Residual Attention Network for Image Classification

E4040.2020Fall.JFXB.report
Weixi Yao wy2350, Yifei Xu yx2577
Columbia University

Abstract

The focus of this report is to summarize the key points of the original paper, to reproduce the Residual Attention Network using TensorFlow framework and to verify the effectiveness of the reproduced model on CIFAR-10. At the end of this report, we successfully reproduce the model entirely from scratch and obtained the test accuracy as 87%. Moreover, we gain lots of useful learnings from comparing the results of the original paper against our slightly worse results.

1. Introduction

It has been proven that humans do not put their attention entirely on the whole scene at once when they perform a recognition task. Instead, humans tend to pay more attention to the most important features of the scene to maximize the amount of discriminative information needed for the task. Inspired by this mechanism, many deep learning research scientists working in computer vision have spent a tremendous amount of effort in researching ways to capture the mixed nature of attention and to integrate the attention mechanism in their networks.

The attention mechanism is used to make CNN learn and focus more on the important information, rather than learning non-useful background information. As the implementation of attention mechanism in deep learning, Attention Module usually consists of a simple 2D-convolutional layer, MLP, and activation function at the end to generate a mask of the input feature map.

However, the study has shown that stacking Attention Modules directly would lead to an obvious performance drop. Therefore, besides the efforts made in creating the Attention Module, efforts have also been made in allowing for the training of a deeper network. Related to the work of Wang et al., Deep Boltzmann Machine contains top-down attention by its reconstruction process in the training stage. Recurrent neural networks and long short-term memory widely applied attention mechanism, where top information is gathered sequentially and decides where to attend for the next feature learning steps. Techniques of residual learning allow for the training of very deep feedforward neural networks. Also developed recently, soft attention can be trained end-to-end for convolutional networks, and was also integrated into the fast-developing feedforward network structure of the Residual Attention Network creatively.

Combining all the great works in Attention Module and residual learning, Wang et al. proposed a revolutionary convolutional neural network with state-of-art feed-forward network architecture in an end-to-end training fashion, called "Residual Attention Network", in their 2017 paper, *Residual Attention Network for Image Classification*[2]. In a nutshell, Wang et al.'s Residual Attention Network is built by stacking Attention Modules, optimized using attention residual learning mechanism, which generates attention-aware features and allows for the training of very deep networks.

2. Summary of the Original Paper

2.1 Methodology of the Original Paper

The implementation of Residual Attention Network is built based on two important achievements, the Attention mechanism, and the Residual Network:

Attention mechanism not only makes the computation focus on specific regions of the images but also strengthens the "recognizability" of the "features" of those regions.

Residual Network allows training of "very deep" networks while achieving extraordinary results in image classification tasks.

Given the above two building blocks, Wang et al. proposed a *Residual Attention Network* that inherits the following important properties. Stacking Attention Modules leads to a consistent performance increase, as different Attention Modules at different layers extract different types of attention. Because of the inclusion of residual units, the Residual Attention Network can be easily extended to hundreds of layers. Under this strategy, the Residual Attention Network significantly decreases the amount of computation while achieving equally good results as other state-of-art networks.

In implantation, the three main contributions of Wang et al.'s paper are:

(1) Capturing different types of attention is made possible by the stacked network structure.
(2) The inclusion of the attention residual learning mechanism allows full learning of attention modules at different layers.
(3) The application of the bottom-up top-down feedforward structure allows feature extraction as well as the development of Attention Map.

