Residual Attention Network for Image Classification

1st Hari Nair
Department of Electrical Engineering
Columbia University
New York, NY
hn2388@columbia.edu

2nd Yash Jain
Department of Electrical Engineering
Columbia University
New York, NY
ycj2103@columbia.edu

3rd Ayush Raj
Department of Electrical Engineering
Columbia University
New York, NY
ar4283@columbia.edu

***Abstract*—In this project, we have implemented and analyzed the performance of Residual Attention Networks proposed in the paper "Residual Attention Network for Image Classification" by Fei Wang et. al using CIFAR-10 and CIFAR-100 datasets against traditional convolutional neural network architectures. The Residual Attention Network is a type of Residual Network that incorporates the attention mechanism. Attention-aware feature generating attention modules which adaptively change with the network depth are stacked to form a Residual Attention Network. While implementing the paper, we have conducted extensive analysis using the CIFAR-10 and CIFAR-100 datasets to inspect and validate the efficacy of the different modules that are used in developing the Residual Attention Networks. Through our experiments we conclude that Residual Attention Networks with 56 weighted layers in the trunk branch outperforms other models that were implemented during experimentation.**

***Index Terms*—attention, residual network, CIFAR, image classification**

I. Introduction

Attention is among the most prominent ideas in the world of deep learning. In deep learning, the Attention mechanism is based on the concept of focusing or paying greater attention to certain specific factors when processing the data. In computer vision, attention can make our networks more interpretable, robust and improve performance. We can define attention as a technique through which features can be weighed by their level of importance and use these weights to attain a goal. We can say that Attention is a part of a network's architecture which is responsible for quantifying the interdependence between (a) input and output elements (called General Attention) and (b) Within the input elements (called Self-Attention).

The concept of attention was born to address problems with seq2seq models in machine translations specifically. The concept was very successful and since then the concept has been extended to computer vision as well. In computer vision, unlike in RNNs, instead of having dependency along the time domain, the dependency is along the spatial domain. In images, being able to understand and learn dependencies beyond the limited receptive field of a convolutional filter becomes important in improving performance and allowing the models to build a wider intuition. For example, in images, the same texture may be present in multiple areas of an image, multiple disjoint semantic cues may give hints about the overall classification of an image or an object may have multiple complex and obscured parts throughout an image.

In the recent years there has been a huge progress in the fields of image processing, image recognition and computer vision and in the applications of deep learning in these fields. Neural Networks [7] have become deeper and more complex. This is done because more layers in neural networks help in extracting more complex features present in the images thus making the model more robust. However, as a neural network grows deeper, it becomes more and more difficult to train because of the vanishing gradient problem. This occurs because it becomes increasingly difficult for the layers to propagate the information from the initial layers through the middle layers and this leads to a loss of information. This generally causes a drop in accuracy and increase in loss, hence, defeating the purpose of using a deep neural network.

This problem is solved using Deep Residual Networks or ResNets. In ResNets, the intermediate layers are bypassed and the shallow/initial layers are connected directly with the deeper layers. In this way, the information which was previously getting lost is now being passed directly as identity function. These connections are called "shortcuts" or "skip connections" and in this, the result from a neuron is added directly to the corresponding neuron of a deeper layer. These layers, with a skipped connection, together are called 'Residual Blocks'. ResNets are built by using these 'Residual Blocks' and stacking them together to form a deep network. Using this method, it has been observed that it is possible to successfully train very deep networks, even networks with over 150 layers.

Residual Attention Network (RAN) is a convolutional neural network which combines the concept of attention with the residual blocks of ResNets. It utilizes 'skip-connections' to jump over intermediate layers. Its main feature is the 'attention modules' which are stacked one after the other. These attention modules produce 'attention-aware' features that change adaptively as we move to deeper layers in the network. The attention module comprises of two different branches. They are the trunk branch and the mask branch. The Trunk Branch uses Residual Units to perform feature processing and the Mask Branch uses bottom-up and top-down

steps for improving the trunk branch features. The features that are extracted from these branches are then incorporated together using the Attention Residual Learning formula described in the original research paper. Subsequently, this is used for training deep Residual Attention Networks with the aim of incorporating scalability without affecting the network performance. So, as we increase the number of Attention modules, the performance of the neural network increases consistently.

