Residual Attention Network for Image Classification

Shengjie Sun
ss5593
Department of Statistics
Columbia University
`ss5593@columbia.edu`

Yiming Tan
yt2658
Department of Statistics
Columbia University
`yt2658@columbia.edu`

Feng Su
fs3658
Department of Statistics
Columbia University
`fs3658@columbia.edu`

Abstract

In this project, we reimplement a convolutional neural network structure called "Residual Attention Network" proposed by Wang et al. [2017]. The great power of attention in machine translation has indicated by Vaswani et al. [2017] and thus we are curious about how the idea of attention can influence the image classification task. The main challenging of implementation is that the attention part of network is too flexible to choose a good hyper-parameters and the time and computing resources is limited for this project. Finally, our implementation on CIFAR dataset got 90% *percentage of accuracy of the paper but we still argued the idea of this paper maybe not a good choice for CIFAR-10 and CIFAR-100.*

1 Introduction

As Wang et al. [2017] mentioned that the residual mechanism in the image classification can serve to both focused location selection and the enhancement of the representations of focused objects. After the network structure like VGG given by Simonyan and Zisserman [2014] and the improvement of computing resources, the newer networks are mostly deeper. Basically, residual attention network is a very deep convolutional neural network connected by residual unit mixed with soft mask, so called attention in this paper.

As for our project, we want to implement the idea of this paper in tensorflow 2 and then compared our result and the result in the paper. Furthermore, we try to exam what should be a good "attention" for the image classification, the attention mentioned in this paper is actually a soft mask, which is quite different from the word "attention" used in the area of natural language processing.

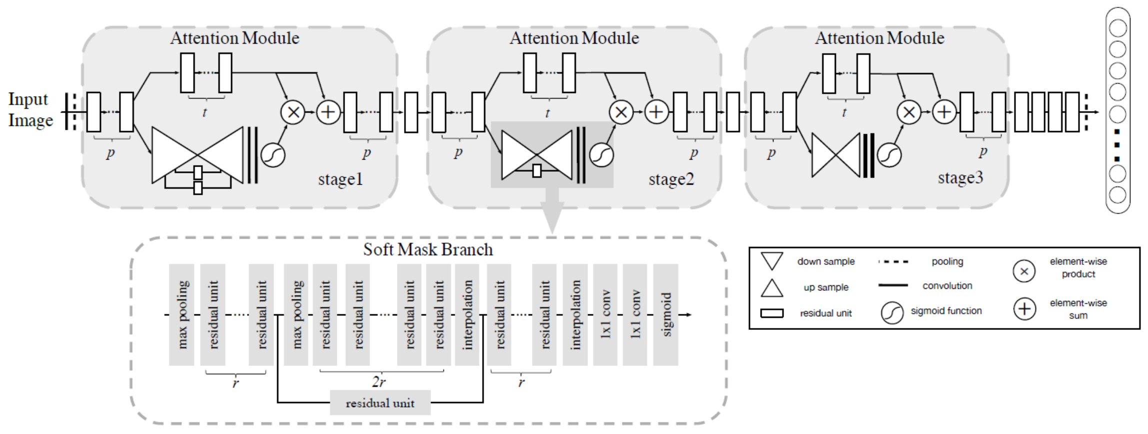


Figure 1 **Example architecture of the residual attention network..** We use three hyper-parameters for the design of Attention Module: p, t and r. The hyper-parameter p denotes the number of pre-processing Residual Units before splitting into trunk branch and mask branch. t denotes the number of Residual Units in trunk branch. r denotes the number of Residual Units between adjacent pooling layer in the mask branch. In our experiments, we use the following hyper-parameters setting: p = 1, t = 2, r = 1. The number of channels in the soft mask Residual Unit and corresponding trunk branches is the same.

There are several challenges and difficulties. First of all, so many implementation details are not provided by the original paper. The details of a residual unit is not clear. The second main challenging is the limited time. Training the model one time is not such time consuming but as we said, the structure of this kind of network is flexible and thus we have so many hyper-parameters to tune. The third one is over-fitting, during training, we found the model tended to over fit the dataset.