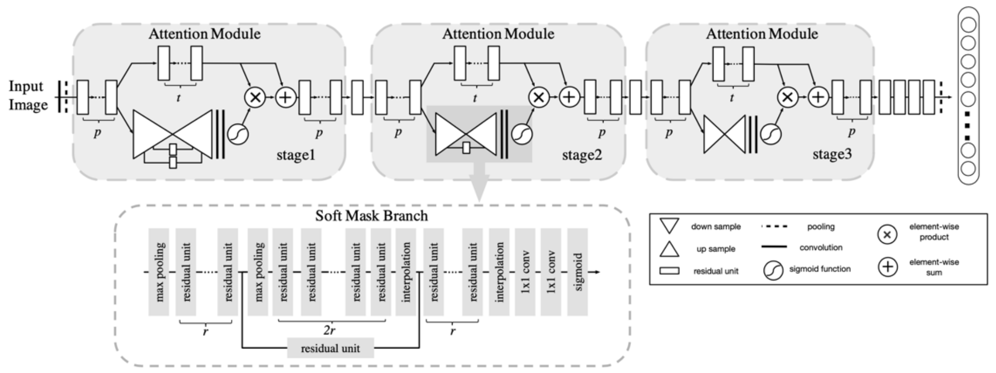

Fig.1 Example architecture of the proposed network. They use three hyper-parameters for the design of Attention Module: p, t and r. The hyper-parameter p denotes the number of pre-processing Residual Units before splitting into trunk branch and mask branch. t denotes the number of Residual Units in trunk branch. r denotes the number of Residual Units between adjacent pooling layer in the mask branch.

According to Wang et al.'s paper, the attention module can be split into two branches, the mask branch, and the trunk branch. The purpose of the mask branch is to output Attention Feature Maps of the same sizes and softly weights the output of the trunk branch which can be any of the SOTA networks. The final output of the attention modules will be the dot product of the mask branch and trunk branch.

Wang et al. also point out that, although the attention modules play an important role in image classification tasks, simply stacking attention modules will lead to a performance drop because of two reasons:

One is that since the output of the mask branch, activated by the sigmoid function, is always 0 to 1, the resulting dot production will degrade the values of features as layers become deeper.

Another reason is that the output attention map of the mask branch can potentially break the good property of the trunk branch, such as the identical mapping of the Residual unit. If the shortcut mechanism in the Residual unit is switched by the mask branch, the gradients of the deep network can hardly be back-propagated.

To solve the above issues, the output of the attention module is replaced by

$$H(x) = (1 + M(x))F(x)$$

In this formula, $M(x)$ takes values between 0 and 1, and the addition of 1 alleviates the issue of degraded values of features in deep layers. Another difference between the Residual Network and the Residual Attention Network is that the $F(x)$ in the Residual Network learns the "residuals" between the input and the output, while the $F(x)$ in the Residual Attention Network indicates the features generated by deep convolutional networks.

Combined with the output of the mask branch, the $F(x)$ can strengthen the important features in the attention map and weaken the unimportant features.

Based on the above improvements, the Attention residual learning can not only keep good properties of the original features, but also gives those features the ability to bypass soft mask branch and forward to top layers to weaken mask branch's feature selection ability. As a result, stacking attention modules consistently increase the expressiveness of the Residual Network.

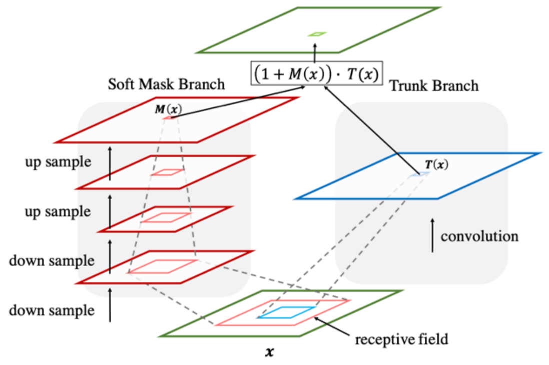

Fig.2 The receptive field comparison between mask branch and trunk brank

In the mask branch, the processing of the attention map is driven by the fast feed-forward sweep and top-down feedback steps. The purpose of the former operation is to quickly encode and to extract global features of the entire image, while the purpose of the later operation is to combine global information and original feature maps. Similar to the setting of the FPN network, the motivation behind the top-down feedback steps is to better integrate the context information as well as the features from various layers.

