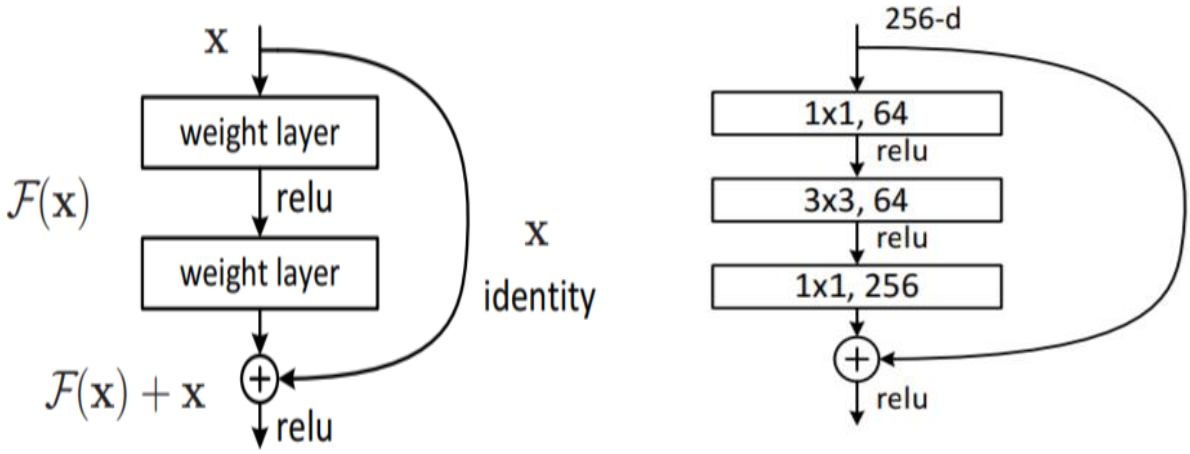
EE569 Digital Image Processing

EE569 Competition --- CIFAR10 Classification

Zheng Wen
zwen1423@usc.edu

Motivation and Logic behind Design

In my approach, the model described in [1] called Residual Attention Network is used. In comparison with LeNet-5, this network is mainly different in two aspects.



(a) Residual Block[2] (b) "Bottleneck" Building Block[2]

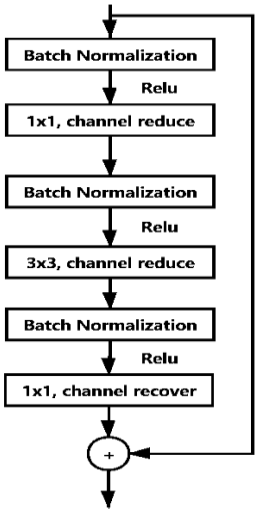


Fig. 1 Structure of Residual Block

Firstly, this network is much deeper than LeNet-5, there are more layers to extract features better, and the residual block structure in Fig. 1(c) is used. Even though deeper network could learn to fit more complex function, and thus extract features better with more hidden layers, the problem is gradient vanishing, which means the value of gradient is getting smaller with respect to weights when the error is transmitted to earlier layers by backpropagation, making the weight in earlier layers be updated much

