Refining Residual Mappings Using Regions of High Attention

Charles Wang
Stanford University
sheemee@stanford.edu

Abstract

In this paper, we explore a method to refine residual mappings with an attention map. We achieve this by paring a classification and attention branch together. Our attention branch generates an attention map using a self-supervised Vision Transformer and learns attention mappings that match the shape of features on the classification branch. These attention maps are then used to mask the residuals in the classification branch. We find that this process allows our models to achieve higher classification accuracy than standard residual networks.

1. Introduction

Image recognition is a core discipline in the field of machine learning and has been successfully adopted by industries ranging from farming to medicine. Until recently, convolutional neural networks (CNNs) have dominated the field due to their intrinsic ability to learn local features and generalize to unseen images. As humans, there are important features that we look for in order to identify animals. We would check for its overall silhouette, the shape of its eyes, how many legs it has, whether it has wings, etc. A CNN does not have this intuition, and must rely on linear convolutions and non-linear activation functions to learn features for local pixel neighborhoods. However, some important features may be far from each other and will not be captured within the receptive field of a single convolution. Take for example a 200x200 image of an animal. Important features may be over 100 pixels away from each other e.g. the ears and tail of the animal. If we take multiple layers of standard 5x5 convolutions with stride 1, it would take over 23 layers before the receptive field could cover both of those features. With 2x2 max pooling every third layer, it would still take 9 layers before the receptive field reaches over 100 pixels. (117 pixels to be exact)

Another aspect of visual processing that ML models do not inherently have is the ability to pay attention to the foreground of an image and ignore the background. If we are given an image of an object we have never seen before, we can easily separate it from the background and learn what the object should look like. This is because our brains have been "training" from birth and have the ability to quickly identify relevant and irrelevant information. In ML models, self-attention can solve this, and recent development in self-attention has shown that image recognition using self-attention can lead to better classification results than CNNs. Though originally used for natural language processing, the advent of transformers in 2017 have allowed for breakthroughs in image recognition accuracy on standard computer vision datasets. Vision transformers can be argued to be more general than an MLP and can learn the global context of an image, since each iteration updates the learnable embedding using attention over all image patches.

Our paper is inspired by these recent developments, and aims to combine CNN based architectures with recent developments in self-attention. Transformers been proven to have higher classification accuracy than all state of the art CNN models, but in exchange require more training data. The authors of An Image is Worth 16x16 words explain that this is due to pure-transformer classifiers lacking the inductive biases of CNNs. [3] Thus, our model will follow the hybrid architecture in order to reap the locality advantages of CNNs as well as the attention benefits of Transformers.

The input to our model will be a single image, which will be sent to a classification branch and an attention branch. Both branches are modeled after ResNets. Our goal is to use attention maps from the attention branch to generate better residual mappings for the classification branch. Details about model structure will be explained later in this work.

2. Related Work

2.1. Convolutional Neural Networks

Inspired by previous works on neocognitrons, the first paper on modern CNN architecture was published in 1989. Since then, there have been developments improving each aspect of the CNN. New optimizers helped weights to converge better, new activation functions allowed for less vanishing gradients, and new architectures helped models become more accurate than ever.

These models all relied on convolutions and non-linear activations as a means of learning relations between pixels and classifying images. Such learned relations are able to provide state-of-the-art classification accuracy, and research by Zhuang Liu et al. [8] has shown that CNNs can still outperform vision transformers in the modern day while being more simple and efficient.

2.2. Self-attention in Image Recognition

Self-attention has been shown to be a powerful tool for image recognition and has been used in varying degrees to augment the efficacy of CNNs. We will examine some successful models, beginning with ones involving minimal use of attention, all the way to those completely based on attention.

CNN classification accuracy can be increased through attention, while minimal modification to the general architecture. This has allowed such models to continue exploiting the benefits of local connectivity. One such design is Woo et al.'s Convolutional Block Attention Module. [11] This block uses spatial and channel wise attention to modify the outputted features of a convolutional layer without changing the size, and can thus be added to any model without changing the architecture. Wang et al.'s [10] residual attention network uses attention modules to mask the output of layers, and can also be built directly on any conventional CNN architecture.

Deeper application of attention leads us to models where the convolution step is completely replaced by attention in a standard CNN architecture. Zhao et al. explore a patch-wise self-attention block that is more powerful than convolutions. [12] These patches are over local neighborhoods similar to how a convolution kernel only takes in a small patch during each pass.

