Design and Analysis of Residual Attention Networks for Image Classification

Deepak Dwarakanath
School of Engineering and Applied Sciences
Columbia University
New York, NY
Email: dd2676@columbia.edu

Suman Kalyan Adari
School of Engineering and Applied Sciences
Columbia University
New York, NY
Email: sa3976@columbia.edu

Abstract—**In this paper, we are going to analyze the Residual Attention Network, which utilizes both residual layers and attention module to generate a very advanced representation of images. This model's use of stacked attention modules that also utilize residual connections allow it to utilize the Attention property, which allows a neural network to adaptively focus on key features in the dataset images [1]. The attention modules analyzed in this paper utilize a top-down, bottom-up approach that use downsampling and upsampling in order to generate these attention aware features. In addition to this, the model utilizes the previously designed residual modules which utilize skip connections to pass forward the results from earlier layers allowing it to preserve more information and generate a more thorough representation of image data [2]. The residual attention network can be hundreds of layers deep, which is necessary for generating a precise representation of complex image sets. The multifaceted nature of the Residual Attention Network allow it to classify image sets with great accuracy. The original paper trained on multiple datasets, include the massive ImageNet repository as well as various CIFAR datasets. Our implementation of this model will train and test on the commonly used CIFAR-10 dataset. We were able to achieve good validation accuracy and loss, typically in the 80-90 % range, while training different variants of the model.**

1. Introduction

Deep convolutional neural networks have proven to be a groundbreaking method of classifying images. Traditionally these use convolution/correlation which applies filter windows that iterate over in image in two dimensions, and stack one or more filters in the 3rd dimension, generating an intricate interpretation of the images pixels. The choice of convolution filters can have the effect of identifying vertical and horizontal edges, and contrasts in the images' color schemes. Most convolutional models used today are often extremely deep, anywhere from tens to hundreds of layers in feedforward sequence. More cutting-edge models also utilize horizontal stacking, like the Inception module. Ultimately the purpose of designing and studying these different models is to find the best way to classify images.

The concept of Attention itself is merely the idea of focusing on a distinct segment or characteristic of some piece of data, whether it's an image or a document, often varying over time (like in sequential language/text processing, like the Transformer model). This study is considered one of the first to use attention in a deep feed-forward convolutional neural network. The intention of the authors was to provide an adaptive, ”mixed” attention mechanism over several very deep segments [1]. Stacking these attention modules over many layers can produce a distinctive ”masking” effect that identifies finer details in an image. However, ”naively” stacking them can lead to a performance drop because you can encounter the vanishing gradient problem. So there are modifications that are made that allow the model to capture more preceding information, by using the principles of residual connections. Also the attention module uses down-sampling and up-sampling to produce what the authors describe as a top-down, bottom-up approach to generate a soft attention mask on the images. In practice this has an effect of diminishing certain features in order to focus on other features. This top-down, bottom-up approach is evidently based on human visual perception [3].

2. Residual Attention Networks

2.1. Methodology of Original Paper

The residual attention network (RAN) as described by Wang et al. [1] integrates two important concepts: stacked attention blocks, and residual connections. Each attention block consists of the input splitting into two channels: the trunk branch and the mask branch. The layers in the trunk branch can be comprised of any convolutional neural network (CNN) structure. In this case, the authors build the RAN on top of ResNet [2]. The output of the trunk branch is a forward pass of the input using these layers, used for feature extraction. The mask branch on the other hand, consists of a set of downsampling and upsampling layers, or a ”bottom-up top-down” approach as referred to by the authors, which computes a soft mask as the output. This output is then combined with the trunk branch's output to become the final output of the attention module. The layout of the attention blocks and the RAN as a whole is depicted in Figure 1.

The authors also incorporate heavy use of residual units within the architecture of the network. This helps combat two problems introduced by the masks: feature degradation and forced masking of the trunk branch. The former issue arises with the repeated application of the masks, which themselves range in values from 0 to 1. With repeated blocks of attention modules, the data gets dulled out because of repeated multiplication of decimal value masks. The residuals units help combat this because of the introduced skip connections that help propagate values before the mask operation and retain feature quality. The latter issue derives from the soft mask branch potentially preventing the identity mapping of the trunk branch. Instead of forcing a mapping at all times, sometimes it may be advantageous to allow the trunk branch to perform its operation unimpeded. This is the motivation for introducing residual skip connections to preserve functionality of identity mapping and help balance the influence of the soft mask branch.

