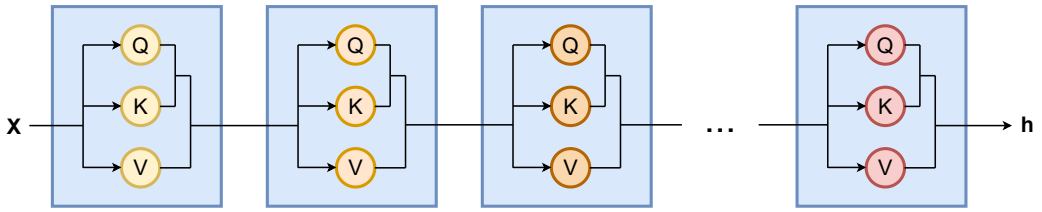


Figure 3: Normal Attention (Scaled Dot Product): The projection weights in each layer are different and independently learnt (different color). Q, K, V vectors are updated in each layer.

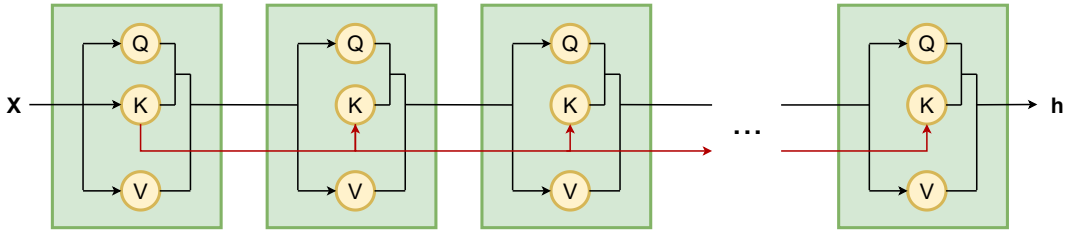


Figure 4: Hierarchical Attention (HA): The projection weights are shared across all the layers. Only Q, V vectors are updated while the K remain the same in each layer.

5.3 Results

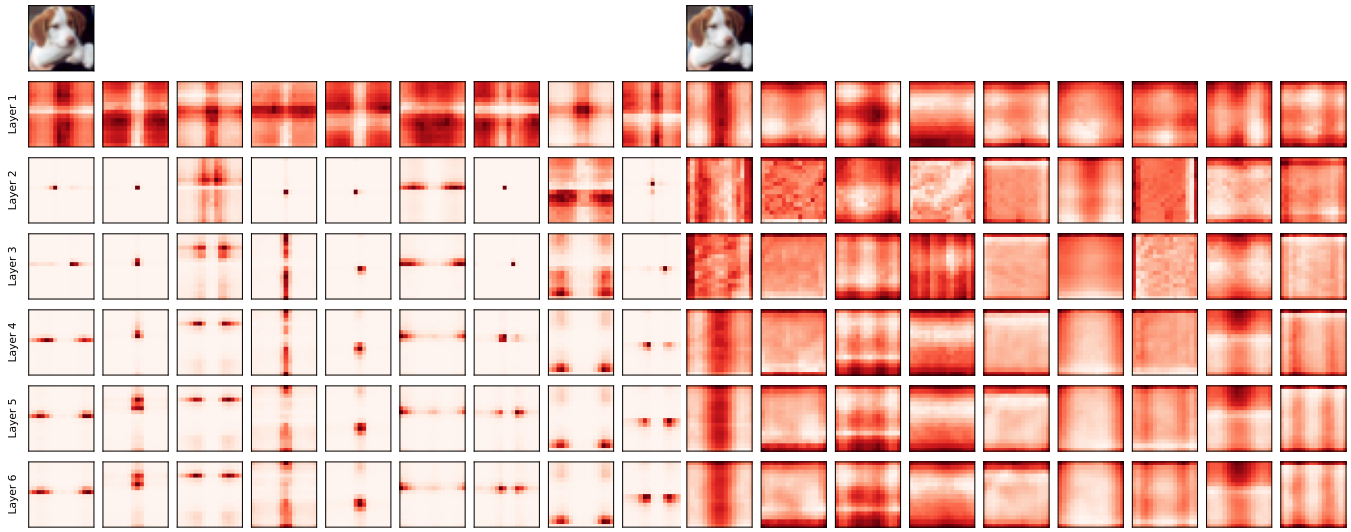
To validate our intuition and to prove the effectiveness of the proposed method, we incorporate HA in all the methods described above without modifying the overall structure and compare it with the normal case. Table 3 compares accuracy for each model with its corresponding HA variant. We see significant improvements in the performance (at least 5% gain) in each case while being able to reduce the parameters by almost $1/5^{th}$ of the original model. We also visualize the attention probabilities for the Hierarchical Learned 2D w/ embedding in Fig. 5. Fig. 5a represents the attention weights for a query pixel located at the center of the image, while Fig. 5b represents the average focus of the model in a given image. Further details on how visualization is done along with corresponding figures for other methods are given in the Appendix. 7.2.