II. Literature Review

The residual attention network [1] is developed by combining an array of attention modules one after the other. Each of these attention units gets divided in two separate branches namely, trunk branch and mask branch. The trunk branch is used for feature processing and can make the network adaptable to any state-of-the-art neural network architectures. If we give input x to the trunk branch, we get T(x) as the output. Now this T(x) is given to the mask branch to learn a mask M(x) with the same size that weighs the output features T(x). M(x) is learned by using the bottom-up top-down branch of the soft mask structure. The final output of a particular attention module is given by equation 1. [1]

$$H_{i,c} = M_{i,c}(x) * T_{i,c}(x) \quad (1)$$

where i spans across all spatial positions and c ϵ Number of channels.
The mask M(x) can be used to update the gradient during back propagation. The gradient mask for the input feature is:

$$\frac{\partial M(x,\theta)T(x,\phi)}{\partial \phi} = M(x,\theta)\frac{\partial T(x,\phi)}{\partial \phi} \quad (2)$$

Fei Wang et al [1] proposed a new way of residual attention learning to mitigate the following problems in the architecture. The dot product with the mask degrades the value of features in deep layers for the above architecture. Also, the soft mask branch can break a few properties of the trunk branch. Therefore, Fei Wang et al [1] modified the output H of the attention module as:

$$H_{i,c} = (1 + M_{i,c}(x)) * F_{i,c}(x) \quad (3)$$

where M $\in \{0, 1\}$.

When M is a 0, H will be F i.e the original features. The mask M(x) acts as a feature selector that can enhance good features and mitigate noise from the trunk features.

The output features are softly weighed to enhance the trunk branch features with the bottom-up top-down structure by the soft mask branch. The bottom-up step employs down-sampling (max-pooling) to collect all the global features of an entire image. The top-down step employs up-sampling (interpolation) to keep the output image size equal to the input image size. Once both these steps are completed, the extracted features are amalgamated using Attention Residual Learning. This helps in scaling up the network i.e adding more layers without a decrease in performance. Therefore, adding several attention modules improves performance as different attentions are captured extensively.

Layer	Output Size	Attention-56	Attention-92
Conv1	112×112	7×7, 64, stride 2	
Max pooling	56×56	3×3 stride 2	
Residual Unit	56×56	$\begin{pmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{pmatrix} \times 1$	
Attention Module	56×56	Attention ×1	Attention ×1
Residual Unit	28×28	$\begin{pmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{pmatrix} \times 1$	
Attention Module	28×28	Attention ×1	Attention ×2
Residual Unit	14×14	$\begin{pmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{pmatrix} \times 1$	
Attention Module	14×14	Attention ×1	Attention ×3
Residual Unit	7×7	$\begin{pmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{pmatrix} \times 3$	
Average pooling	1×1	7×7 stride 1	
FC,Softmax	1000		
params$\times 10^6$		31.9	51.3
FLOPs$\times 10^9$		6.2	10.4
Trunk depth		56	92

Fig. 1. Typical Architecture of a Residual Attention Network [1]

Fei Wang et al [1] assessed the performance of this proposed Residual attention network on 3 different datasets CIFAR-10, CIFAR-100 [3] and ImageNet. [4]

Fei Wang et al [1] experimented on the CIFAR datasets using naive attention learning as the base model. It employs attention units where there is a scalar product of the features with the soft mask. The total number of weighted layers in the trunk branch can be given by 36m + 20 where m = number of attention modules. Fei Wang et al [1] experimented using Attention-56 (m = 1), Attention-92 (m = 2), Attention-128 (m = 3), Attention-164 (m = 4) for two different methods namely Naive Attention learning and Attention Residual Learning.

Network	ARL (Top-1 err. %)	NAL (Top-1 err.%)
Attention-56	**5.52**	5.89
Attention-92	**4.99**	5.35
Attention-128	**4.44**	5.57
Attention-164	**4.31**	7.18

Fig. 2. Classification error on CIFAR-10 [1]

The results obtained by running this experiment clearly indicated that Attention Residual Learning clearly outperformed Naive Attention Learning. The Top-1 percent error for Attention residual learning is less when compared to the Naive Attention Learning. The lowest Top-1 percent error is