Finally, there are models that classify images using transformer models as proposed by Vaswani et al. [9]. The first application of this is Dosovitskiy et al.'s Vision Transformer. [3] Their model splits an image into equally sized non-overlapping patches before applying appropriate transformations and feeding it into a standard Transformer model.

Our work aims to explore a field somewhere in the middle.

2.3. Self-Supervised Learning

In self-supervised learning, a single image is augmented twice with different augmentation schemes. These two images are sent into two networks with identical architectures, with the goal of generating similar features. The goal of self-supervised learning is to minimize the difference between features, which ideally creates robust features. Once the self-supervised training is complete, the student branch can be discarded and the teacher branch can be used for downstream tasks. Variations on the standard self-supervised contrastive learning paradigm, such as self-DIstillation with NO supervision (DINO) [2], demonstrate the efficacy of self-supervised pretraining.

3. Methods

We will begin with an overview of our model, then go explain the relevant details of each piece, before finally putting together the entire model.

Given an input image $\mathbf{x}$, we generate attention mask $\mathbf{x}_a$. Our model has two branches, once for learning classification and one for applying attention maps onto the classification branch. From now on, we will refer to the classification branch as B_c and the attention branch as B_a. Image $\mathbf{x}$ is used as input to branch B_c and its corresponding attention map $\mathbf{x}_a$ is used as input to branch B_a.

3.1. Vision Transformer

We first provide an overview of the mechanisms behind the Vision Transformer that are relevant to our model. For further implementation or algorithmic details, please refer to Dosovitskiy et al. [3]. Attention is a key feature of the Vision Transformer, and the original Transformer model used for NLP is almost completely unmodified for computer vision applications.

Attention is computed on a given query Q, key K, value V, and dimension d_k using the following equation:

$$Attention = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Using these ideas, the Vision Transformer first tokenizes images to allow for input to a standard Transformer architecture. An image $\mathbf{x} \in R^{H\times W\times C}$ is split into discrete, square, non-overlapping patches of size P. All patches are then flattened into shape $P^2 \cdot C$, linearly embedded, and concatenated with learnable positional encodings. These patch vectors are prepended with a learnable class embedding, which we will refer to as $[CLS]$ in order to maintain consistency with the works we are using. The vectors are then fed into a standard Transformer encoder, where each transformer block consists of multiheaded attention followed by a two layer MLP.

In each step information from all vectors are aggregated into the $[CLS]$ token. This token is what we will be using to generate attention map x_a. More details on how our attention map is created will be discussed in Section 5, since there are various ways we could genereate our map.

3.2. DINO (self-DIstillation with NO supervision)

We also provide an overview of the self-supervised learning paradigms that are relevant to our model.

The knowledge distillation self-learning paradigm consists of a student g_s and teacher g_t model. These models take an input $\mathbf{x}$ and output a probability P given by:

$$P(x)^{(i)} = \frac{exp(g(\mathbf{x})^{(i)}/\tau)}{\sum_j exp(g(\mathbf{x}^{(j)})/\tau)} \quad (2)$$

where τ is a temperature that is controls the sharpness of the probability distribution. [6] The cross entropy between the probabilities of models g_s and g_t are then minimized through training. DINO follows this process, but where it differs from traditional knowledge distillation is that the teacher network is built from past iterations of a student network.

DINO uses Vision Transformers as the backbone for the student and teacher networks. Self-supervised training was performed with unlabled ImageNet-1K data, and the cross entropy loss between teacher and student networks was minimized like in standard self-supervised learning. The model was trained end to end, including the fully connected layer at the end for classification. This resulted in a teacher network that could generate strong features and that could achieve state-of-the-art accuracy when fine-tuned on downstream tasks such as image classification. In our case, we simply discard the classifier and final MLP layer since we only need the $[CLS]$ token.

For our model, we use a ViT backbone with patch size 8×8 pretrained by Caron et al. [2], so please refer to their paper for further implementation details such as update rules for the teacher network during training. This backbone is not fine-tuned on our datasets, because intuitively, the object in the image does not need to be known in order to pay attention to it. For example, if a human receives an image of something they do not recognize, they will still know where they should look and what parts are background that can be ignored.

3.3. Classification and Attention Stems

The classification and attention branches start with a stem block, where we downsample the images by 4x. A typically ResNet accomplishes this downsampling using a 7×7 convolution with stride 2 and a 2×2 max pooling layer. However, evidence has shown that taking macro design rules from Transformers and applying them to CNN architectures leads to an increase in classification accuracy. [8]. Thus, our downsampling is performed using a 4×4 convolution layer and stride 4 to mimic the patches taken by a Vision Transformer. Figure 1 shows the full stem blocks.