We summarize the hierarchical operation as follows:

1. Enable weight sharing between the layers of the model by reusing the K vector and updating the Q, V vectors only. This helps the model progressively extract higher level features.
2. The method of progressive refinement ensures that the attention matrices do not get over expressive. This leads to a significant improvement over the corresponding non-hierarchical cases.

Model	Normal SA			Hierarchical SA		
	Accuracy	# of params	Wall time	Accuracy	# of params	Wall time
Learned embedding w/ content	0.871	29.5M	5.374	**0.910**	4.75M	4.703
SAN Pairwise	0.818	3.74M	16.586	**0.853**	0.35M	12.913
SAN Patchwise	0.857	3.74M	9.773	**0.912**	0.36M	7.419
Vision Transformer	0.843	37.4M	7.094	**0.877**	6.53M	6.302

Table 3: Comparison between models using normal SA and HA. Wall times are average inference times in milliseconds for the models over 300 iterations



(a) Attention probabilities for a query pixel at the center of the given image

(b) Average Attention probabilities to determine overall focus of the model

Figure 5: Visualizing the attention weights from a trained Hierarchical Learned 2D w/ embedding

3. The total number of parameters in the model is independent of the number of layers and hence the model can be made deeper without worrying about memory constraints. This property of the model significantly reduces the number of parameters.

4. Even if the model is provided more layers than are necessary, every layer learns to attend to a different pattern. The new features learnt at every layer add on to those learnt by the previous ones, which provides its characteristic hierarchical nature. By visualizing the attention scores on a test image, we obtain convincing evidence to support this hypothesis. (Fig. 5 where the model gradually refines the regions each head attends to, which is in contrary to the case in Fig. 2b)

6 Conclusion

In this report, we study the application of self-attention for image recognition, specifically image classification. We validate the author's claims by performing detailed experiments on the CIFAR10 dataset. We discuss two recent papers which propose stand-alone attention based models for image classification. Finally, we introduce a novel Hierarchical Attention operation which significantly improves performance across various models while also reducing the number of parameters. These preliminary results raises various questions like: do we actually need multiple independent layers in large transformers?, does this improved performance also translate to large datasets and across various other image recognition tasks like object detection, image segmentation, etc? We would like to answer all these questions and provide a more rigorous understanding of the proposed method.

References

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[2] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[4] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

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

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

[7] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *CoRR*, abs/1711.07971, 2017.

[8] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *CoRR*, abs/1906.05909, 2019.

[9] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation, 2020.

[10] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019.

[11] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *CoRR*, abs/1911.03584, 2019.

[12] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

7 Appendix

7.1 Hyperparameters

Model	Epochs	Mini batch size	Optimizer		Scheduler	
			Type	LR	Type	Warm up/step epochs
ResNet18	300	100	SGD	0.1	Cosine	10
Quadratic emb.	300	100	SGD	0.1	Cosine	15
Learned emb. w/o content	300	100	SGD	0.1	Cosine	15
Learned emb. w/ content (SA/HA)	300	100	SGD	0.1	Cosine	15
SAN Pairwise (SA)	300	256	SGD	0.1	Multistep	100, 200, 250
SAN Patchwise (SA)	300	256	SGD	0.1	Multistep	100, 200, 250
Vision Transformer (SA)	100	32	SGD	0.01	Cosine	5
SAN Pairwise (HA)	300	40	SGD	0.1	Cosine	10
SAN Patchwise (HA)	300	100	SGD	0.1	Cosine	10
Vision Transformer (HA)	100	32	SGD	0.01	Cosine	5

Table 4: Hyperparameter configuration for all experiments. SA refers to normal self-attention and HA refers to Hierarchical Attention. The momentum and weight decay for SGD were set to 0.9 and 0.0001 respectively for all experiments.

7.2 Attention Visualization

The attention maps included in this report are visualized in two ways - attention probabilities for a given query in an image, average attention to determine the overall focus of the model in the given image. Given an $N * N$ attention matrix, every row/column (based on the softmax normalization axis) represents the relationships between a query pixel and the others. To visualize the average attention, every element is first divided by its row-wise sum to ensure that all values are in scale. Then the mean of the attention probabilities in each column is computed to determine an importance value for each pixel. If a point is strongly related to multiple points, the importance value will be higher determining that the model has a stronger focus on that given pixel. We now visualize the attention maps for different methods for both the hierarchical and non-hierarchical case.