Our solution to the first difficulty is to give a model with similar size of parameters. Even thought there are many details, the skeleton of the structure is clear. Therefore, we tried to give an implementation with similar size of parameters. The solution to the second one is that we tried as many acceleration methods as we can to squeeze our computation ability. And we also sacrificed some accuracy based on the real situation. We tried all the method learnt from class but there is still a little bit overfitting.

2 Summary of the Original Paper

2.1 Methodology of the Original Paper

The key idea of the whole paper lies on the structure of attention module and it can be roughly summarized by the following formula:

$$\begin{aligned} H_{i,c}(x) &= (1 + M_{i,c}(x)) * T_{i,c}(x) \\ &= T_{i,c}(x) + M_{i,c}(x) * T_{i,c}(x) \end{aligned}$$

$T(x)$ is the common feed forward layers while the elements in $M(x)$ ranges from [0, 1] and serves as feature selectors.

If we consider $M_{i,c}(x) * T_{i,c}(x)$ as x, it would be exactly the same as residual learning. But it is indeed based on attention structure, and thus called residual attention network.

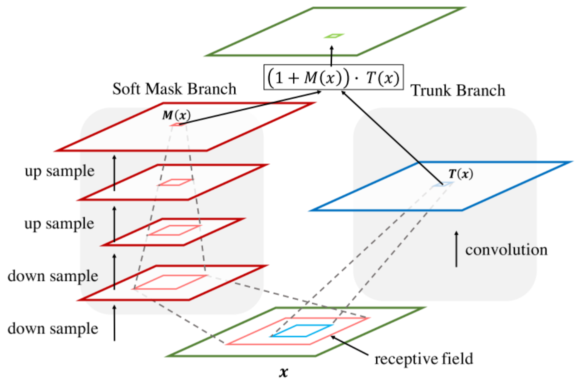


Figure 2 **Structure of Attention Module..** $T(x)$ indicates the features generated by deep convolutional networks. $M(x)$ is the mask branche which work as feature selectors that enhance good features and suppress noises from trunk features.

2.2 Key Results of the Original Paper

2.2.1 Residual structure benefits

This paper argues that utilizing residual structure enables the stacking of more attention modules as shown in Figure 3. This is not surprising too much as this is just why residual learning, i.e. skipping connection, is so popular and useful.

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

Figure 3 **Classification error (%) on CIAFR-10..** The networks trained using attention residual learning technique consistently outperform the networks trained with baseline method. The performance increases with the number of attention module when applying attention residual learning. In contrast, the performance of networks trained with "naive attention learning" method suffers obvious degradation.

The Figure 4 proved above argument in details, we can see that residual attention model has similar relative mean response to ResNet-164.

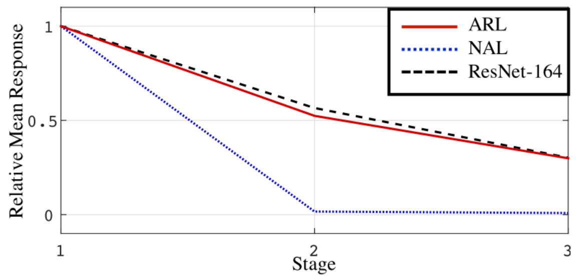


Figure 4 **The mean absolute response of output features in each stage..** The response generated by the network trained using naive attention learning quickly vanishes in the stage 2 after four attention modules compared with network trained using attention residual learning.

2.2.2 Attention structure benefits

The paper shows that their model outperforms the state of art models. On the dataset CIFAR-10 and CIFAR-100, the attention residual learning scheme can effectively reduce the number of parameters in the network while enhancing the accuracy of the classification.

Network	params$\times 10^6$	CIFAR-10	CIFAR-100
ResNet-164	1.7	5.46	24.33
ResNet-1001	10.3	4.64	22.71
WRN-16-8	11.0	4.81	22.07
WRN-28-10	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**

Figure 5 **Comparisons with state-of-the-art methods on CIFAR-10/100.**

3 Methodology

Basically, the model is constructed by attention modules shown in Figure 1. One attention module has two branches: trunk branch and soft mask branch. At each stage, there could be several attention modules and in Figure 1, each stage contains exactly one module. More specifically, in the soft mask branch, there are different number of skip connections between down sampling and up sampling layers.