slowly and thus under-fitted. Another problem is the redundancy caused by deeper layers. When the network goes deeper, there may be some layers are redundancy, which means the former layers have fitted the dataset well, the deeper layers could not extract additional information and thus the output of the redundant layers is expected to be identity. However, because of the random initialization and the different number of filters, it is hard for the redundant layers to become identity transformation, and thus the performance of the whole network degraded because of the deeper structure. To solve these two problems, residual network is proposed in [2]. The residual network is mainly composed by residual block, the structure is shown in Fig. 1(a). Unlike LeNet-5 where all layers are connected sequentially, there are two paths connecting to the output in each residual block, one is the normal one like LeNet-5, the input is convolved with some kernels and activated by some activation functions, the other is the shortcut connecting the input to the output directly. As illustrated in Fig. 1(a), suppose the input is x, the output after convolution layers is F(x), so the total output is F(x) + x. From the aspect of image classification, the more information given to the classifier, the higher accuracy it could obtain. So, for the structure like LeNet-5, the best output after convolutional layers is the input x, which means if there is no shortcut in the residual block, the optimal output of F(x) is F(x) = x. But with the shortcut, the optimal output of convolutional layers becomes F(x) = 0, such that the holistic output of the residual block would become F(x) + x = x, which is the same as previous conclusion, but the optimization is much simpler than the normal network. Even there are some redundancy hidden layers, the identity transformation is much easier to obtained. In residual block, the gradient could not only propagate through convolutional layers, but also propagate through the shortcut, through which the gradient could be fully exploited without any degrading. Another advantage of residual block is that it offers additional information to the output through the shortcut, the accuracy could thus be improved, so the effect of residual block is better than the basic structures in LeNet-5 for the deeper network. Another "Bottleneck" Building Block[2] is shown in Fig. 1(b). This block is originally designed for ImageNet with less parameters, leading to less training time and deeper network structure possible. In my approach, this structure is referenced to reduce the number of parameters. In my approach, the structure of the block is refined according to Fig. 1(b), shown in Fig. 1(c). Because CIFAR-10 is a small dataset, there is totally no need to use a deeper network to get a better accuracy, the main reason using residual block is to get more information from each input. Because the dataset is smaller, so some batch normalization layers[3] is added into the residual block to prevent overfitting and speed up the training process, so more training epochs are possible. The first activation behind the first batch normalization layer is indeed the activation for the output of the former residual block, the only difference is in Fig. 1(c), the output of a layer is batch normalized before activation. The convolution layers with channel reduce means that the output channels are reduced to one forth in comparison with the input feature map, for example, if the input feature map has 64 channels, the number of filters in the channel reduced convolutional layer is 16, just for parameter reduction. The convolution layers with channel recover means the output channels is recovered to be the same with input feature map to make the addition operation possible. The structure above is stated in [1].

Secondly, the structure called Attention Module[1] is set up with residual block, as shown in Fig. 2. There are two parts in a single attention module, the upper part is a stack of blocks set up by residual block, which is commonly used in residual network, called Trunk Branch. The lower part is called Soft Mask Branch, it has a similar structure like Fully Convolution Network (FCN)[4], FCN is designed for image segmentation, so it could be used to form the region of interest in image classification, making the Trunk Branch more focus on the region of interest. The feature extracted by the Soft Mask Branch is activated by Sigmoid function to form the attention region, with value between 0 to 1. To avoid the Soft Mask

Branch breaking the original feature extracted by Trunk Branch, the output is combined with the original Trunk Branch output and the multiply of attention region and the original output In the Soft Mask Branch, there are also a block called residual unit connecting the feature map with the same size from the process of downsampling and the upsampling process, to give more information to the final feature map. In different stage of attention module, because the input size of feature map is different, so the number of residual units is also different.

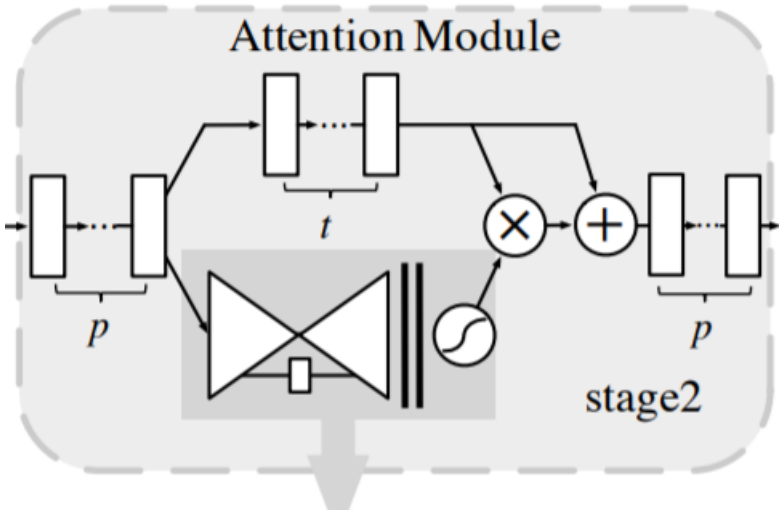


Fig. 2 Attention Module (stage 2)[1]

In my approach, in stage 1, there are two residual unit, in stage 2, there are one residual unit and no residual unit in stage 3. Because the size of input image of CIFAR-10, there is only three stage of attention module. The different stages of attention modules are connected by residual blocks with different output channels. The structure of the whole network is shown in Fig. 3. There is no fully connected layer is used to reduce the number of parameters. The number in the brackets behind convolution layers denotes the size of convolution kernel and the number of kernels, and the brackets behind residual block and attention module denotes the number of output channels of these blocks.