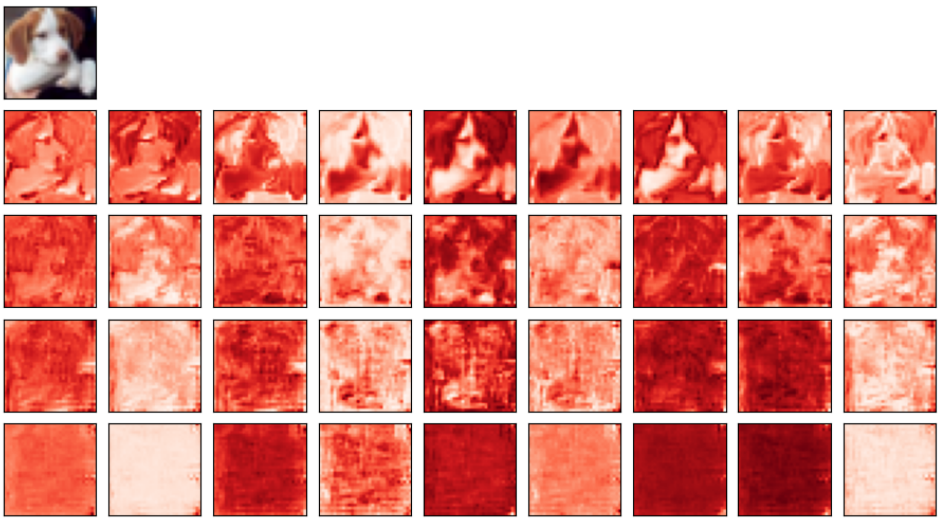


Figure 6: Patchwise Attention + Heir (Average)

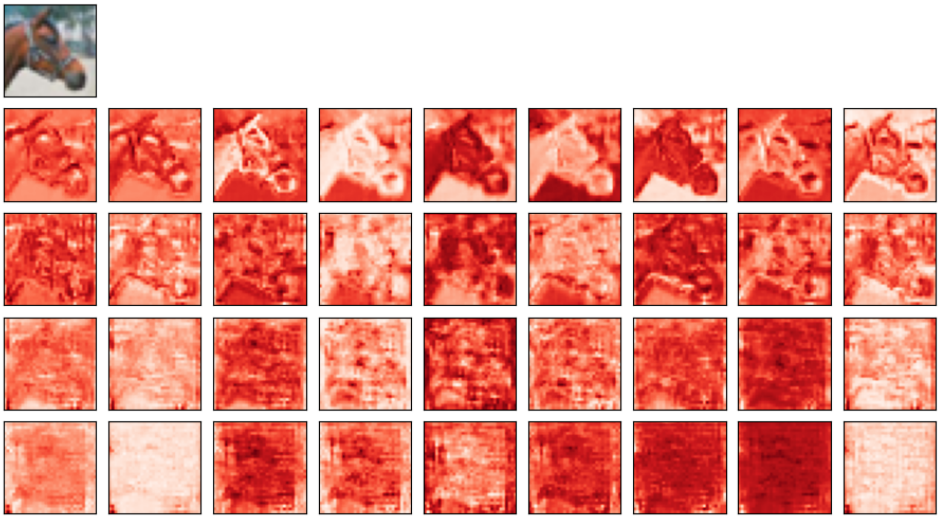


Figure 7: Patchwise Attention + Heir (Average)

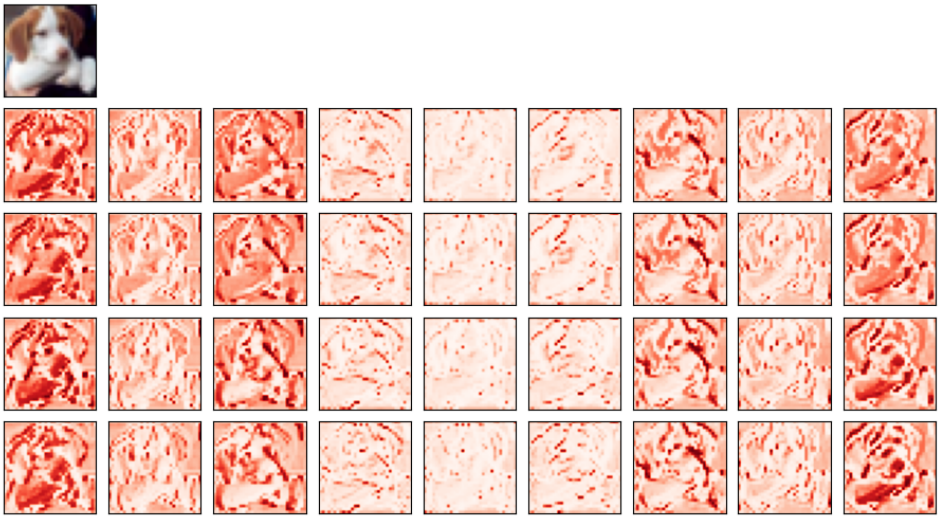


Figure 8: Pairwise Attention + Heir (Average)