3.1 Objectives and Technical Challenges

The paper did not provide the details of the residual unit and up or down sampling layers. But for the reimplementation, these details are crucial. In our project, we tried to build Attention56 for CIFAR-10, i.e. in each stage, there is only one attention module. The soft mask branch has two skip connections at stage 1, one skip connection at stage 2 and one at the last stage which is exactly same to to the Figure 1.

The main object is to get the accuracy as close as the one provided in the paper. But as mentioned several times, the details is important but hidden in this paper, so lots of experiments required but the time and computing resources limited.

The other challenge is the coding writing, it is not difficult to build the overall structure for a specific model, say Attention56. But it is super time consuming to adjust the hyper-parameters. It is quite clear there are lots of sampling steps and skip connections. Therefore the dimension and the shape of images/tensors become a little bit annoying to keep consistent. Besides, the different model requires different attention model, at the end, the dimensions become chaos and requires lot of time to adjust. So, if you want to give codes which can reused for many models, it is somehow difficult to design. If not, there will be plenty of repeating codes that fixed the inputs and outputs dimension for each layers.

The last but not least is how to prevent over-fitting. The learning ability of these model are high and over-fitting is common. Even there are so many skip connections, the over-fitting is still obvious. Finding a good way to prevent it for this kind of model structure is difficult for us.

3.2 Problem Formulation and Design

The main goal is to reimplement the idea of residual attention net proposed by the paper. The whole residual attention neural network mainly consists of three attention modules, so the main challenge in implementation is to construct all the components in the attention module. The overall software design is shown in Figure 6, where three attention module objects are used in the residual attention network.

Within the Attention Module class, a MaskBranch class and a TrunkBranch class is built to implement each branch. To build up these two branch classes, several basic units are used. DownSampleUnit, UpSampleUnit and ReidualUnitIdentity are three classes used to construct the MaskBranch, while the TrunkBranch class is a block of ResidualUnit objects.

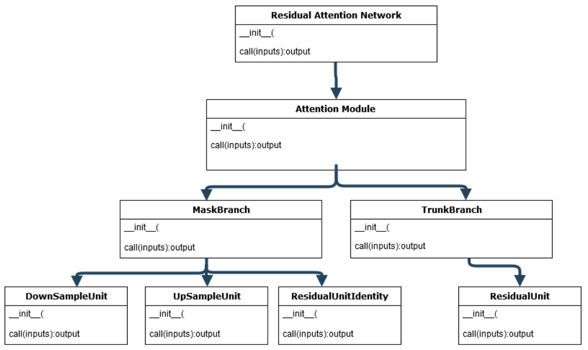


Figure 6 **The architecture of the whole model.**

The further details about the implementation in shown in the next section.

In order to check whether the residual structure can enhance the number of stacking attention module, we exam the weights and gradient during the training process which is discussed in the result section.

4 Implementation

4.1 Deep Learning Network

The architectural diagrams are Figure 6 and Figure 1. As for the training details, it is shown in Figure 12 in Appendix. The general parameters are **epoch and batch**: a maximum of 300 epochs with a batch size of 128 in each epoch is used in training. If the early stopping condition is not satisfied, the model would be trained in at most 300 epochs. **Learning rate and decay**: a decaying learning rate with patience is used. The learning rate is .001, if the validation accuracy can not be improved in 10 epochs, a factor of .6 will be multiplied to the learning rate. **Data Augmentation**: for data augmentation, horizontal and vertical flipping on the original pictures is done. **Early stopping**: early stopping is introduced here by checking the validation accuracy of the model.

If the best validation accuracy could not be improved in 20 epochs, the training would be stopped.

In the following part, we will give details of how the mask and truck brunches are implemented.

Mask Branch Implementation. The algorithm design of Mask Branch is shown in Figure 7. The input of this branch first goes through a loop of down-sampling units until all the down-sampling units are done. After each down-sampling unit, the output is put into an identity list for later use in skip connections. After all the up-sampling units are done, a similar loop consisting of corresponding up-sampling units is done until all the up-sampling units are finished. Before each up-sampling unit, the corresponding identity value in the identity list is added to the output to establish the skip connections between sampling layers. After all the up-sampling units are completed, the output information then go through two convolutional layers and a sigmoid activation function before it is returned as the output of this branch.