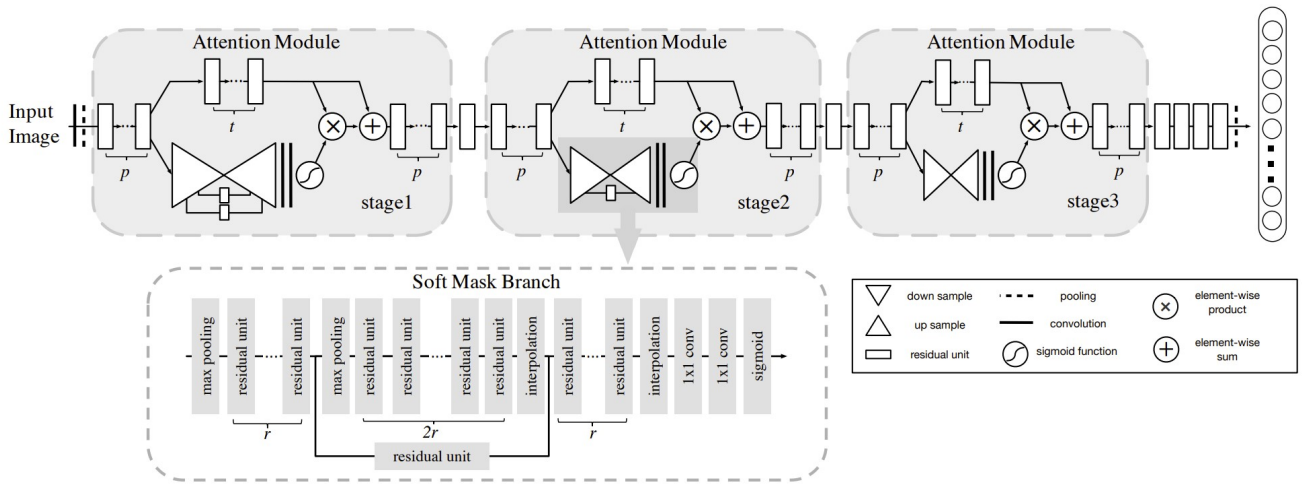

Fig. 3. Typical Architecture of a Residual Attention Network [1]

achieved by the Attention-164 architecture when trained using Attention Residual Learning. The summary of the results of the experiments performed by Fei Wang et al [1] on CIFAR-10 can be seen in Fig 2.

III. Implementation

A. Description of the Problem

This work evaluates the efficacy of Residual Attention Networks [1] for the task of image classification. For this, we implement attention residual learning mechanisms proposed in the original paper [1] which include Attention-56 (1 Attention module), Attention-92 (2 Attention Modules) and Naive Attention Learning mechanisms. Furthermore, experimentation with hyperparameters and modification of the proposed architectures is performed over the original paper.

The performance of these methods are also compared with traditional convolutional neural networks and a deep residual network ResNet-152. The implemented methods are evaluated based on their performance on the CIFAR-10 and CIFAR-100 datasets. These datasets contain 60k images, each 32 x 32 RGB images that belong to 10 and 100 different classes respectively. They consist of 50k images in the training data and 10k images in the test dataset.

B. Description of student's implementation

We implement a Naive Convolutional Neural Network, A Deep Residual Network (ResNet-152) [5], Naive Attention Learning, Attention-56 and Attention-92 architectures. The performance of all these architectures is evaluated on both CIFAR-10 and CIFAR-100 datasets. For the training process, the images are standardized during the preprocessing stage and data augmentation techniques [2] are used to train models with better generalization capacities. Since the images are 32 x 32, the images are padded to maintain shape while passing through the convolutional blocks and avoid loss of information during this process. Categorical crossentropy is used to measure the loss and Stochastic Gradient Descent with momentum and Adam are experimented with as the optimizers during training. [6] We experimented with different values of learning rate to achieve fast convergence. The following pointers give insight into the implementation and results achieved in different architectures.

All experiments have been performed on a single NVIDIA Tesla T4 GPU. Each architecture was trained on this single GPU for 100 iterations to observe the trends in the performance of each architecture.

IV. Results

A. Naive CNN

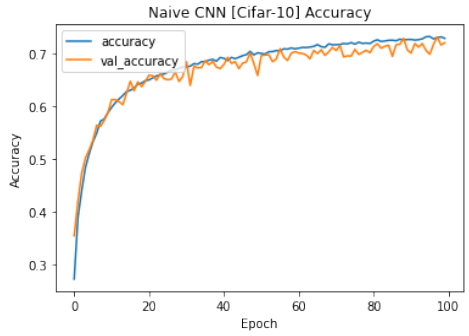

Fig. 4. Training Naive CNN on CIFAR-10

Training on CIFAR-10: (Refer Fig. 4)
Accuracy: 72.07 % (validation), 73 %(train)
Training time per epoch: 20 seconds
Approximate total training time: = 35 minutes