2.2. Key Results of the Original Paper

The authors were able to achieve state-of-the-art results in CIFAR-10, CIFAR-100, and ImageNet tasks, of which we will focus on CIFAR-10. The achievements in CIFAR-10 are highlighted in Table 1.

TABLE 1: Test Errors on CIFAR-10 for different RAN configurations compared with state-of-the-art performances [1]

Network	Test Error
ResNet-164	5.46
ResNet-1001	4.64
WRN-16-8	4.81
WRN-28-10	4.17
Attention-92	4.99
Attention-236	4.14
Attention-452	**3.90**

The authors also demonstrate that attention residual learning (ARL) is consistently superior to naive attention learning (NAL). In this context, ARL refers to the use of residual connections in the application of the mask to the trunk branch's output. With NAL, there is no such connection, meaning the identity mapping can no longer be computed. The results are summarized in Table 2

TABLE 2: Top-1 errors compared for RAN architectures using ARL and NAL in the CIFAR-10 task [1]

Network	ARL	NAL
Attention-56	**5.52**	5.89
Attention-92	**4.99**	5.35
Attention-128	**4.44**	5.57
Attention-164	**4.31**	7.18

Overall, the RAN model yields a significant improvement over its predecessors. Further, the results indicate that the best performances are achieved by deeper networks with an ARL configuration.

3. Our Approach

The objective of this work is to replicate the results of the authors of the residual attention network and investigate whether modified architectures with fewer parameters but the same number of layers can achieve similar performances. The authors ran more than 20 different versions/training iterations of various Residual Attention models, primarily using the massive ImageNet dataset, as mentioned in the abstract, as well as the smaller and simpler CIFAR-10 and CIFAR-100 datasets. It wasn't realistic for us to train on the million-plus ImageNet dataset (which is also much larger in terms of pixels, set to $224x224$, versus $32x32$ for CIFAR). We tried a few variants of each model, though not to the scale undertaken by the authors. They trained a few very deep models, as deep as Attention-456. The numerical suffix for each attention model version refers to the number of weighted layers in the trunk branch of the Attention modules, with the formula $36 * m + 20$, with m being the number of Attention modules (roughly) at each stage (1 for Attention-56, mean(2) for Attention-92, as so on. Though in actuality for Attention-92, the authors used $1- > 2- > 3$ attention modules at each stage.

3.1. Motivation

3.1.1. Objectives and Technical Challenges. The goal of our investigation was to try to perform as many of the experiments done in the paper as possible. Obviously running each model would have been costly in terms of time and resources, particularly if we used ImageNet, which would have taken days to train, most likely. Therefore we stuck to CIFAR-10 (and time willing CIFAR-100).

Trying to recreate the paper's exact architecture, from the exact architecture of each attention module down to the convolutional dimensions was a challenge. In particular if you refer to Figure 1, the authors only provided a detailed architecture of Attention Module 2. However, we could gleam from the diagram that Attention Module 1 had 2 skip connections and Attention Module 3 had none. From there we had to research what the best way to implement these were. However, as we were not certain, we looked at existing implementations to get a better understanding of how to actually implement the model. We looked at the authors' Github page, where they had coded their model using Caffe. However there was only information on ImageNet models, soways of implementing the Attention 56 and Attention 92 models. However we wrote our own code for generating these architectures, and made some modifications.