3.4. Classification and Attention Branches

Branch B_c consists of a modified ResNet basic block architecture, where the residuals added after each block are first modified by the corresponding soft attention map learned from branch B_a. We will refer to these residuals as $\mathcal{F}(\mathbf{x})$ to maintain consistency with the original ResNet paper, [5] and will refer to any blocks in this branch as a classification block. The convolution strides follow the same striding pattern as a standard ResNet8 model.

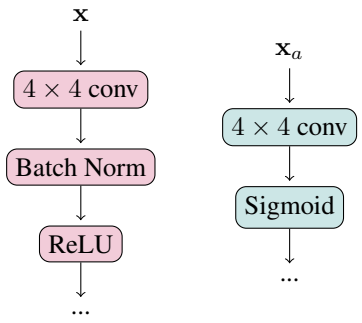

Figure 1. The classification stem block for branch B_c (left) and the attention stem block for branch B_a (right). For clearness, layers on branch B_c will always be shown in pink and layers on branch B_a will always be shown in teal.

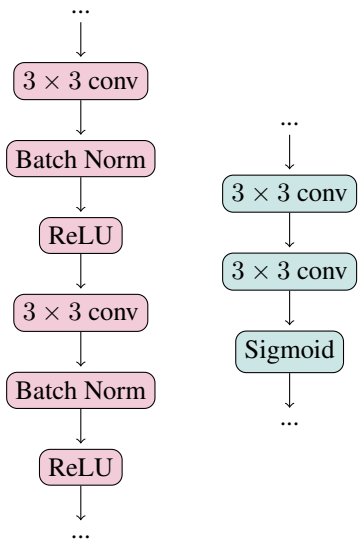

Figure 2. One block of branch B_c (left) and branch B_a (right) without any residual connections. These blocks are the building blocks for our network, and residual connections between blocks will be shown in Figure **??**

Branch B_a is also made up of blocks, which we will refer to as attention blocks. Branch B_a runs parallel to branch B_c and consists of blocks that aim to learn new attention maps with the same dimensions as the features in branch B_c. To do this, each attention block consists of two convolutional layers followed by a sigmoid layer to normalize all weights between 0 and 1. This will allow the output of the attention block to serve as a mask for the features from the corresponding classification block. Details on the layers in each block are shown in Figure 3.

3.5. Model Architecture

Our model takes in a 224×224 image $\mathbf{x}$. Using self-supervised ViT features as described above, we generate an attention map $\mathbf{x}_a$. Image $\mathbf{x}$ is passed to branch B_c and the corresponding attention map $\mathbf{x_a}$ is passed to branch B_a. The attention map after each block is the same size as the features outputted from the corresponding block in branch B_c. This map is used to mask the features.

The strength of the mask is determined by hyperparameter a, where $0 <= a <= 1$. A hyperparameter of $a = 0$ means that no mask will be applied, and the model will be equivalent to a standard ResNet model. A hyperparameter of $a = 1$ means that the mask will be applied with full strength. More formally, the masked residuals are calculated as:

$$\mathcal{F}(\mathbf{x}^{(i)}, \mathbf{x}_a^{(i-1)}) = [\mathbf{x}^{(i-1)} \odot \mathbf{x}_a^{(i-1)} \cdot a] + [\mathbf{x}^{(i)} \cdot a] \quad (3)$$

where $\odot$ represents the Hadamard product, $\mathbf{x}^{(i)}$ represents the features outputted from block i, and $\mathbf{x}_a^{(i)}$ represents the attention map outputted from block i.

To enhance the feature $x^{(i)}$, we simply add the masked residual with the feature:

$$\mathbf{x}^{(i)} = \mathcal{F}(\mathbf{x}^{(i)}, \mathbf{x}_a^{(i)}) + \mathbf{x}^{(i)} \quad (4)$$

Figure **??** provides a visual for how features and residuals are updated. The blocks are arranged following a ResNet8 architecture. After all of the blocks, we have an adaptive average pooling layer and a fully connected layer. The output of the fully connected layer is a score vector of size C, where C is the number of classes in the dataset. The loss used for training and backpropogation is cross entropy loss as given by the following equation:

$$L = -log\left(\frac{exp(y_i)}{\sum_{j \neq i}^{C} exp(y_j)}\right) \quad (5)$$

where y_i is the score of the correct class in the output vector, and y_j are the scores for all other classes.