According to *Fig.1* and *Fig.2*, the detailed architecture of a sample Residual Attention Network is the following:

(1) From input, max pooling is performed several times to increase the receptive field rapidly after a small number of Residual Units.
(2) After reaching the lowest resolution, the global information is then expanded by asymmetrical top-down architecture to guide input features in each position.
(3) Linear interpolation up samples the output after some Residual Units. The number of bilinear interpolation is the same as max-pooling to keep the output size the same as the input feature map. The motivation behind this step is to maximize the expressiveness of the feature map through integrating local and global features.
(4) Two consecutive 1x1 convolution layers then outputs a feature map that has the same dimensionality as the input but has only one channel.
(5) Then a sigmoid layer normalizes the output range to [0, 1].
(6) Skip connections between bottom-up and top-bottom parts are also added to capture information from different scales.

2.2 Key Results of the Original Paper

According to Chen Qian, the Residual Attention Network applies a popular attention idea to very deep feedforward networks successfully. It improves the parameter efficiency performance compared to state-of-art networks, such as ResNet, ResNeXt, and Inception-ResNet in image classification tasks. The effectiveness of the Residual Attention Network was further verified by extensive analyses conducted on CIFAR-10, CIFAR-100, and ImageNet datasets.

The classification error (%) on CIFAR-10 is posted below:

Network	clf err. % on CIFAR-10
Attention-56	5.52
Attention-92	4.99
Attention-128	4.44
Attention-164	4.31

As we can see from the results above, as the layers of network gets deeper, the classification error consistently improves. Similar results are shown on CIFAR-100:

Network	clf err. % on CIFAR-10	clf err. % on CIFAR-100
Attention-92	4.99	21.71
Attention-236	4.14	21.16
Attention-452	3.0	20.45

Wang et al. also evaluated the Residual Attention Network on the ImageNet dataset, comparing the results of other state-of-art models:

Network	clf err. % on ImageNet
ResNet-152	22.16
Attention-56	**21.76**
ResNet-200	20.1
Inception-ResNet-v2	19.9
Attention-92	**19.5**

Due to limited computing results, we only focused on the experiment results of CIFAR-10, generated using the Attention -56 network which is also the network we tried to reproduce.

3. Methodology (of the Students' Project)

3.1 Objectives and Technical Challenges

The objectives of this report are to reproduce the Attention-56 network using TensorFlow framework and evaluate the model using CIFAR-10 dataset. The most technically challenging parts of this task are:

Reproduce the Attention Module Even though the paper has provided all the parameters and the entire architecture, we had huge troubles developing the Attention Module entirely from scratch, especially during the early stage of the development.

Obtain Good Performance It is hard to reproduce results equally good as that of the original paper. Although we managed to reproduce a model that has almost identical architecture of Attention-56 network, it's initial results on CIFAR-10 were still far from ideal. More specifically, the model easily overfits and doesn't generalize well enough on the test set. Also, the low resolution of CIFAR-10 raised the difficulty of our challenging job.

3.2 Problem Formulation and Design Description

The above issues were finally alleviated by the following approaches.

Ultimately, we successfully developed a working version of Attention Module based on inspirations drown from the architecture of residual block in ResNet (detailed flow chart and illustration diagram will be posted in the following section).

To alleviate overfitting, we started by augmenting the dataset, using similar approaches discussed in the original paper (detailed implementation will be discussed in the following section). We further tackle the overfitting issue by downsizing the residual units from three layers to one (detailed implementation will be discussed in the following section).

4. Implementation

In our project, we implemented the residual attention network on the CIFAR-10 and CIFAR-100 dataset. The model architectures shown in the Wang et al.'s paper is designed for ImageNet. When we reproduced an exactly same model as paper, we found the model is overfitting with training acc 98% and validation acc around 70%. Thus, we modified the model architecture to fit our dataset, including the reduce of layers both in the residual unit and attention module.