We will briefly go over the reason for using residual modules in the system. Deeper neural networks, whether dense or convolutional tend to be hard to train. There are issues with deep networks where adding layers can cause a

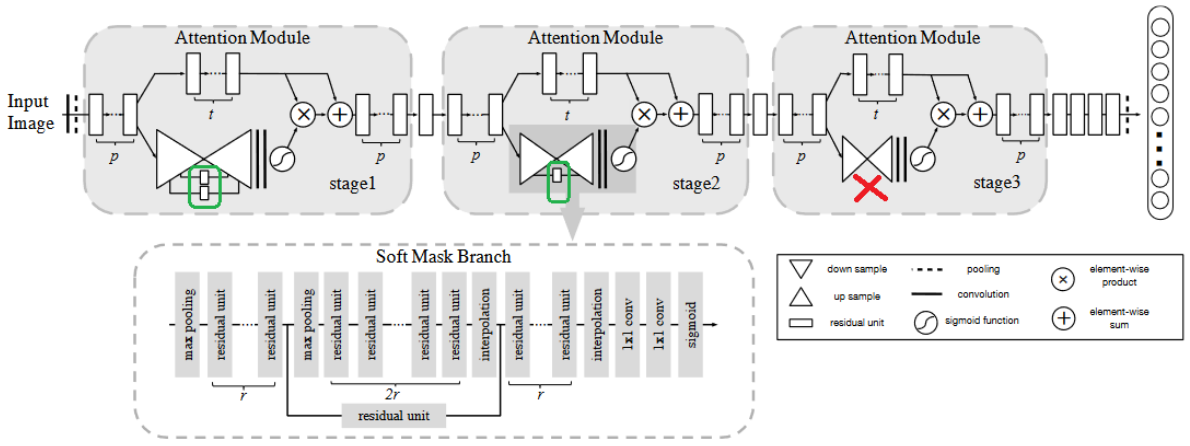
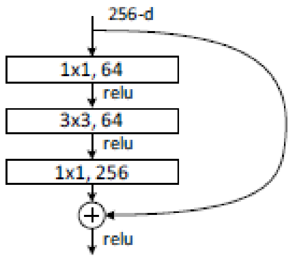


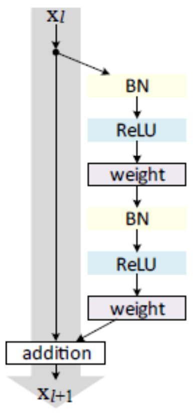
Figure 1: Illustration of the stacked attention blocks in the RAN (note the presence or absence of skip connections in the soft max branch) [1]

greater training error in addition to validation error, which is contrary to conventional wisdom, that a deeper network can learn the underlying training data better. The paper by He et al. shows that using an identity skip connection, as shown in [2] (also see Figure 2a) can help resolve the degradation at deeper layers. The nonlinearity of neural networks accounts for some of the issues at deeper layer, where it would be harder to optimize the underlying functions. Adding the skip connection seems to help mitigate some of the degradation. Since the Residual Attention Network is very deep, and will likely have such degradation issues, the use of residual units as building blocks is necessary.

Also we utilize pre-activation, as shown in Figure 2b, which is from the paper on Identity Mappings [4]. By using batch normalization and activations before passing the input into the weighted (convolutional) layer, you induce regularization in the model, as both batch normalization and activation functions reduce the variance of the input data. 2.



(a) Basic Residual Unit [2]



(b) Pre-Activation in Residual Units [4]

Figure 2: Residual Unit Structures

3.2. Implementation

As mentioned, the paper itself was missing some details on how to implement all the Attention modules, and also was less detailed about the CIFAR implementation than about ImageNet. Nevertheless we were able to come up with a design after looking at various sources and examples. The first part was implementing the residual units which form the basic building block of much of the network. We used the basic structure showed in Figure

We combine the higher level structure of Figure 2a, with the pre-activation of Figure 2b. Note that the skip-connection input and the output of the last $1x1$ convolutional layer have to have the same dimensions so they can be added element-wise. The basic dimensions structure of our residual unit is shown in Table 4. It is very important to note that residual units used in the Attention Module must keep preserve the incoming dimensions, so all the Conv2D strides are set to 1. However, outside of the Attention Modules, the middle $3x3$ Conv2D block has stride set to to 2, as is the Conv2D for the skip identity layer. This is because outside of the Attention Modules, the image dimensions get reduced (within the Attention module the dimensions of inputs and outputs stay the same). We then had to finalize the structure of the Attention Modules. At each stage (1, 2, 3), we had a different implementation of the Attention Module, with the difference being in the mask branch (Look at Figure 3).

