Residual Attention Network for Image Classification

Vasanth Margabandhu – vm2656 Aman Anand - aa4821 Bhavin Dhedhi - bdd2115

***Abstract*— In this project, we try to reproduce the results from the paper titled Residual Attention Network for Image Classification [4]. The network in question is generated by stacking series of Attention Modules. These modules generate attention-aware features, which change adaptively as a function of the depth in the network. A brief introduction into the concept of attention for learning and residual networks is developed and motivated for the image classification task. Some relevant literature which inspired the work of the authors is also mentioned. The structure and functioning of each component of the Residual Attention Network is described, and the authors' implementations are listed for comparison. The Residual Attention Network is tested on benchmark datasets, namely CIFAR-10, CIFAR-100 and ImageNet. The code for the attention module is reproduced and the performance our own implementation is compared with the results of the paper, and the shortcomings are documented comprehensively.**

Keywords—Attention Residual Learning, CIFAR-10, CIFAR-100, ImageNet, Keras.

I. INTRODUCTION

Attention as a concept has not been comprehensively explored in the field of image classification. Not many implementations have reported stat-of-the-art results. The nuances of attention have been explored in detail however, in preceding research work. Attention is a mechanism that can both help to focus on the right regions, as well as improve the representations of distinct objects within a given region. The Residual Attention Network leverages the image classification capability of deep neural networks and enhances it by incorporating the attention mechanism. The authors report that stacking Attention Modules consecutively improves the performance, since different masks highlight different features of an image, and more and more attention-aware features are discovered. Inspired by bottom-up top-down feedforward structures applied to other areas like pose estimation and image segmentation, the authors incorporate this into the Attention Modules, allowing for end-to-end training in a single feedforward process.

II. BACKGROUND WORK

The motivation for the attention mechanism comes from human perception, which utilizes information from the top level to direct the bottom-up feedforward process. Residual networks make use of skip connections in the layers, which allow paths for the backpropagation of the gradient. This nullifies the possibility of vanishing gradients with increasing depth of neural networks, allowing for progressively deeper neural networks to learn and generalize efficiently. To leverage the ability of stacked Attention Modules to capture different attentions, they are coupled with Residual learning to give Residual Attention Networks. The attention mechanism has found widespread application in text processing applications using recurrent neural networks and long short-term memory, where the attention mechanism captures both the query context and the query itself. In the context of image classification, the authors' work involves the repeated splitting and joining of images. To allow for this, the network possess sufficient depth, which is taken care of by residual learning.

Top-down attention mechanisms are not nascent to the field of image classification, with similar approaches in region proposal, image pyramids, sliding windows and control gates well documented.

Region proposal method uses top information in the form of proposed regions, which are sections of the image, and uses these regions for learning features for image classification. Similarly, control gates, which are used in LSTMs as well as in image classification methods with attention, also incorporate information from the top in the training process.

Methods like Dropout, Stochastic Depth and Batch Normalization also inspire the work of the authors who aim to train a "very deep" architecture without compromising on test accuracy.

Their work is also inspired by recent application of soft attention in the spatial transformer module, wherein the module uses information from the input image to generate an affine transformation, which on application to the input image generates an attended region. This process is repeated and achieves state-of-the-art performance in image segmentation. Soft attention with residual learning also performed admirably in human pose estimation, where the training of an appropriately named stacked hourglass network is influenced by both top level and bottom level information. This can be achieved via skip connections between the layers capturing top level (global) and bottom level (local or pixel level) features.

III. Network Architecture

The Residual Attention Network consists of a series of Attention Modules stacked on top of each other. Each module consists of two branches, namely the Mask branch and the Trunk branch. The Trunk branch performs feature selection, while the Mask branch is used for downsampling to create a soft mask. At the end of the module, the outputs from both branches are combined and fed forward to the next module. This process is repeated several times.

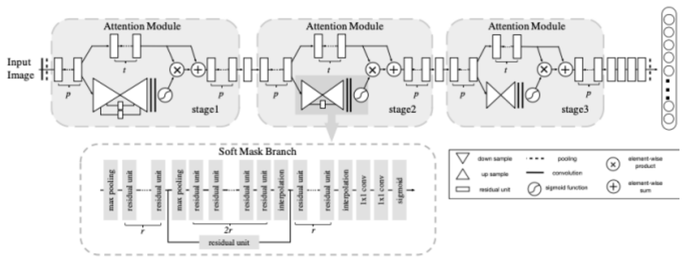


Fig. 1. Example architecture of Residual Attention Network for ImageNet as shown in the paper.