4.1 Deep Learning Network

Data Description The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Data Preprocess We used the preprocessing methods similar to the paper:

- Validation set: Split the trainset into training and validation sets.
- Mean Subtraction: Subtract the mean of training set from all images.
- Data Augmentation: Three kinds of augmentation methods are applied, that is, width shift, height shift and horizontal flip. The transfer effects are shown in *Fig.3*. We set the parameters as follows:

i) width_shift_range = 0.125,
ii) height_shift_range = 0.125,
ii) horizontal_flip = True

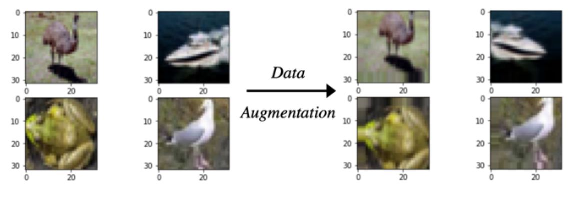

Fig.3 Samples for data augmentation

Architectural Blocks

Layer	Parameters
Conv1	3×3
Max Pooling	2×2
Residual Unit	3×3, 32
Attention Module	Attention$\times$ 1
Residual Unit	3×3, 64
Attention Module	Attention$\times$ 1
Residual Unit	3×3, 128
Attention Module	Attention$\times$ 1
Residual Unit	3×3, 256
Residual Unit	3×3, 256
Residual Unit	3×3, 256
Average Pooling	2×2 stride 2
Flatten	-
FC, softmax	512

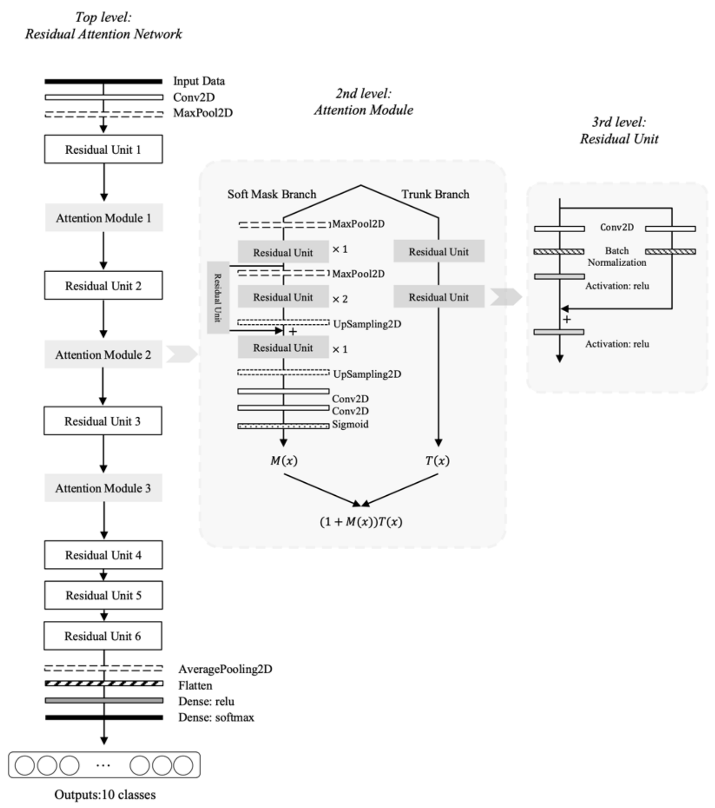

Fig.4 Architecture Flow chart: 3 levels in our model implementation code

Parameters We trained the model using the same optimizer and parameters with that in paper, that is nesterov SGD with a mini-batch of size 64 and epoch 200. We also used a weight decay of 1e-4, a momentum of 0.9 and the initial learning rate. Moreover, we changed the learning rate at some point, using the original methods in paper for reference. In the paper, they trained 160k iterations in total and making learning rate divided by 10 at 64k and 96k. We used "LearningRateScheduler" and set the scheduler to make a division at 75 and 150 epochs.