Training on CIFAR-100: (Refer Fig. 5)
Accuracy: 32.13 % (validation), 39.59 %(train)
Training time per epoch: 20 seconds

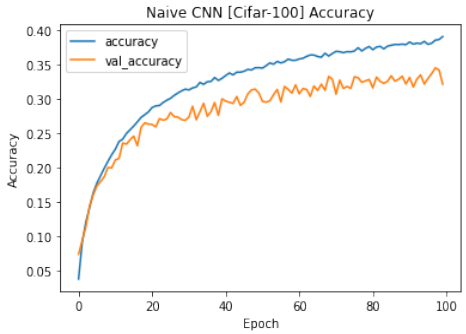

Fig. 5. Training Naive CNN on CIFAR-100

Approximate total training time: = 35 minutes

The accuracy for training a naive CNN is better for CIFAR-10 dataset compare to CIFAR-100. This can be attributed to the fact that in CIFAR-100 there are very few samples of a particular class than in CIFAR-10.

B. ResNet-152

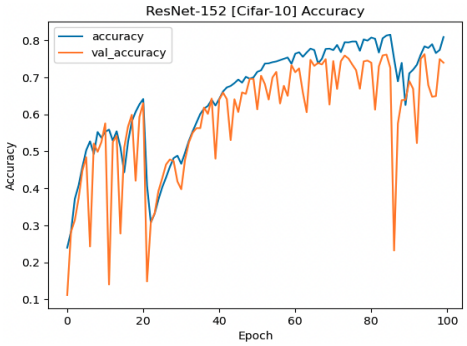

Fig. 6. Training ResNet-152 on CIFAR-10

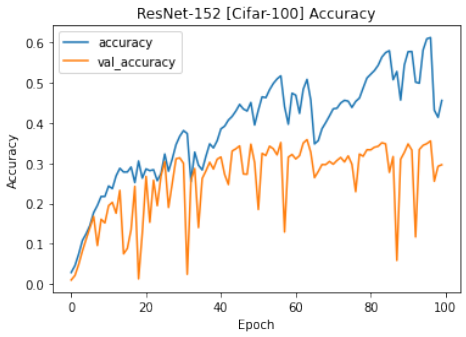

Fig. 7. Training ResNet-152 on CIFAR-100

Training on CIFAR-10: (Refer Fig. 6)
Accuracy: 60.33 % (validation), 63 %(train)
Training time per epoch: 93 seconds
Approximate total training time: = 2 hours 35 minutes

Training on CIFAR-100: (Refer Fig. 7)
Accuracy: 29.62 % (validation), 45.38 %(train)
Training time per epoch: 94 seconds
Approximate total training time: = 2 hours 30 minutes

C. Naive Attention Learning

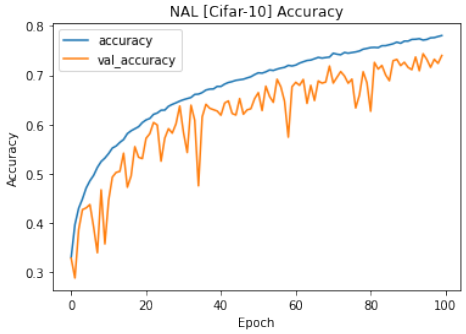

Fig. 8. Training NAL on CIFAR-10

Training on CIFAR-10: (Refer Fig. 8)
Accuracy: 74 % (val), 78.11 %(train)
Training time per epoch: 50 seconds
Approximate total training time: = 85 minutes

D. Attention-56

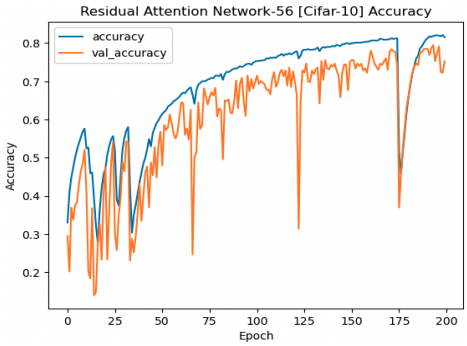

Fig. 9. Training Attention-56 on CIFAR-10

Training on CIFAR-10: (Refer Fig. 9)
Accuracy: 75.27 % (validation), 81.57 %(train)
Training time per epoch: 35 seconds
Approximate total training time: = 58 minutes

Training on CIFAR-100: (Refer Fig. 10)
Accuracy: 44.6 % (validation), 56.29 %(train)
Training time per epoch: 36 seconds
Approximate total training time: = 60 minutes