All the soft mask branches have a special output convolutional layer with a sigmoid function output, as seen at the end of the diagram. We implemented a version of this that used pre-activation, like the residual units.

Our final Model Diagram (for Attention 56) is as Follows:

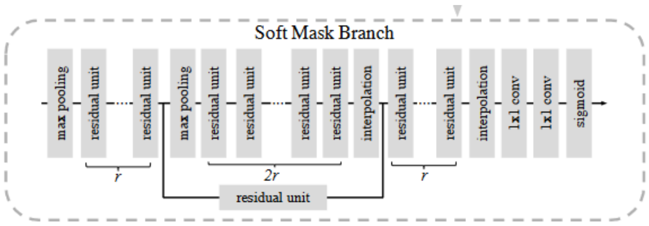


Figure 3: Softmax Branch Closeup [1]

TABLE 3: Residual Unit Structure

Layer-Type	**Dimensions- c is channel size**
BatchNorm 1	-
ReLU 1	-
Conv2D 1	$1x1x$c1, stride=1
BatchNorm 2	-
ReLU 2	-
Conv2D 2	$3x3x$c2, stride=1(Attn), 2(non-Attn)
BatchNorm 3	-
ReLU 3	-
Conv2D 3	$1x1x$c3, stride=1
Conv2D ID (Input Identity)	$1x1x$c3, stride=1(Attn), 2(non-Attn)
Add	(Conv2D 3 + Conv2D ID)

TABLE 4: Differences between Attention Layers at Each Stage

Attention Stage	**Attention Module Characteristics**
1	Same as Figure 3, except an Outer Skip Residual Unit connecting the node between MaxPool 1 and Res Unit, to node between Interpolation and Res Unit
2	See Figure 3
3	No skip connections, and no downsampling-upsampling, just 4 res units

For Attention 92, we stacked 1, 2, and 3 attention layers at each stage, each of type 1, type 2, and type 3. We plotted graphs of the architectures as well. The files are called:

We also implemented a modified version of our design, called version 2, based on an example we found online [5], specifically a popular version we found in Github ran:git. We were interested in finding out how it performed. It notably uses the trunk input in parts of the soft attention mask, and its version of the 2nd and 3rd attention modules are smaller,

TABLE 5: Attention 56 Architecture

Layer	**Output Size**	Layer Dimensions
Input	32x32x3	-
Conv1	32x32x32	3x3x32
BatchNorm	32x32x32	-
ReLU	32x32x32	-
Res1	32x32	Channels=(16,16,64)
Attn1	32x32	Channels = (16,16,65)
Res2	16x16	Channels=(32,32,128)
Attn2	16x16	Channels = (32,32,128)
Res3	8x8	Channels=(64,64,256)
Attn3	8x8	Channels = (64,64,256)
Res4	8x8	Channels = (128,128,512)
Res5	8x8	Channels = (128,128,512)
Res6	8x8	Channels = (128,128,512)
BatchNorm	8x8X512	-
ReLU	8x8x512	-
AvgPool	1x1x512	-
Dense-Softmax	1x1x512	

yet the capacity was much greater, as it used filers sizes of (32,32,128),(64,64,256),(128,128,512),(256,256,1024). We sketched a diagram of this model's attention modules, as follows in Figure 4:

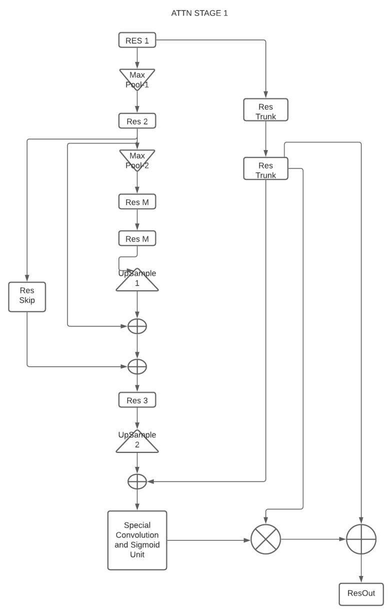
2.