4. Data

Due to training limitations, we will use the CIFAR-10 and CIFAR-100 [7] datasets as benchmarks for our model. This way, our classification accuracy can be compared to various other computer vision models in order to determine the efficacy of our methods.

Each dataset is already split into 50000 training samples and 10000 test samples. We further split the training samples into 45000 training samples and 5000 validation samples, with an equal amount of validation samples for each class. The resolution of each sample is 32×32

For better model generalization, we perform data augmentation on our training set. We pad each image with 4 pixels on each side, and randomly crop a 32×32 area. We also perform random flipping. Finally, we use a relatively new augmentation method that has proven efficacy: random erasing [13]. The exact method we decided on involves erasing a rectangle between 2% to 33% of the image area and filling it with random noise. This erasing is performed with a 20% chance, since emperical evidence has shown that 20% is a good minimum to have in order to increase model performance.

Finally, after all augmentation is performed, we scale each image up to 224×224 using bicubic interpolation. This is to support training on deeper ResNet models that downsample continuously throughout each residual block.

5. Experiments

5.1. Hyperparameter Tuning

The learning rate and weight decay hyperparameters were first turned with a random search, then refined using a dense grid search. We do this because certain hyperparameters will have a greater effect on the model accuracy, and selecting random groups of hyperparameters will allow for better coverage. [1] As mentioned previously, our AdamW optimizer and AdamW hyperparameters were manually selected following experimental results from Liu et al. [8]. All weights are initialized using a Kaiming normal distribution. [4]

Hyperparameter	value
Optimizer	AdamW
Epochs	100
γ(lr)	0.2
λ (Weight decay)	5e-4
β_1 (AdamW hyperparam)	0.9
β_2 (AdamW hyperparam)	0.999

Table 1. Model hyperparameters used for tuning hyperparameter a

We also experiemented to find optimal batch sizes for each dataset, keeping the other hyperparameters static. Optimal batch sizes are reported in Table 3

Dataset	Batch size
CIFAR-10	32
CIFAR-100	128

Table 2. Batch sizes used for each dataset

Once these hyperparameters were found, we began to tune our custom hyperparameter a, which controls the strength of the attention map applied onto residuals. The

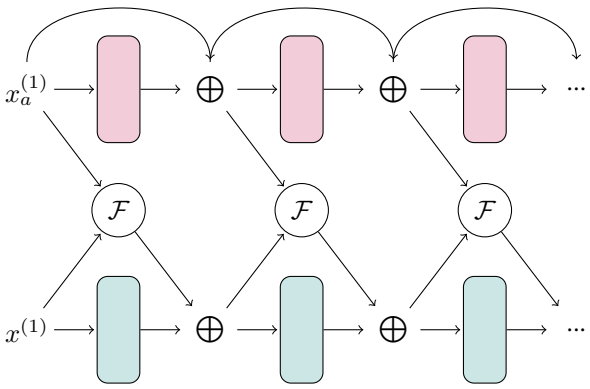

Figure 3. The overall model architecture, and the residual connections between blocks in branch B_c and B_a. All arrows represent identity mappings from the node at the start of the arrow to the node being pointed to by the arrow. To maintain consistency with Figure 3, pink blocks represent classification blocks and teal blocks represent attention blocks. Note that this diagram does not show $x^{(0)}$ and $x_a(0)$ and the respective stem blocks that they are passed through. It also does not show the final pooling and fully connected layers. Only the classification and attention blocks and residual connections are shown.

rest of our discussion will focus on the effects of this hyperparameter, keeping all other ones locked to the values in Table 1.

5.2. Generating Attention Masks

The first step is to generate an attention map $\mathbf{x_a}$ for input to branch B_a. This is done using a Vision Transformer self-supervised with ImageNet-1k. [2]. An input $\mathbf{x}$ is split into 8×8 patches, linearly embedded with learned weights and position encodings, and sent through a Transformer encoder with 12 attention heads. Then, we take the self-attention with the $[CLS]$ token as the query in the final attention layer in the Transformer, right before the final MLP and classification layers. The equations for self-attention can be found in Equation 1

Since there are 12 attention heads, we obtain 12 different attention maps from querying the $[CLS]$ token. Each of these attention weights tends to different parts of the image. However, since we only have one attention branch in our network, these weights are all aggregated into one attention map that will serve as x_a. An examples of this aggregated attention map generated input $\mathbf{x}$ can be found in Figure 4.