E. Attention-92

Training on CIFAR-10: (Refer Fig. 11)
Accuracy: 72.8 % (validation), 77.68 %(train)

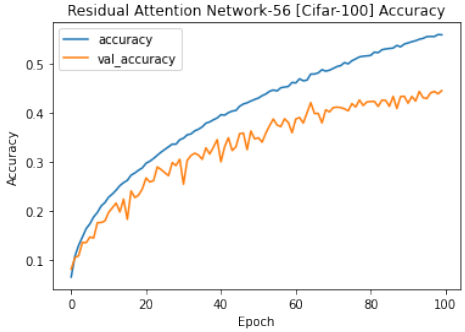

Fig. 10. Training Attention-56 on CIFAR-100

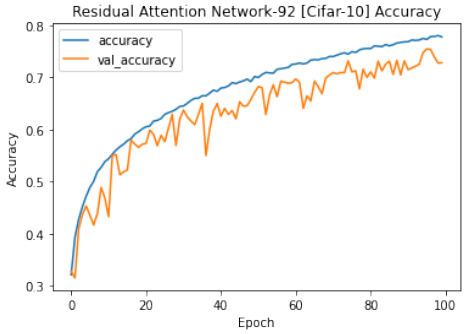

Fig. 11. Training Attention-92 on CIFAR-10

Training time per epoch: 51 seconds
Approximate total training time: = 85 minutes
Training on CIFAR-100: (Refer Fig. 12)
Accuracy: 40.01 % % (val), 50.72 %(train)
Training time per epoch: 49 seconds
Approximate total training time: = 82 minutes

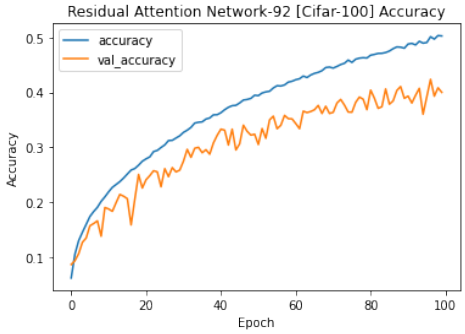

Fig. 12. Training Attention-92 on CIFAR-10

The test accuracy of all models on the test sets of CIFAR-10 and CIFAR-100 can be seen in Fig. 13.

V. Discusssion and Conclusion

From the results obtained in the above experiments, we can observe that the Attention-56 model, i.e., the model with 1 attention module in one stage outperforms all other models in terms of accuracy for classifying the images

Architecture	CIFAR-10	CIFAR-100
Naive CNN	76.9%	37.01%
ResNet-152	66.67%	31.07%
Naive Attention Learning (NAL)	76.38%	-
Attention-56	77.67%	47.03%
Attention-92	70.93%	42%

Fig. 13. Model Accuracy on Test Data for CIFAR-10 and CIFAR-100

contained in CIFAR-10 and CIFAR-100 datasets. Even though, Attention-92 is a bigger and more complex model than Attention-56, it does not give a good validation accuracy for both the datasets. One of the reasons for this abnormal behaviour could be that Attention-92 model tends to overfit on the dataset and hence have a lower validation accuracy and a higher training accuracy.

Additionally, the accuracy for these various models can be potentially improved by training for a longer duration. We have trained these models for 100 epochs due to time constraints. However, we believe that with availability of time and computational resources, these architectures can reach the performance potential as shown by Fei Wang et al. [1] by training for 160k epochs.

One of the major problems that we faced during the implementation of Residual Attention Networks was the unavailability of sufficient computational resources to train our models. Although the validation accuracy is close to the training accuracy, the fluctuations signify that the model has high variance which makes it challenging to tune the Network Architecture hyperparameters on NVIDIA Tesla T4 GPU.

References

[1] F. Wang et al., "Residual Attention Network for Image Classification," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6450-6458, doi: 10.1109/CVPR.2017.683.
[2] Keras https://keras.io/. 2015.
[3] CIFAR-10 and CIFAR-100 dataset https://www.cs.toronto.edu/ kriz/cifar.html
[4] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.
[5] He, Kaiming Zhang, Xiangyu Ren, Shaoqing Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.
[6] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." ArXiv abs/1609.04747 (2016)
[7] Marius, Popescu Balas, Valentina Perescu-Popescu, Liliana Mastorakis, Nikos. (2009). Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems. 8.