Roughly speaking, the Architecture for this novel version of the RAN is as follows: This model has a much higher capacity than our original model, and it showed in terms of training times.

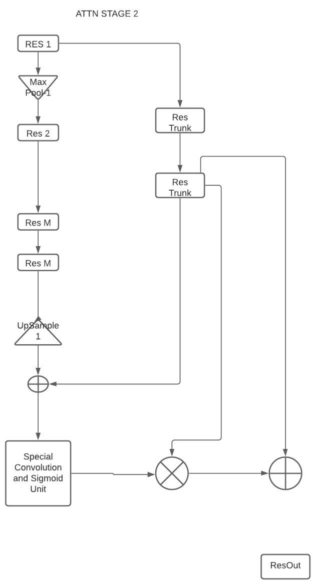
One last thing: we also plotted the architectures of our versions as follows: 'ran56_v1_archtecture.png', 'ran56_v2_archtecture.png', 'ran92_v1_archtecture.png', 'ran92_v2_archtecture.png'. Look at these in the graphs folder for more information on the architectures.

4. Experiments

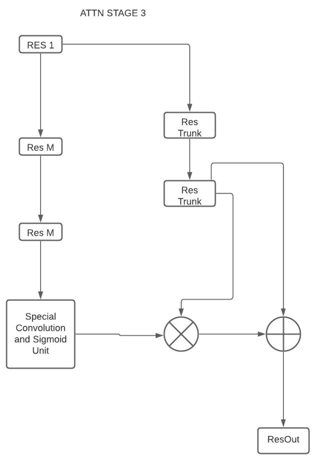
In this section, we evaluate all our models on the CIFAR-10 dataset given the computational requirements of the models. The first part of the experiments seeks to partially replicate the results of the original paper, implementing ARL



(a) Modified Attention 1 Model



(b) Modified Attention 2 Model



(c) Modified Attention 3 Model

Figure 4: Attention Modules for Modified RAN Model

TABLE 6: Attention 56 Architecture

Layer	**Output Size**	Layer Dimensions
Input	32x32x3	-
Conv1	32x32x32	3x3x32
BatchNorm	32x32x32	-
ReLU	32x32x32	-
Res1	32x32	Channels=(32,32,128)
Attn1	32x32	Channels = (32,32,128)
Res2	16x16	Channels=64,64,256)
Attn2	16x16	Channels = (64,64,256)
Res3	8x8	Channels=(128,128,512)
Attn3	8x8	Channels = (128,128,512)
Res4	8x8	Channels = (256,256,1024)
Res5	8x8	Channels = (256,256,1024)
Res6	8x8	Channels = (256,256,1024)
BatchNorm	8x8X1024	-
ReLU	8x8x1024	-
AvgPool	1x1x1024	-
Dense-Softmax	1x1x1024	

and NAL versions of Attention-56 and Attention-92. The second part of the experiment explores the effect of reduced parameters on the performance for both Attention-56 and Attention-92.

4.1. CIFAR-10

Details The CIFAR-10 dataset is comprised of 60,000 32x32 RGB images spanning over 10 classes. By default, the split is 50,000 training samples and 10,000 test samples.

For all ARL model configurations, we normalize the training set by subtracting the total RGB pixel-value mean from every sample. A similar process is done with the test set and its data. However, for the NAL model configurations, the mean and standard deviations are computed channel-wise for the red, green and blue channels in the training data. Then, the train and test data are standardized using these computed values from the training set to serve as a slightly alternate, more precise method of normalizing the data than the method used by Wang et al. [1].

On top of this, we incorporated custom preprocessing procedures that are described in Table 7.

TABLE 7: Preprocessing Transformations

Transformation	Parameter Value
Rotation Range	20
Width Shift Range	0.2
Height Shift Range	0.2
Horizontal Flipping	True

We also further split the training dataset to arrive at 40,000 training samples and 10,000 validation samples. Our preprocessing routine is implemented using the Keras data generator.