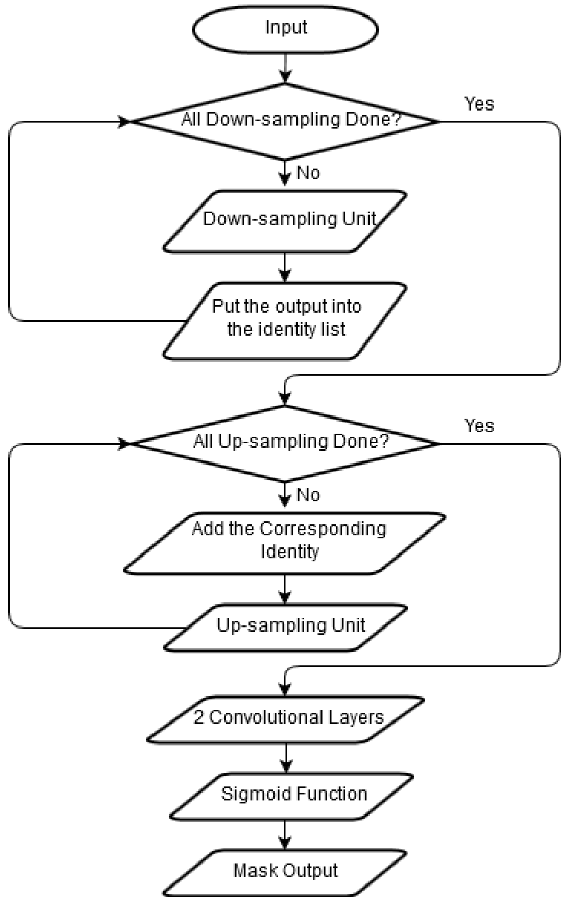


Figure 7 **Mask Branch in an Attention Module.**

Trunk Branch Implementation. As shown in 8, the structure of Trunk Branch is easier and more straightforward. The input of the Trunk Branch goes through residual units until all the residual units defined in this branch all completed. The result after these residual units is then returned as the output of the Trunk Branch.

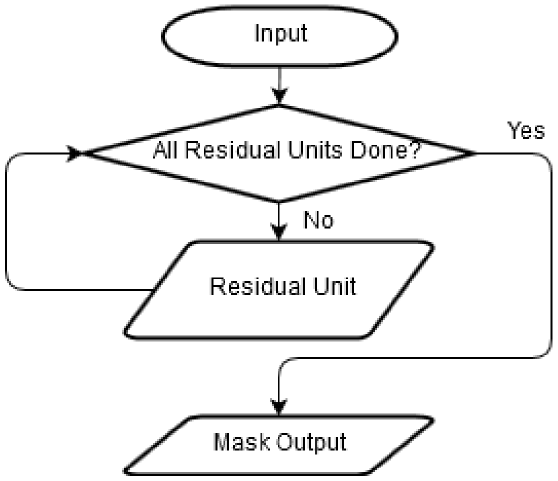


Figure 8 **Trunk Branch in an Attention Module.**

4.2 Software Design

The flow charts have been shown in Figure 6, Figure 8 and Figure 7. The details are shown clearly in the github repository, all the code are well commented.

The following is an example of pseudo code of define a tensorflow class of a model.