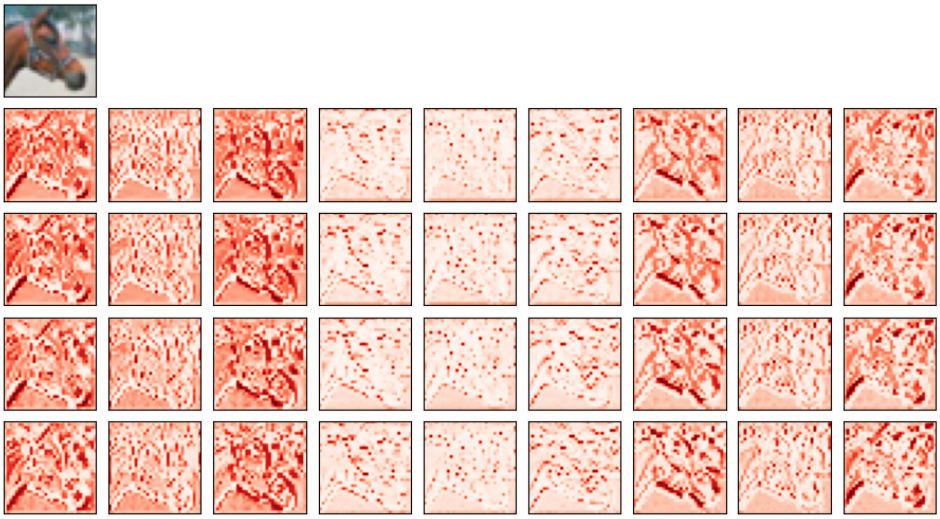


Figure 9: Pairwise Attention + Heir (Average)

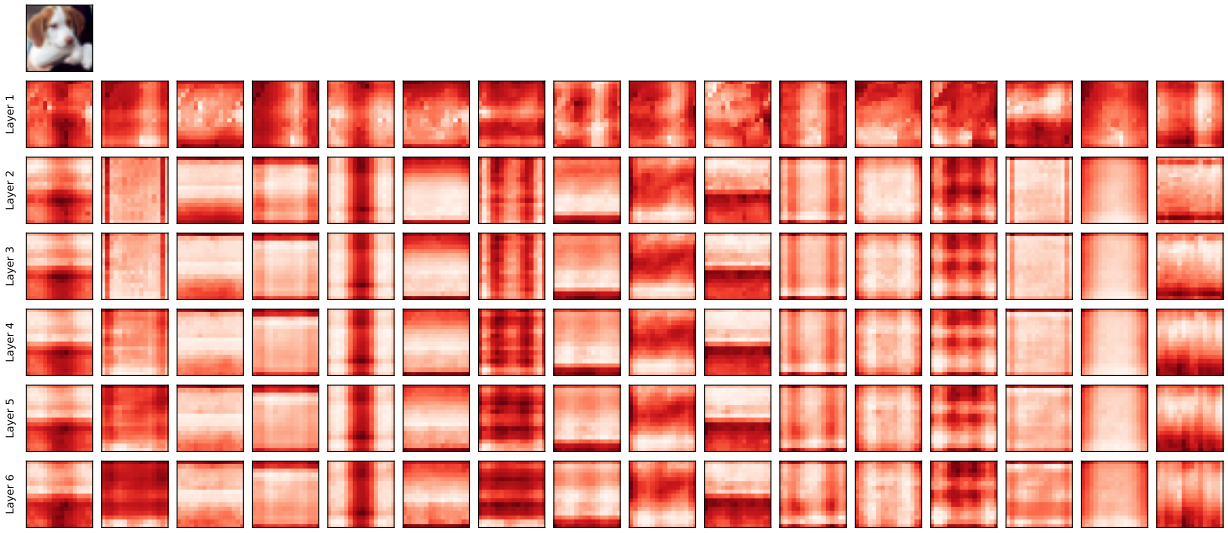


Figure 10: Learned 2D + Heir (16 Heads, Average)