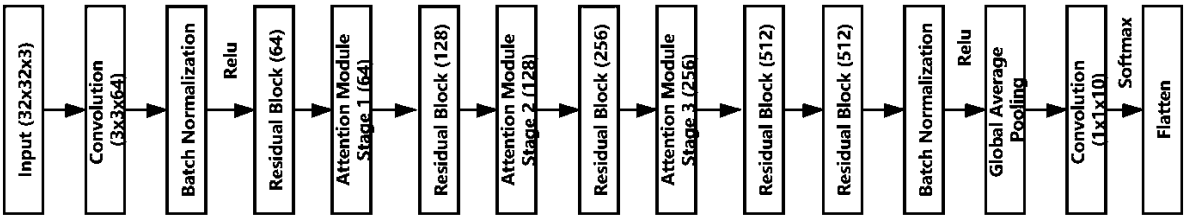


Fig. 3 Residual Attention Network for CIFAR-10

Some other tricks are also used during the training process, these ideas come from [5].

One of these tricks is data augmentation. Data augmentation is commonly used in deep learning, especially in CIFAR-10 classification where the number of training samples is small. Through data augmentation, each image is flipped, rotated, translated or stretched to some degree uncertainly. In this way, more training data could be fed into the network. Even though these images are similar in human

eyes, they are different to the network. So, the problem of overfitting could be overcome, the model will have a better generalization and the accuracy could become higher. In my approach, besides some basic geometrical transformation like flipping and stretching, a method called cutout[6] is used to get a better performance. The basic idea is to use some square with random noise to block out some area of the original image. When encountering an input image, the image could be preprocessed with cutout with a certain probability. The cutout square is randomly generated with random function, as well as the position of each block. From the experiment in [6], the performance of CIFAR-10 could be improved by 1%-2%. A plot illustrating the principle from [6] is shown in Fig. 4. In my implementation, the probability of cutout is set to 0.5, and there are two hyperparameters to decide the ratio between the erased part and the original image called s, and the aspect ratio of the erased region called r, all of these two hyperparameters are set according to [6], where s is a random number between 0.02 and 0.4, r is a random number between 0.3 and 1/0.3.