In Fig. 1, the *p* stands for the number of pre-processing Residual units; *t* denotes the number of Residual units in the trunk branch and *r* denotes t the number of Residual units between the mask branch and the pooling layer.

Trunk Branch: It is the upper branch of the attention module. As mentioned above, its purpose is extraction of the attention aware features. For any given input *x*, let its output be denoted by *T(x).*

Mask Branch: It uses bottom-up top-down structure to learn the mask M(x). This M(x) is used as control gates similar to those in an LSTM or the Highway Network mentioned in the relevant literature section.

In toto, the output of the Attention module can be expressed as:

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

In the above equation, i stands for all locations in space, while c is the running index denoting the current channel.

During backpropagation:

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

Here, θ's are mask branch parameters and φ's are trunk branch parameters. This allows for the attention mask to also serve the purpose of a filter for gradient updates during the backward pass. The structure of the mask branch is such that it doesn't allow noisy gradients from incorrect labels to backpropagate and update the parameters of the trunk branch.

Attention Residual Learning: The authors state that stacking of Attention modules result in an improvement in performance; however, there is a caveat. Naively stacking layers may be detrimental, as the consecutive dot products between layers with *M(x)* taking a value between 0 and 1 causes the vanishing of the feature value with increasing depth. To circumvent this problem, the soft mask is modeled as a residual connection, described by the following equation:

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

As a result, the original features *F(x)* of the trunk branch are preserved, and stacking consecutive modules improves the performance as the feature maps become progressively more refined and clearer with an increase in the number of layers.

Soft Mask Branch:

This branch is inspired by the stacked hourglass network for human pose estimation, in that it combines global information obtained from a feedforward pass step and local pixel-level information from the top-down feedback steps. Max pooling is applied repeatedly on the input image to downsample it until the lowest resolution is reached. Following this, linear interpolation layers upsample the output of the max pooling layers. Skip connections between these parts of the network allow both top level (global) and bottom level (local) data to influence the training. Finally, a sigmoid layer is applied to normalize the outputs to lie in between 0 and 1 after two 1x1 convolutions.

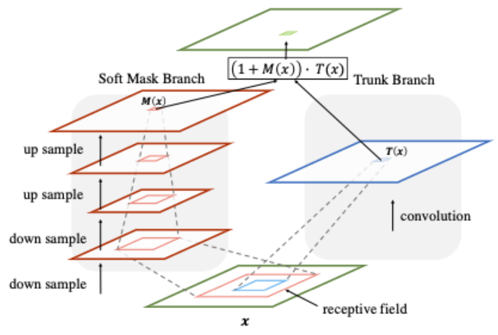


Fig. 2.

Soft mask branch and trunk branch receptive field comparison.

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

Fig. 3. Overall Architecture of the Residual Attention Network

IV. Original Implementation

CIFAR-10 & CIFAR-100:

Both datasets have the same number of training and test images of same resolution. ResNet is used as the base model. Zero-padding is done with 4 pixels on each side. A random 32x32 section of this padded image is extracted and per-pixel average value across the RGB channels is subtracted. The optimizer used is SGD with Nesterov momentum. Batch size is 64. Weight decay is set to 0.0001, momentum to 0.9 and initial learning rate to 0.1. It is scheduled to be divided by 10 after 64k and 96k iterations.

Experiments are conducted with varying number of Attention modules (varying from 1 to 4). The baseline for evaluating the performance of the Residual Attention Network is the Naively stacked Attention Network, since this kind of network is unprecedented and has no suitable benchmarks to compare performance with.

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. 4. Naïve Learning vs Residual Learning comparison.

From Fig.4, we can see that Attention Residual Learning outperforms Naïve learning for different configurations of the network.

Noise Level	ResNet-164 err. (%)	Attention-92 err. (%)
10%	5.93	5.15
30%	6.61	5.79
50%	8.35	7.27
70%	17.21	15.75

Fig.5. Test error with different noise levels

Fig. 5 shows that the Network outperforms the ResNet-164 for varying levels of noise. In this context, noise denotes the percentage of incorrect labels.

Network	params×10^6	CIFAR-10	CIFAR-100
ResNet-164 [11]	1.7	5.46	24.33
ResNet-1001 [11]	10.3	4.64	22.71
WRN-16-8 [39]	11.0	4.81	22.07
WRN-28-10 [39]	36.5	4.17	20.50
Attention-92	1.9	4.99	21.71
Attention-236	5.1	4.14	21.16
Attention-452†	8.6	**3.90**	**20.45**

Fig.6. Comparison with state-of-the-art models

The Attention-452 outperforms all the state-of-the-art networks. From Fig.6. it can be inferred that performance can be improved while simultaneously reducing the number of parameters of the network.