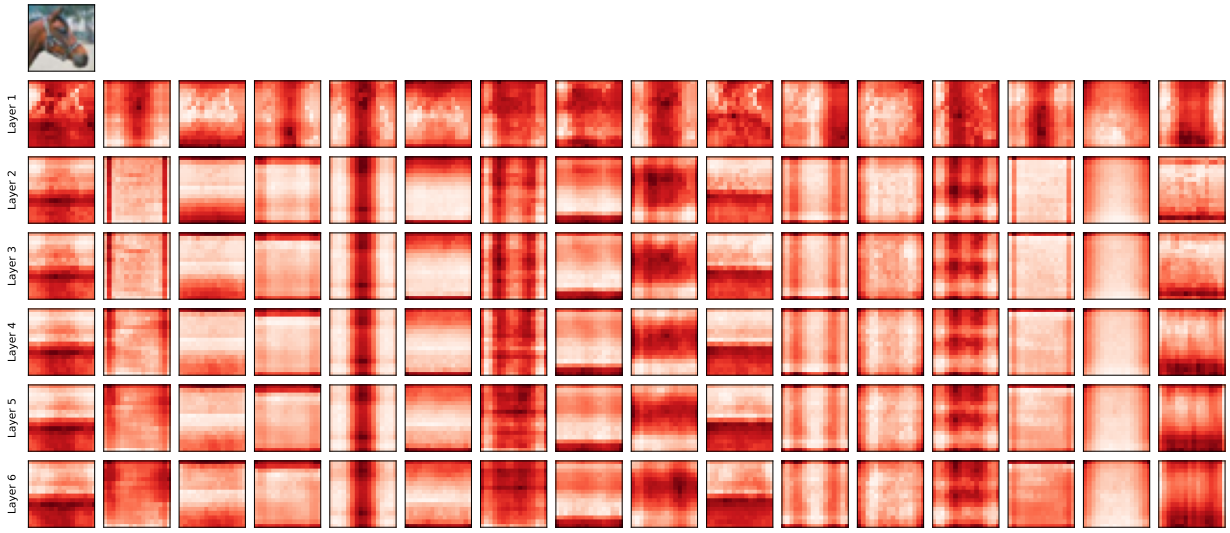


Figure 11: Learned 2D + Heir (16 Heads, Average)

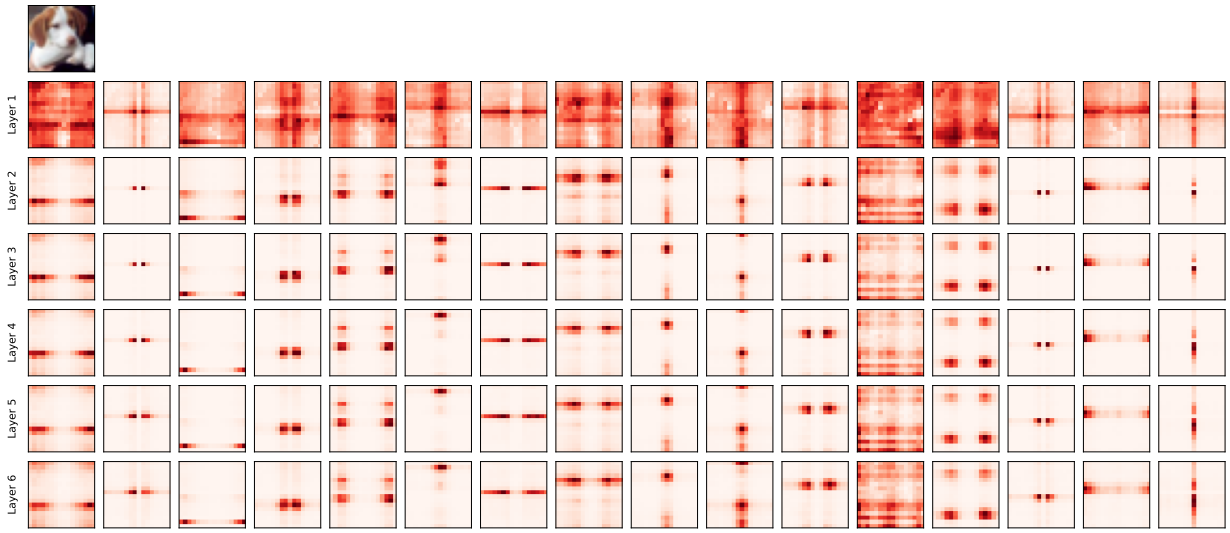


Figure 12: Learned 2D + Heir (16 Heads, Query pixel)