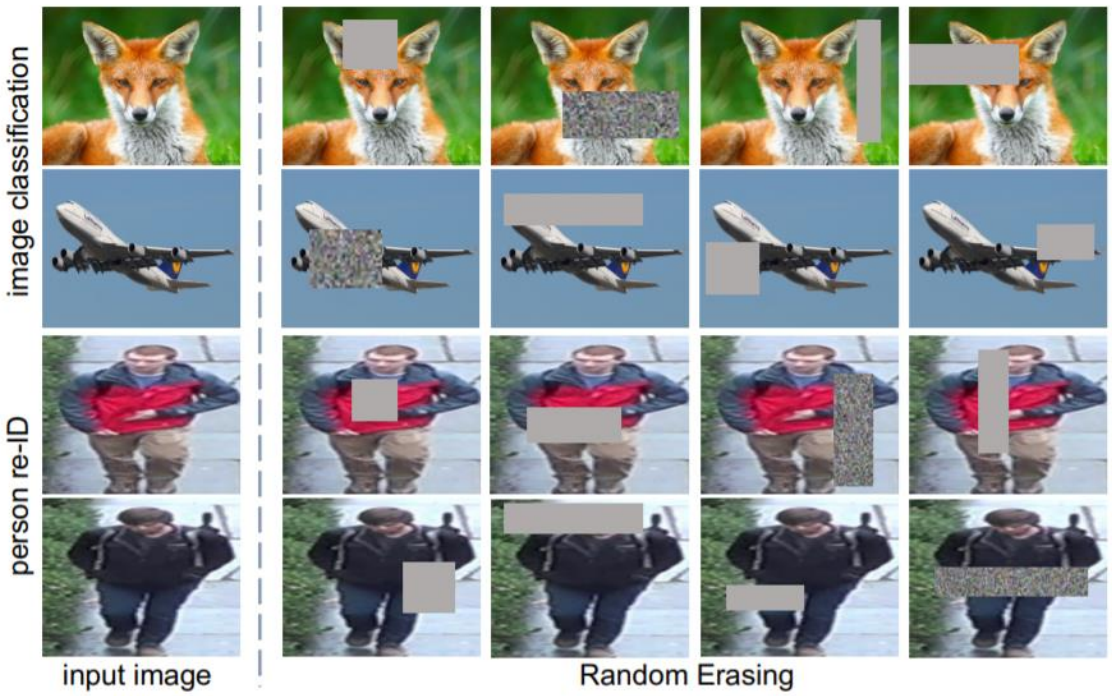


Fig. 4 Images with Cutout

Another trick called cosine decay with warmup[4] is also used to schedule the learning rate. In this method, the learning rate is scheduled as a cosine function, decaying with the steps growing. Warmup is another strategy to prevent the network from overfitting for the initial several epochs, which means that in the beginning of training, the learning rate is set to an extremely small value and grow as the step (batch) growth. In this way, because of the smaller initial learning rate, the distribution of weight could become more stable, which is good for the later training process. From [4], the effect of combining these two tricks could make the performance of the network better. In my implementation, the core function is referenced from TensorFlow official Github[7] and implemented by myself because the original code is designed for tensorflow1.x, it is hard to import it using Keras directly. The number of total epochs is 250, the number of warmup epochs is 25, the base learning rate is set to 0.001, the batch size is set to 64, the curve of the learning rate in my implementation is shown in Fig. 5.

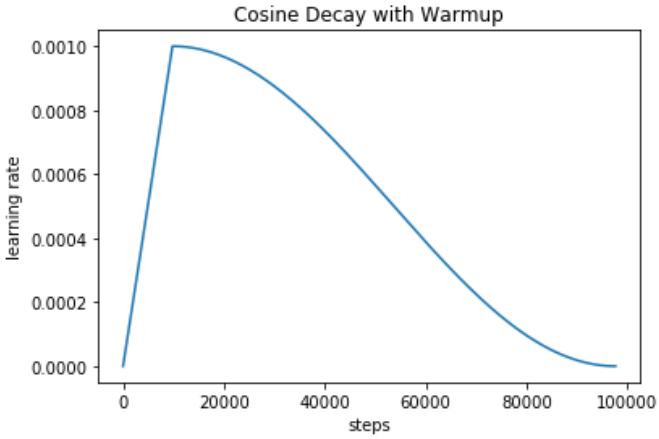
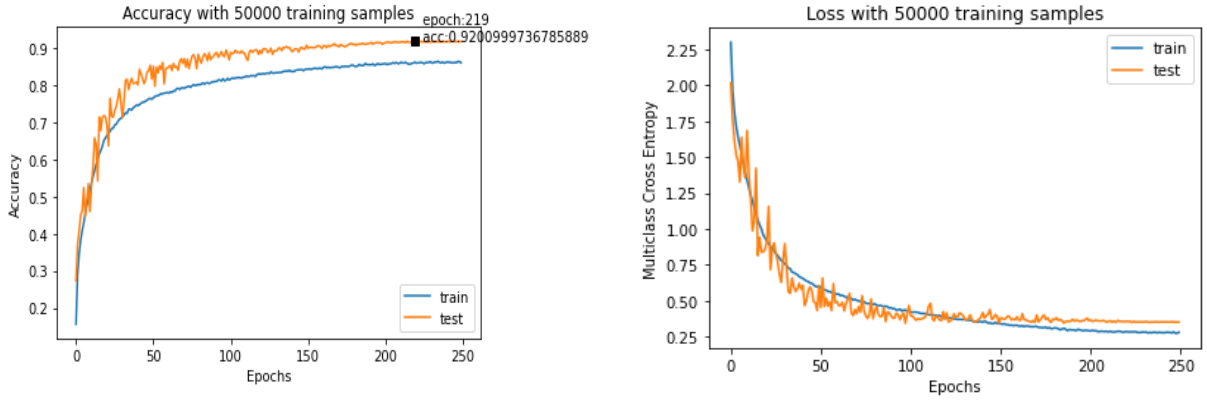


Fig. 5 Learning Rate Used in my Implementation

From the description above, the performance improvement mainly coms from the deeper network structure, the residual block, and the attention module, their advantages are stated above. Besides, data augmentation like cutout and learning rate schedule could also help to improve the performance.