Hyperparameters We used default Adam optimizer as set by TensorFlow for all experiments, utilizing categorical crossentropy loss.

4.2. Deep Learning Network

TABLE 8: The number of parameters for each architecture, the GPU used, and the training time in hours for 200 epochs.

Network	params$\times 10^6$	GPU	Train Hours
Attention56-v1	2.92	RTX3080	2.94
Attention56-v2-ARL	11.1	K80	3.05
Attention56-v2-NAL	11.1	RTX3080	4.77
Attention92-v1	5.03	RTX3080	4.56
Attention92-v2-ARL	18.2	RTX3080	5.33
Attention92-v2-NAL	18.2	RTX3080	8.17

Table 8 captures the number of parameters, the GPU used, and the training hours for each of the architectures. The custom models, denoted by a suffix of ”-v1”, had considerably less parameters and took less time to train than their counterparts.

Attention92-v2-ARL was only trained for 100 epochs due to time constraints.

4.3. Evaluation Methods

The models are evaluated with Top-1 error and Top-5 error performance metrics for the purposes of direct comparison with the results of Wang et al [1].

Top-1 Error can be thought of as the inaccuracy percentage. It is the proportion of test cases in which the correct class is not predicted by the model.

Top-5 Error is the proportion of test samples in which the correct class is not within the top five guesses of the model, ranked by probability.

5. Results

5.1. Project Results

TABLE 9: Top1-Error and Top-5 Errors of various models on the CIFAR-10 test set.

Network	Top1 Error	Top5 Error
Attention56-v1	8.07	0.339
Attention56-v2-ARL	9.43	0.519
Attention56-v2-NAL	7.93	0.390
Attention92-v1	8.37	0.309
Attention92-v2-ARL	12.7	0.59
Attention92-v2-NAL	33.6	4.31

The results of our experiments are displayed in Table 9. From the results, it’s most apparent that our reduced models performed quite well, being able to nearly match their full-architecture counterparts in performance despite having a fraction of the original number of parameters. This observation is consistent across both attention56 and attention92 variations.

However, the NAL version of attention92 failed to properly converge, hurting its results on CIFAR-10.

5.2. Comparison with Original

TABLE 10: Top1-Error comparison of our replicated model (using ARL), our reduced model (suffix ”-v1”), and the original results described by Wang et al. [1].

Network	Replication	Reduced	Original
Attention56	9.43	8.07	5.52
Attention92	12.7	8.37	4.99

A comparison of our results against Wang et al. [1] is made in Table 10. The gap between the original results and our replicated results is significant, possibly due to minor deviations in architectural implementations as well as the optimizer not being consistent with the authors.

Despite that, our reduced models prove to be relatively promising, as they showcase a marked improvement over our own replicated results. However, there is still a significant gap between our best results and those achieved by Wang et al. [1].

5.3. Discussion

As a whole, we were unable to replicate the results obtained by the original authors. We note minor architectural differences such as the use of batch normalization and such, as well as differences in optimizers and hyperparameters that could likely lead to a significant difference in performance.

Overall, our results show that the reduced model is able to learn the task well and perform better than our replicated results despite containing less than half of the original number of parameters. Furthermore, they offer boosted training times on top of better performance.

6. Future Work

Considering that our replicated models performed worse than the original models used by Wang et al **??**, there is much room for potential improvement with our reduced architecture. It remains to be seen how the reduced architecture will perform given we correct our implementation to be identical to that of Wang et al.

Another possible avenue would be to also explore the benefits of reduced architectures as the base models get deeper. The original authors have implemented architectures such as Attention452, allowing for the differences in performance between the original and reduced models to be measured. With both experiments, we can determine if performance gains or at least performance parity can be achieved with significantly less parameters and training resources used.

7. Conclusion

We attempt to replicate the results of state-of-the-art residual attention network models as implemented by Wang et al [1] on the CIFAR-10 dataset to limit computational requirements. We were able to achieve good results, but failed to replicate the results of the original authors, most likely due to minute differences in implementation. In addition, we defined custom architectural variants of the ones proposed by the original authors with reduced parameter sizes. These models were able to train quicker and achieved better results than the original models that we replicated. While we didn't achieve state-of-the-art results, the figures we gathered are promising and open up further avenues into deeper experimentation. For one, the performance parity can be measured across varying depths of the architecture to see if the reduced networks are more efficient to train for very deep models.