4.2 Software Design

Flow charts We drew a flow chart exactly the same with our model implementation design, which containing 3 levels block. Charts are shown in *Fig.4*

(1) The top level consists of Conv, MaxPooling, Dense layers, 6 Residual Units (UR) and 3 Attention Modules (AM). The ordinary layers are stacked similar with other classic neural network models, while URs and AMs blocks are stacked alternatively.

(2) The second level is an *Attention Module*. It consists of two branches: Trunk Branch and Soft Mask Branch. Trunk Branch is quite simple. It contains only 2 residual units. While the Soft Mask Branch is much more complex. It includes process of down-sampling and up-sampling, in our implementation, they are coded as 'MaxPooling2D' and 'UpSampling2D'. We set the parameters same as the paper, that is, $\{p = 1, t = 2, r = 1\}$.

(3) The third level is a *Residual Unit*. It also consists of two branches. We designed this block according to the ResNet, but different with the original paper, we only used one combination [CONV, BN, RELU] instead of three.

Github Link: https://github.com/ecbme4040/e4040-2020fall-project-jfxb-wy2350-yx2577

5. Results

5.1 Project Results

We applied all the methods, architecture design and parameters we mentioned above, and obtained a best model with training accuracy 0.9836, validation accuracy 0.8740 and test accuracy 0.8662. The training history of our best model are shown in *Fig.5*. We could see that there are 2 'jumps' in the acc plot, corresponding to the learning rate division at epoch 75 and 150.

We also tried several different models with different data preprocessing and parameters to better tune the model and better understand the networks.

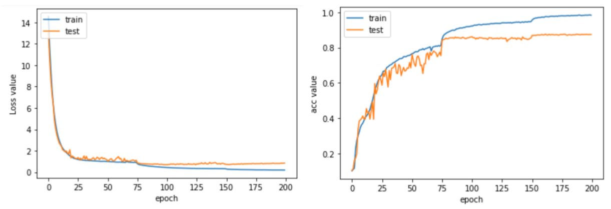

Fig.5 Best model training history plot

Starting Point We first build a model which has the same layer architecture as the final model but without data augmentation and weight decay. We obtained training accuracy 0.9813 and validation accuracy 0.7155. From the plot, we could easily find that the model is totally over-fitting. Even if we tried to apply some normal methods dealing with over-fitting like Dropout and Batch Normalization to the model, we could only obtain a val acc around 0.75.

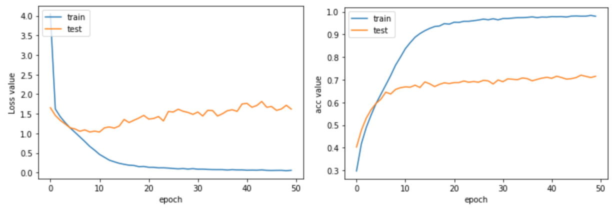

Fig.6 Start point model training history plot

More Aggressive Data Augmentation We tried to add more augmentation, that is, vertical flip and zca whitening. However, the model performance is poorer than the previous augmentation. Test accuracy is 0.8042.

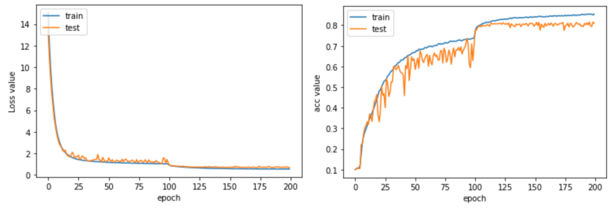

Fig.7 Aggressive Data Augmentation training history plot

Without Learning Rate Decay If we apply the learning rate division only at epoch 100, we will get a result slightly lower than best model.