Algorithm 1 Class of the Residual Attention Model

```
1: function $\_\_init\_\_(config)$
2:     $conv_0$ = layers.Conv2D
3:     $residualunit_0$ = layers.ResidualUnitBetween
4:     $stage_1$ = AttentionModule
5:     $residualunit_1$ = layers.ResidualUnitBetween
6:     $stage_2$ = AttentionModule
7:     $residualunit_2$ = layers.ResidualUnitBetween
8:     $stage_3$ = AttentionModule
9:     $residualunit_3$ = layers.ResidualUnitBetween
10:    $bn$ = layers.BatchNormalization
11:    $flatten$ = layers.Flatten
12:    $fc_1$ = layers.Dense
13:    $fc_2$ = layers.Dense
14: end function
15: function $call(inputs, training)$
16:    out = $residualunit_0(conv0$(inputs))
17:    out = $residualunit_1(stage_1$(out))
18:    out = $residualunit_2(stage_2$(out))
19:    out = $residualunit_3(stage_3$(out))
20:    out = $avepooling(bn$(out))
21:    flatten = $flatten$(out)
22:    logits = $out(fc_2(fc_1$(flatten)))
23:    return logits
24: end function
```

5 Results

5.1 Project Results

For this project, we mainly reimplemented the Attention56 on both CIFAR-10 and CIFAR-100. As shown in Figure 9, this is the training and validation accuracy of Attention56 on CIFAR-10. The best validation accuracy is 0.78.

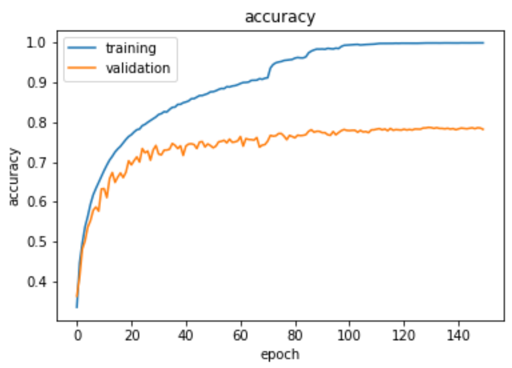
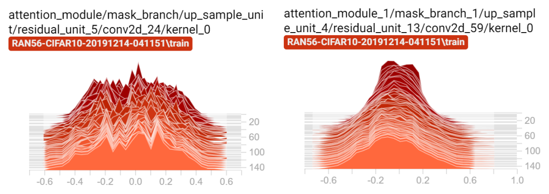


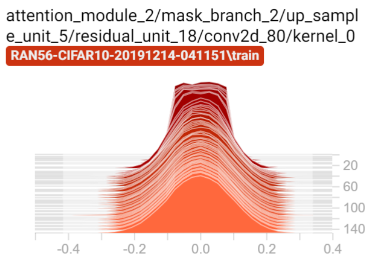
Figure 9 **An Example of training and Validation history of Accuracy.**

During the hyper-parameters tuning, the best validation accuracy on CIFAR-10 in 0.85 and 0.63 on CIFAR-100.

We also exam the weights of the last convolution kernel of mask branch at each stage as in Figure 10. We thought that the gradient back propagate well until before the first stage.



(a) Stage 1 (b) Stage 2



(c) Stage 3

Figure 10 **The weights in the last layers of the three attention module.**

5.2 Comparison of Results

As we can not get the ImageNet dateset, we can only compare the result on the CIFAR-10 and CIFAR-100.

Comparison of Results			
Models	params $\times 10^6$	CIFAR-10	CIFAR-100
Our	≤ 1.9	0.85	0.63
Paper	2.7	0.94	-

The most critical problem for us is over-fitting shown in the follow Figure 11. It's may not clear in the Figure 9 of accuracy but is obviously indicated by the loss. All the methods we learnt from class like regularization, reducing the capacity, dropout and so on has been tried, but can still reduce this kind of over-fitting.

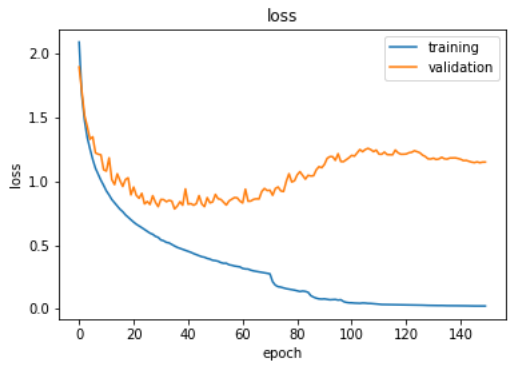


Figure 11 **An Example of training and Validation history of Loss.** The yellow line is for validation loss, the over-fitting is clearly shown.