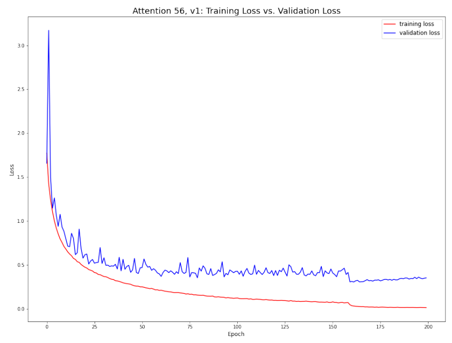
8. Appendix

8.1. Individual Student Contributions in Fractions

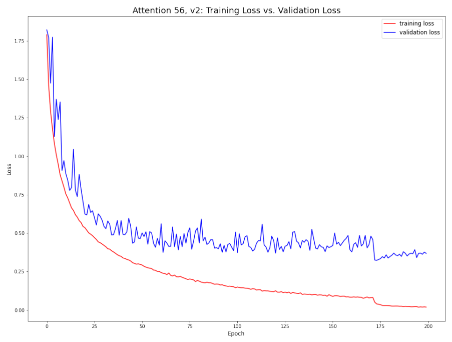
TABLE 11: Individual Student Contributions in Fractions

	dd2676	**sa3976**
LastName	Dwarakanath	Adari
Fraction of (useful) total contribution	0.5	0.5
What I did 1	Implemented code	Train models
What I did 2	Half the paper	Half the paper
What I did 3	Generated graphs and diagrams	Evaluation results

??
??



(a) Losses against number of training epochs of reduced attention56



(b) Losses against number of training epochs of attention56-NAL

References

[1] Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., … Tang, X. (2017). Residual Attention Network for Image Classification. *CoRR, abs/1704.06904*. Opgehaal van http://arxiv.org/abs/1704.06904

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *CoRR, abs/1512.03385*. Opgehaal van http://arxiv.org/abs/1512.03385

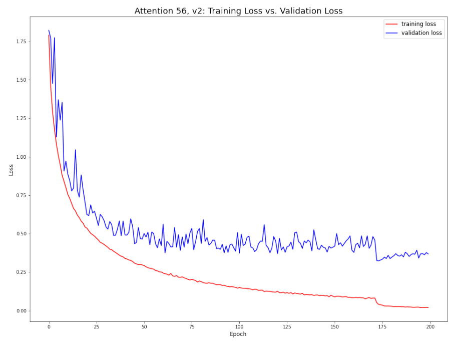
[3] Mnih, V., Heess, N., Graves, A., & Kavukcuoglu, K. (2014). Recurrent Models of Visual Attention. *arXiv [cs.LG]*. Opgehaal van http://arxiv.org/abs/1406.6247

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity Mappings in Deep Residual Networks. *CoRR, abs/1603.05027*. Opgehaal van http://arxiv.org/abs/1603.05027

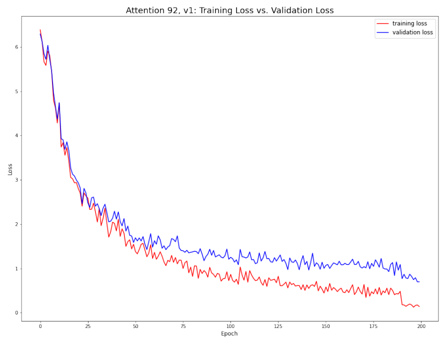
[5] 'Alternative Implementations of RAN'. Retrieved from: https://paperswithcode.com/paper/residual-attention-network-for-image

[6] "Novel Implementation of RAN". Retrieved from: https://github.com/tengshaofeng/ResidualAttentionNetwork-pytorch

[7] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.



(a) Losses against number of training epochs of reduced attention92



(b) Losses against number of training epochs of attention92-NAL