Figure 4. Image $\mathbf{x}$ and its corresponding attention map $\mathbf{x}_a$ generated using a self-supervised ViT

5.3. Hyperparameter a

Our datasets are relatively simple, so we are only looking at the top-1 accuracy for different values of a. We found that higher values of a helped to increase accuracy compared to a traditional ResNet8. Values of a at the upper end did not make as much of a difference as compared to the lower end, but still resulted in increased training. Results for top-1 accuracy for each dataset are reported in Table **??**

Top-1 Accuracy (%)		
a	CIFAR-10	CIFAR-100
0	83.3	41.1
0.5	86.6	42.9
1	87.0	43.4

Table 3. Top-1 classification accuracy on CIFAR-10 and CIFAR-100 for mask strength a

The top-1 accuracy for $a = 0$ is equivalent to a standard ResNet8 being run with our data augmentation and training scheme. ResNet8 has been shown to have up to 86% top-1 accuracy on CIFAR-10, though, meaning that our data augmentation and hyperparameters were not optimal. If we were to reach 86% top-1 accuracy for CIFAR-10 using $a = 0$, we can assume that setting $a > 0$ would beat that accuracy. Similar evidence is seen with CIFAR-100; increasing a also increases the top-1 accuracy, but our base model with $a = 0$ does not reach the accuracy of the best CIFAR-100 ResNet.

We believe that the reason for this lower accuracy, despite applying new methods such as using transformer optimizers and patch-based convolutions[8], is that CIFAR-10 and CIFAR-100 are not meant to be trained on a model with input size 224×224. In fact, the originial authors of ResNet

explained in their paper that a different network structure should be used for smaller 32×32 images. [5] As mentioned before, we added a stem block to take in larger input sizes with the goal of training the same model on bigger datasets. Unfortunately due to time constraints we were not able to do so, and also most likely lowered our classification accuracy for CIFAR-10 and CIFAR-100.

Surprisingly, we did not run into too much trouble with overfitting. Before beginning our experiments, we feared that our model was too complex for small datasets like CIFAR-10 and CIFAR-100, and that we would end up with too-complex weights that would not generalize well.

Additionally, training converges faster with higher values of a. Unfortunately we do not have full data on this due to time, but a loss and validation accuracy graph will be shown in the poster and presentation. We believe this is simply due to the fact that we are essentially performing model pre-training and transfer learning. Though only applied to the residual mappings instead of the entire classifier, previous work has shown that residual mappings improve model classification accuracy. Thus, it makes sense that "pre-training" the residual mappings will lead to better classification accuracy and faster convergence.

Given our results, we confidently conclude that our model architecture outperforms a standard ResNet. However, it does not beat state-of-the-art (SOTA) models. If scaled to larger ResNet models that are close to SOTA, it is possible that enhancing residual mappings with attention will reach SOTA accuracy.

6. Conclusion and Future Work

Our experiments show that enhancing residual mappings using attention leads to better classification accuracy. However, our results as of now do not surpass state-of-the-art models, so applications of this model are limited to very niche situations. For example, if someone is working on a classifier without much data and needs a simple ML model, our model can be applied to that situation. This is because our model offers higher accuracy than base ResNet architectures, and the attention mapping can be taken from off-the-shelf models and do not require fine tuning. However, if training resources are unlimited and training data is abundant, larger models will be more effective at classification.

Given more time, we would like to explore some other datasets first. The reason images were scaled to 224×224 is because we had originally planned to train on datasets such as ImageNet, where images are larger, contain more information, and have more classes. We would like to see if our accuracy improvements still apply on larger datasets. Also, the ViT used for attention maps was self-supervised using ImageNet, so it could potentially lead to even greater accuracy improvements when compared to applications that the ViT is not fine-tuned for. Additionally, with these bigger datasets, we wanted to make new classification and attention blocks similar to the bottleneck blocks in ResNets. This would allow us to create deeper networks.

Finally, I would like to explore training a model like this end-to-end, rather than taking a backbone for generating attention maps. Ideally, this would lead to even better maps being generated, and thus giving even better classification accuracy.

7. Contributions and Acknowledgements

You might have noticed that everything is written using plural pronouns. All the work here was done by myself, and I only wrote my paper this way to sound more professional. This journey of trying to create my own architecture has taught me a lot about computer vision. Shoutout to my course instructors Fei-Fei Li and Ehsan Adeli, and also to my project mentor Cem Gokmen.

References

[1] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[2] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[7] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[8] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[10] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. In *Proceedings of the IEEE conference on*

computer vision and pattern recognition, pages 3156–3164, 2017.

[11] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[12] H. Zhao, J. Jia, and V. Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10076–10085, 2020.

[13] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.