ImageNet:

Scale and aspect ratio augmentation are applied to the input image. Horizontal flip and standard color augmentation are also employed. A random 224x224 section of the image is extracted from the augmented image. Following this, the per-pixel value across the channels is scaled to lie between 0 and 1, the mean value subtracted as in the previous implementation, and the variance divided. As before, optimizer is SGD with initial learning rate 0.1 and momentum 0.9. Learning rate is scheduled to be divided by 10 after 200k, 400k and 500k iterations. Training is done for 530k iterations.

Network	params×10^6	FLOPs×10^9	Test Size	Top-1 err. (%)	Top-5 err. (%)
ResNet-152 [10]	60.2	11.3	224 × 224	22.16	6.16
Attention-56	31.9	6.3	224 × 224	**21.76**	**5.9**
ResNeXt-101 [36]	44.5	7.8	224 × 224	21.2	5.6
AttentionNeXt-56	31.9	6.3	224 × 224	**21.2**	**5.6**
Inception-ResNet-v1 [32]	-	-	299 × 299	21.3	5.5
AttentionInception-56	31.9	6.3	299 × 299	**20.36**	**5.29**
ResNet-200 [11]	64.7	15.0	320 × 320	20.1	4.8
Inception-ResNet-v2	-	-	299 × 299	19.9	4.9
Attention-92	51.3	10.4	320 × 320	**19.5**	**4.8**

Fig.7. Comparison of validation error with state-of-the-art classifiers.

The Residual Attention Network outperforms the classifiers having similar or greater number of parameters on a complex dataset like ImageNet, thereby proving its efficacy.

V. Student's Implementation

This section of the report discusses our (the students) implementation of the techniques discussed in the paper. For all the models discussed in this part of the report, we referred to [1], [2] and [3] for the network architecture implementation. The flowchart for our implementation is exactly the same as in Figures 1 and 2.

CIFAR-100 Dataset

We create attention modules for image classification based on the implementation and ideas discussed in the original paper along with some changes. We create two models for the CIFAR-100 dataset (one with batch size of 32 and another with batch size of 64). For the implementation with batch size of 64, we augment the training dataset with vertical flip, rotation, width shift, height shift and zoom range which was not done in the original implementation. We also add dropout of 0.25 and use the Adam Optimizer instead of SGD as implemented in the original paper. Rest of the implementation remains same. We use same number of iterations (160k) as described in the paper and similar learning rate decay. The hyperparameters (p, t and r) have also been kept same as in the original paper. With this implementation, we were able to achieve a training accuracy of 58.89% and a validation accuracy of 48.35%.

To further experiment with the ideas mentioned in the paper, we instantiated a second model for the CIFAR-100 dataset with different configurations. In particular, we reduced the batch size from 64 to 32, did not perform dataset augmentation and learning rate decay and trained the model for 31250 iterations. This resulted in a significant increase in the training accuracy (of 94.55%) however the validation accuracy remained low at 34.01%. We suspect this to be due to the model overfitting on the training data which could be reduced by experimenting with regularization and different dropout strategies.

CIFAR-10 Dataset

Similar to our implementation of attention modules for the CIFAR-100 dataset, we instantiated two different models for the CIFAR-10 dataset.

In the first implementation, we kept the number of iterations and learning rate decay to be same as given in the original paper but used a different optimizer (Adam instead of SGD). We also augment the image in the same way as we did for the CIFAR-100 dataset described in the previous section. With this configuration, we were able to achieve a training accuracy of 77.28% and a validation accuracy of 77.34%.

For the second implementation, as done for the CIFAR-100 dataset, we reduce the batch size to 32 and train the model without data augmentation and learning rate decay for 31250 iterations. With this implementation, we got a training accuracy of 89.63% and validation accuracy of 58.35%.

Tiny-ImageNet Dataset

The original paper had implemented the attention network with the ImageNet dataset. However due to constraints on the memory usage, we implemented our attention network on the Tiny-ImageNet dataset. The network structures remains same as the first network(s) used for the CIFAR-100 and CIFAR-10. However, in this case we trained the network for 1024 epochs. For the ImageNet dataset however, even after 145 mins of training, we were able to achieve a training accuracy of 12.86% and a validation accuracy of 13.19% indicating less than optimal performance of the network as compared to the original implementation. We believe this to be the case due to changes in the data pre-processing stages of our implementation where we tried to augment the dataset with vertical flip, zoom, height and width shift instead of standard colour augmentation as described in the original paper.