5.3 Discussion of Insights Gained

Personally, this kind of structure is too flexible and required an elaborated setting of parameters, even the idea is quite easy to understand. The two dataset of the paper used is CIFAR-10&100 and ImageNet. The project is done on CIFAR. The whole structure of network is following the paper, i.e the hyper-parameters related to the structure is set to be exactly same but other parameters and kernel size, stride size, epochs, etc. are not shown in the paper so we set by ourselves.

The main problem is our model over-fitting a little bit. And thus our model's accuracy is about 0.1 lower than the original paper.

6 Conclusion

We exam the idea of the residual attention which is a simple combination of attention and skip connection. At least for our project, **we don't think this is a good idea for the CIFAR dataset**, because first the dataset is small and most of the time, one picture for one object which means there is not so many things can distract the neural network. Therefore, attention maybe useless and the increase of number of parameters can even cause over-fitting.

We also don't think this is a good interpretation of the idea of attention in image classification. The attention means soft mask in this paper which is quite different from that in the machine translation. First, no one argues that machine understand the picture in the same way as people and we do not know thoroughly what is the features extracted in the higher layer. Softmax is such rude way to intervene the features generated by the previous layers.

In the future, we tried to release the over-fitting further and exam the different interpretation of "attention" on image classification.

7 Appendix

7.1 Github Repository

https://github.com/cu-zk-courses-org/e4040-2019Fall-Project-SJST-ss5593-fs2658-yt2633

7.2 Individual Student Contributions

Shengjie Sun (ss5593)

Fraction: $5/12$ What I did:
Code the `residual.py`, `attention_module.py`, `models.py`, `residual_attention.ipynb`.
Report writing, model training and hyper-parameter tuning.

Yiming Tan (yt2658)

Fraction: $4/12$ What I did:
Code the `attention_module.py` and `models.py`.
Report writing, model training and hyper-parameter tuning.

Feng Su (fs3658)

Fraction: $3/12$ What I did:
Report writing.

7.3 Additional Diagrams

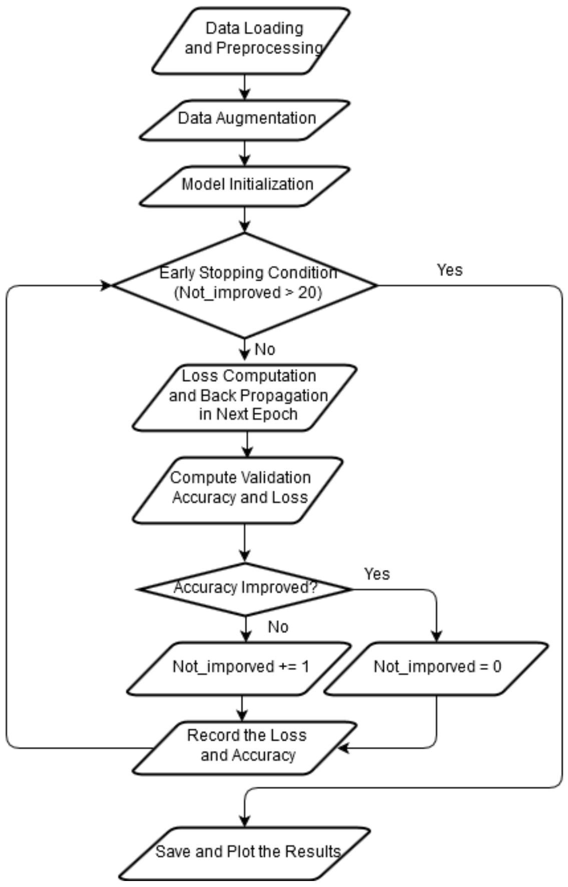

Data Loading and Preprocessing
Data Augmentation
Model Initialization
Early Stopping Condition (Not_improved > 20)
Yes
No
Loss Computation and Back Propagation in Next Epoch
Compute Validation Accuracy and Loss
Accuracy Improved?
Yes
No
Not_imporved += 1
Not_imporved = 0
Record the Loss and Accuracy
Save and Plot the Results

Figure 12 **Training Details.**

References

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2017.