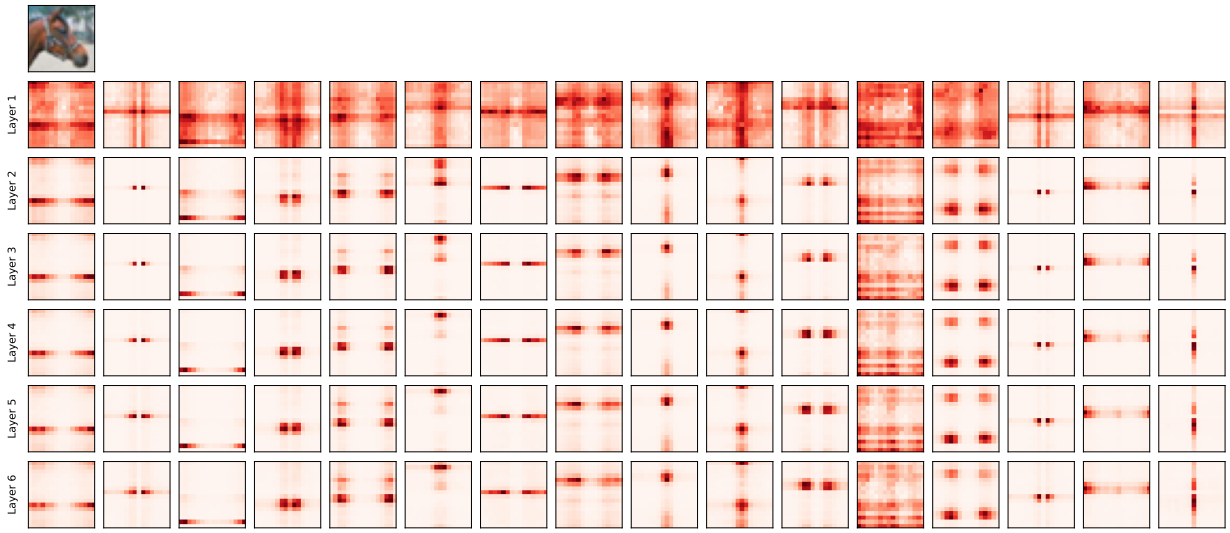


Figure 13: Learned 2D + Heir (16 Heads, Query pixel)

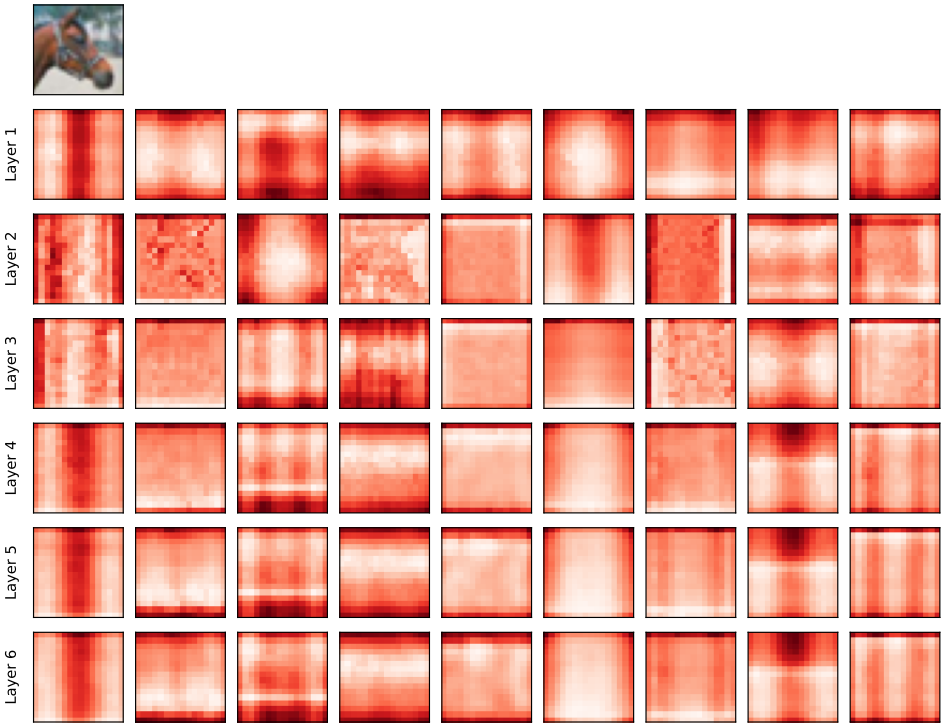


Figure 14: Learned 2D + Heir (16 Heads, Average)

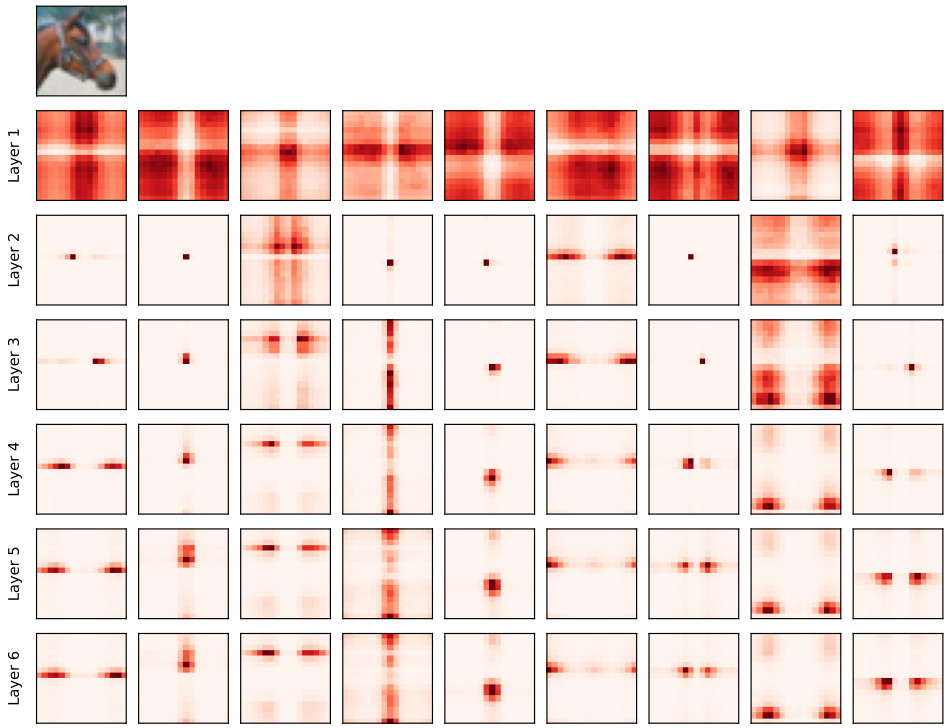


Figure 15: Learned 2D + Heir (8 Heads, Query pixel)