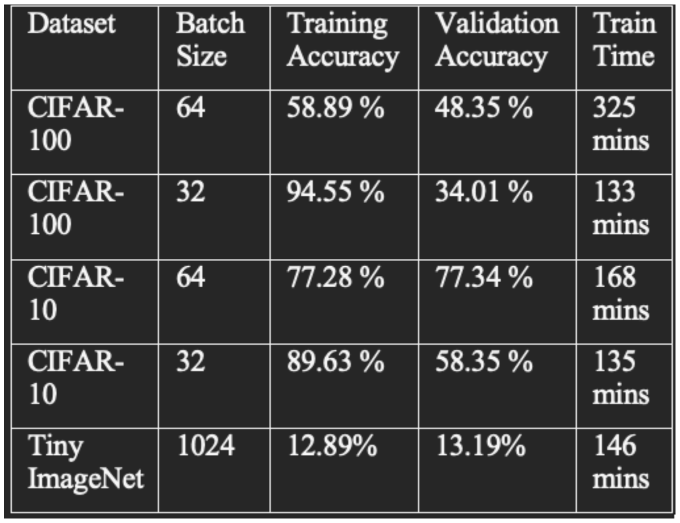
VI. DISCUSSION OF RESULTS

In this section, we discuss the results of our implementation, compare them to the original implementation. For the CIFAR-100 dataset, by following the original implementation with a minor change in the choice of optimizer (Adam instead of SGD), we were able to achieve a training accuracy of 58.89% and a validation accuracy of 48.35%. In comparison to our implementation, the original paper achieved an accuracy of 78.29%. In order to further improve our accuracy, we reduced the batch size and increase the number of epochs which resulted in an increase in training accuracy to 94.55% ; the validation accuracy however remained low at 34.01%. We attribute this (possibly) to change in the optimizer along with data pre-processing involving padding and random crop of the image as done in the original implementation.

For the CIFAR-10 dataset, we again performed two implementations; one with similar batch size and learning rate schedule as in the original implementation and another with reduced batch size and larger number of epochs. In the case of the former implementation, we achieved a training accuracy of 77.28% and a validation accuracy of 77.34%. In the case of the latter implementation where we deviated a bit from the original paper, we were able to achieve a training accuracy of 89.63% and a validation accuracy of 58.35%. As in the case of the CIFAR-100 implementation, we suspect this to be the case due to non-padding and cropping of the image as performed in the original paper.

The third dataset which the performed tested with the attention module was the ImageNet dataset. However due to constraints on the memory requirement, we tested our implementation with the Tiny-ImageNet dataset. In this case, the performance of our model was less than optimal with the training accuracy at 12.89% and the test accuracy at 13.19%. We attribute this to lack of colored-image augmentation in our implementation which was performed in the original paper; we instead chose to augment the dataset with vertical flip, height and width shift and zoom.

The sub-optimal performance of our model on the Tiny ImageNet dataset indicates that colored augmentation performed in the original paper is a major pre-processing step that can have a lot of impact on the final training and validation accuracy.

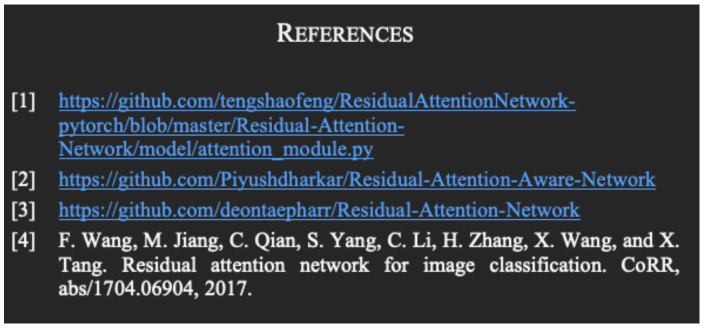


Dataset	Batch Size	Training Accuracy	Validation Accuracy	Train Time
CIFAR-100	64	58.89 %	48.35 %	325 mins
CIFAR-100	32	94.55 %	34.01 %	133 mins
CIFAR-10	64	77.28 %	77.34 %	168 mins
CIFAR-10	32	89.63 %	58.35 %	135 mins
Tiny ImageNet	1024	12.89%	13.19%	146 mins

Table 1: Comparison of Results.

ACKNOWLEDGMENT

We would like to extend our gratitude to Professor Zoran Kostic and all the TAs of the ECBM4040 course for their support and guidance throughout the course of the semester.