Classification Accuracy and Model Size

The best accuracy I could achieve is 94.56% with the same structure as Fig. 3 except that the number of channels is fourfold of that. The number of parameters is about 7.2 million. Considering the trade of between the number of parameters and the accuracy, the structure shown in Fig. 3 is finally adopted. The number of parameters is **396,810** with 389,258 trainable and 7,552 non-trainable. The highest accuracy is **92.01%** with all training samples, obtained in 219 epoch. The training curve is shown in Fig. 6. The training time is 33s/epoch on Nvidia Tesla P100 GPU offered by Google Colab Pro, the total training time to achieve such a model is 8250 seconds, about 2.3 hours. Because of the data augmentation and the dropout layer at the end of the whole model preventing overfitting, accuracy on the augmented training set is lower than that on test set. After the whole process of training, the accuracy on the original training set is tested, the result is 99.59%. The inference time on training set is 11 second on Nvidia Tesla P100 GPU offered by Google Colab Pro, the inference test time with the same GPU is 2 second.



(a) Training Accuracy

(b) Training Loss

Fig. 3 Training Curve

To show the performance degrading as the number of training sample is reduced, the number of training samples is reduced every 1000 images randomly by class and for each reduced dataset, the model is trained for 100 epochs because the later epochs could only increase the accuracy little by little, the accuracy in the first 100 epoch could achieve 89%, the last 150 epoch only increase the accuracy by 3%. Because of the learning rate scheduler, if the model is trained with 50000 samples for 100 epochs, the performance will remain similar, more epochs could only make the performance more stable and have a little impact on the final accuracy. So, training with 100 epochs is enough to show the trend. The number of warmup epochs is also reduced to 10 accordingly. All of the other hyperparameters are not changed. The training accuracy under different number of training samples is shown in Table. 1. The training accuracy is evaluated from the reduced training set, while the test accuracy is evaluated from the whole test set. The plot of test accuracy as the number of training sample degrades is shown in Fig. 4. The full training process is attached in jupyter notebook in the zip file, also available in my Colab. The link is attached below.

Highest accuracy: https://colab.research.google.com/drive/1DUUowry9EtOYgjf0GlNHDG-InmloJG6d

Ablation study: https://colab.research.google.com/drive/1sleUY-UIRVnH7bfPK-MWj6UIRWdLv7hy

Training sample number	50000	25000	12500	6250	3120	1560
Training accuracy (%)	99.59	96.43	95.86	95.86	94.38	51.57
Test accuracy (%)	92.01	89.08	87.15	84.61	79.34	94.29

Table 1. Accuracy under different number of training samples

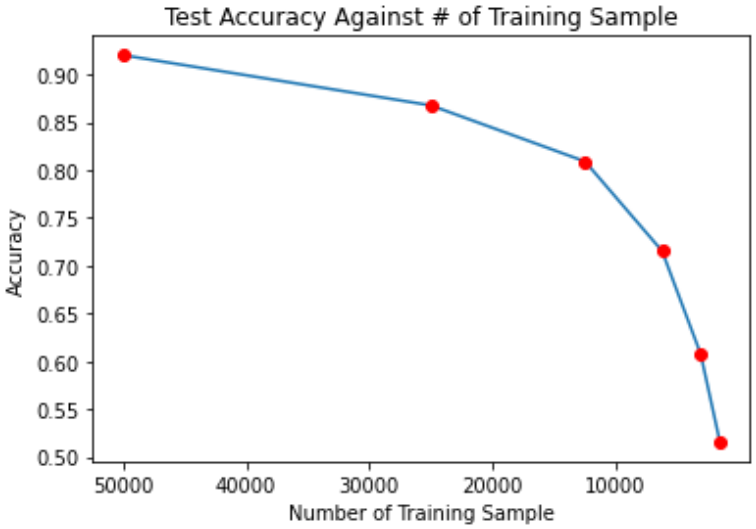


Fig. 4 Test Accuracy with different number of training samples

Reference

[1] Wang F, Jiang M, Qian C, Yang S, Li C, Zhang H, Wang X, Tang X. Residual attention network for image classification. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017 (pp. 3156-3164).

[2] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 770-778).

[3] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167. 2015 Feb 11.

[4] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. InProceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 3431-3440).

[5] He T, Zhang Z, Zhang H, Zhang Z, Xie J, Li M. Bag of tricks for image classification with convolutional neural networks. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2019 (pp. 558-567).

[6] Zhong Z, Zheng L, Kang G, Li S, Yang Y. Random erasing data augmentation. arXiv preprint arXiv:1708.04896. 2017 Aug 16.

[7]https://github.com/tensorflow/models/blob/master/research/object_detection/utils/learning_schedules.py