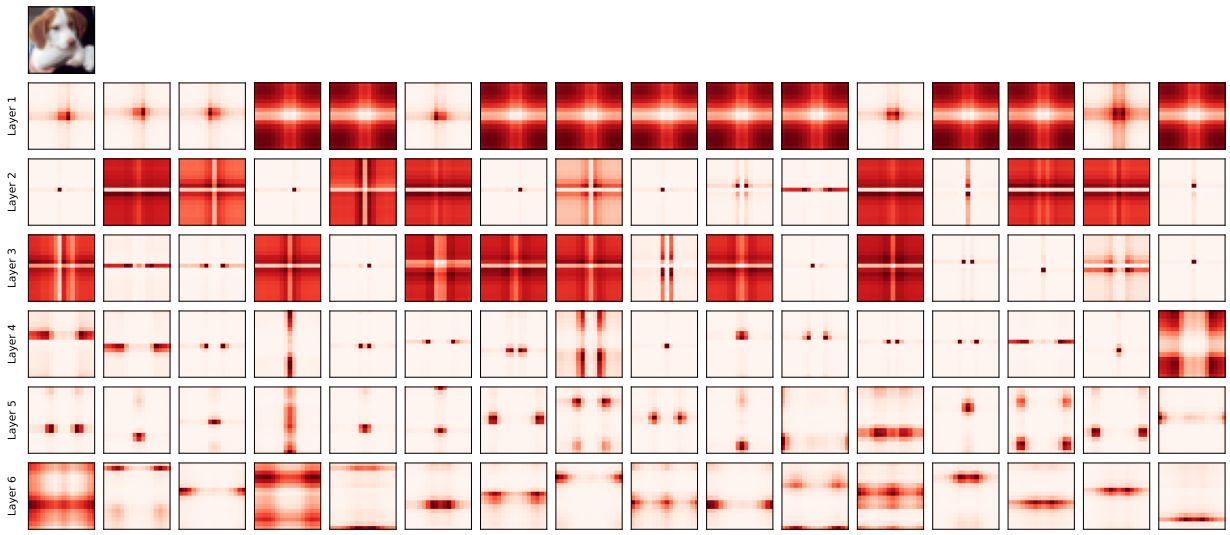


Figure 16: Learned 2D w/o embedding + Heir (16 Heads, Query pixel)

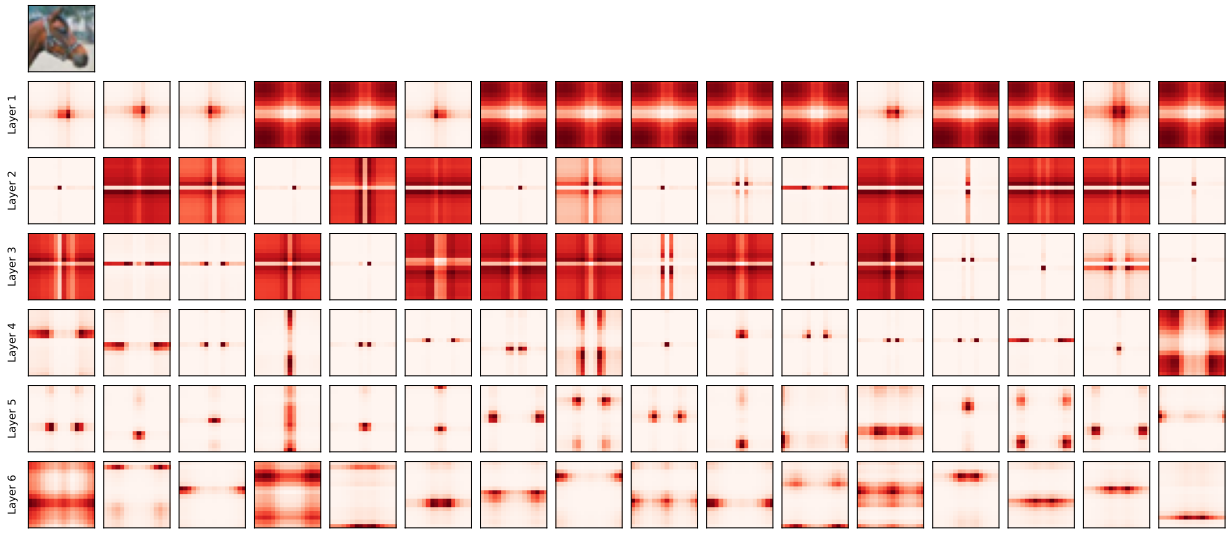


Figure 17: Learned 2D w/o embedding + Heir (16 Heads, Query pixel)