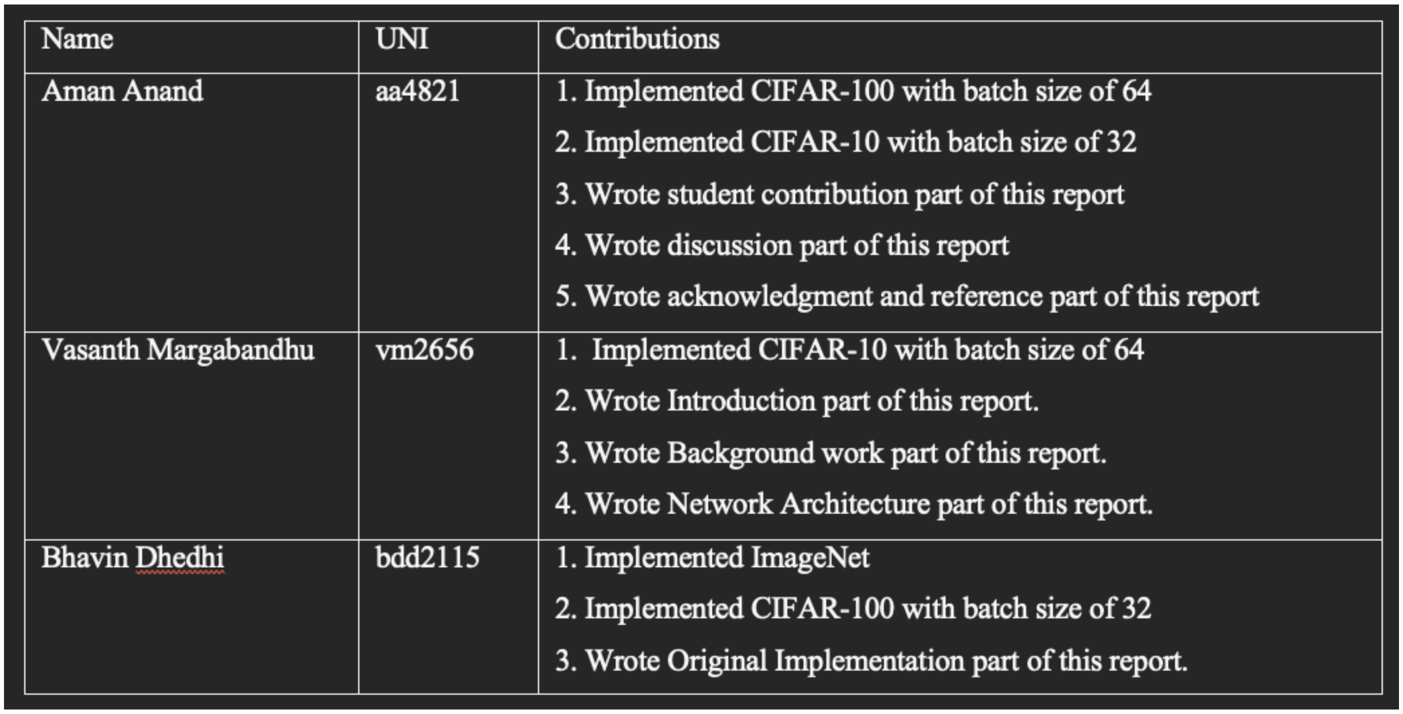
REFERENCES

[1] https://github.com/tengshaofeng/ResidualAttentionNetwork-pytorch/blob/master/Residual-Attention-Network/model/attention_module.py

[2] https://github.com/Piyushdharkar/Residual-Attention-Aware-Network

[3] https://github.com/deontaepharr/Residual-Attention-Network

[4] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. CoRR, abs/1704.06904, 2017.



Name	UNI	Contributions
Aman Anand	aa4821	1. Implemented CIFAR-100 with batch size of 64 2. Implemented CIFAR-10 with batch size of 32 3. Wrote student contribution part of this report 4. Wrote discussion part of this report 5. Wrote acknowledgment and reference part of this report
Vasanth Margabandhu	vm2656	1. Implemented CIFAR-10 with batch size of 64 2. Wrote Introduction part of this report. 3. Wrote Background work part of this report. 4. Wrote Network Architecture part of this report.
Bhavin Dhedhi	bdd2115	1. Implemented ImageNet 2. Implemented CIFAR-100 with batch size of 32 3. Wrote Original Implementation part of this report.

Table 2. Individual student’s contribution.