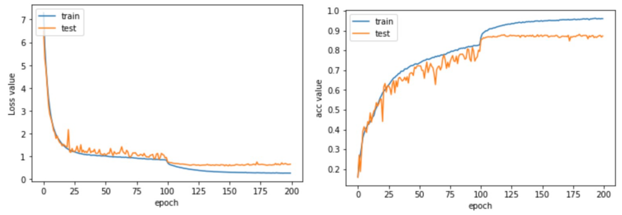

Fig.8 Learning rate division only at epoch 100

Methods Change	Train acc	Val acc	Test acc
Best model	0.9836	0.8740	0.8662
Starting Point	0.9813	0.7155	0.7083
Augmentation	0.8575	0.8094	0.8042
Learning Rate	0.9604	0.8727	0.8618

5.2 Comparison of the Results Between the Original Paper and Students' Project

Comparing the best CIFAR-10 results between Wang et al.'s and ours, I believe the causes of our slightly worse results are:

(1) The working environments and the computing resources aren't exactly the same. As we can see, the model built in the original paper was written in caffe framework and was allowed to run over 100k iterations, the huge difference in computing resources might prevent us from boosting our model to its best capability.

(2) Although the architectures of the models are almost identical, Wang et al. might use slightly different parameters and other fine-tuning tricks that are not fully disclosed in the original paper.

We applied a more aggressive image processing method than Wang et al. do in the original paper. For example, Wang et al. propose to zero-pad the original images to 40x40 by randomly crop 32x32 out of the padded images. However, we directly shift the images vertically or horizontally by 4 pixels, filling the shifted regions with "closest" pixels.

5.3 Discussion of Insights Gained

Most of our insights came from our battle against overfitting. According to our results, the biggest sign of overfitting is the consistently growing gap between training accuracy and validation accuracy as epoch goes deeper. Here are the most important insights we gained from battling overfitting:

- Always augment the images before training, especially when the data has relatively low resolution. The motivation behind this approach is to increase the expressiveness and diversity of the images. Especially when the original data has relatively low resolution, data augmentation is very useful.
- Always test various learning rates during different stages of training or use self-adaptive optimizer if possible. According to our initial modeling results, validation accuracy stagnates at early epochs. This is common when the epochs number is large, and decreasing the learning rate supposedly helps pull the models from the stuck saddle points. Our intuition was ultimately confirmed by our later training results, where we divde the learning rate by 10 at various epochs, similar to what the original paper proposed.
- For very deep network, always include some forms of regularization. We started by a network without data augmentation and kernel regularizer, and we noticed that the gap between the training and validation accuracy becomes too large too early.

6. Conclusion

The focus of this report is to summarize the key points of the original paper, to reproduce the Residual Attention Network using TensorFlow framework and to verify the effectiveness of the reproduced model on CIFAR-10.

Although our results on CIFAR-10 are slightly worse than the results achieved in the original paper, as discussed above, lots of intuitions regarding the residual learning and attention mechanism were verified.

We also verified the effectiveness of various regularization methods, such as data augmentation and kernel regularizer.

Although we achieve initial success in reproducing the residual unit and attention module, there is still room for future performance optimization. For example, we still need to further explore the difference between the architecture of the two models and understand what prevents our models from achieving equally good results. After that, we could start researching the effectiveness of different model designs, such as vertically, instead of horizontally, stacking attention modules.

7. Acknowledgement

As we mentioned above, the implementation of our residual unit was inspired by the architecture of the residual block in ResNet, plus extra convolution and batch normalization layer. Other than that, to battle against slow improvement in deep epochs, we also referred to the official keras documents of LearningRateScheduler class for help. However, the actual implementation was based on observation of actual training/validation curve.

8. References

[1]Github Link: https://github.com/ecbme4040/e4040-2020fall-project-jfxb-wy2350-yx2577

[2] Wang, Fei, et al. "Residual Attention Network for Image Classification." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, doi:10.1109/cvpr.2017.683.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.

[4] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. In ICML, 2015

9. Appendix

9.1 Individual Student Contributions in Fractions

	wy2350	Yx2577
Last Name	Yao	Xu
Fraction of (useful) total contribution	1/2	1/2
What I did 1	Paper Reading and Data preparing	
What I did 2	Modeling	
What I did 3	Writing: Summary of original paper and Methodology	Writing: Implementation and Results