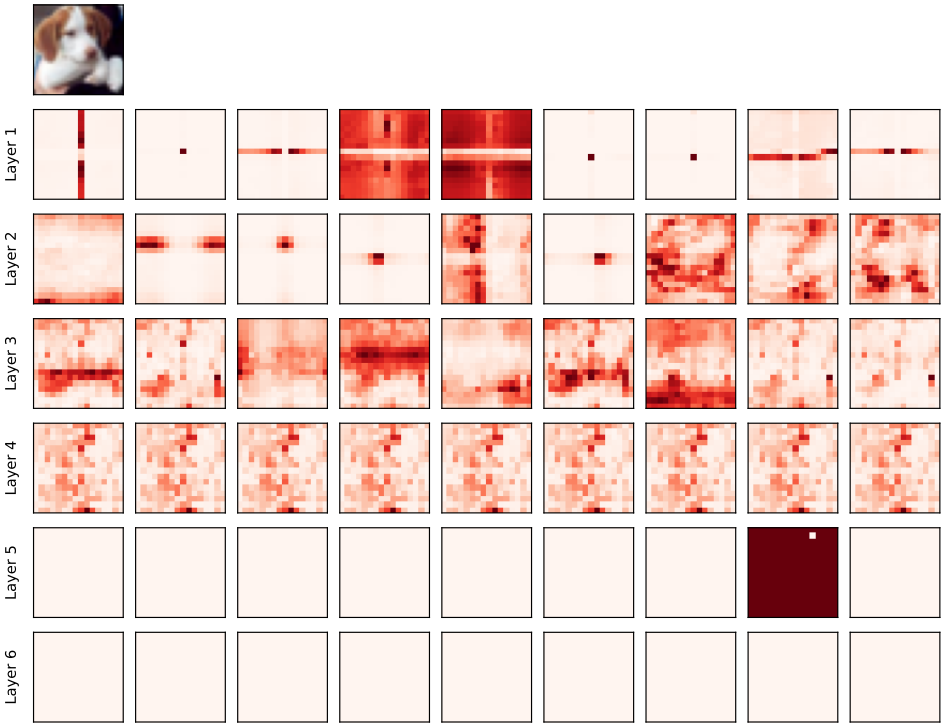


Figure 18: Learned (8 Heads, Query pixel)

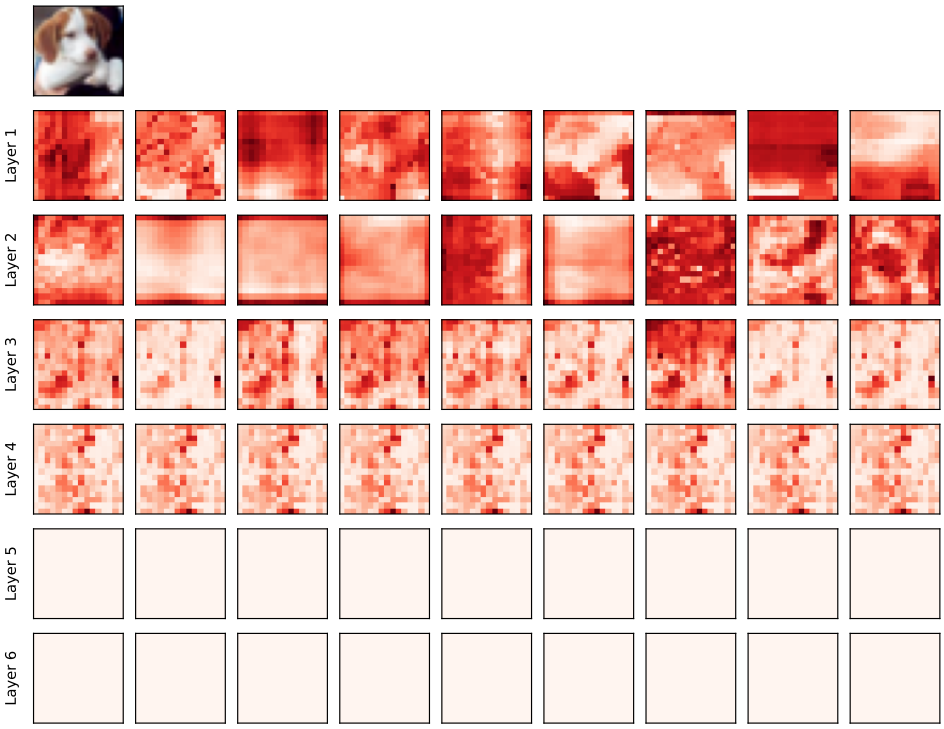


Figure 19: Learned 2D (8 Heads, Average)