RESNET-modified with Architectural Changes

Rohit Kumar Sah
Department of Computer Science
Texas A&M University
College Station, TX 77843
`rohit.sah@tamu.edu`

Abstract

Resnet models brought a revolution to the field of image classification and highly improved the accuracy of such classifications. Resnet model allows much deeper networks to train efficiently. ResNet also solved the vanishing Gradient problem with some of the earlier models, which is what makes it effective in comparison to models such as AlexNet or VGGNET. This project takes ResNet as basic model and focuses on improving the accuracy through some of the architectural changes.

1 Introduction

There are multiple combinations of things which could improve the accuracy of the Resnet model. Out of all those combinations, model architecture and the quality and size of data can highly impact the efficiency of Deep Learning models. Experiments on these two are detailed in next sections. This project implements some of the architectural suggestion from [1] Bello et al work on Improving Training and scaling strategies.

2 Model Architecture

This project uses **ResNet50** architecture as base and then implements the below 2 changes to the model structure.

2.1 ResNet-D[1]

- In the stem block, 7X7 block is replaced by 3 smaller 3X3 convolutions.
- The strides are switched for first 2 convolutions.
- the stride-2 1×1 convolution in the skip connection path of the downsampling blocks is replaced by stride-2 2×2 average pooling and then a non-strided 1×1 convolution.
- the stride 2 3×3 max pool layer is removed and the downsampling occurs in the first 3×3 convolution in the next bottleneck block.

Preprint. Under review.

2.2 Squeeze and Excitation[2]

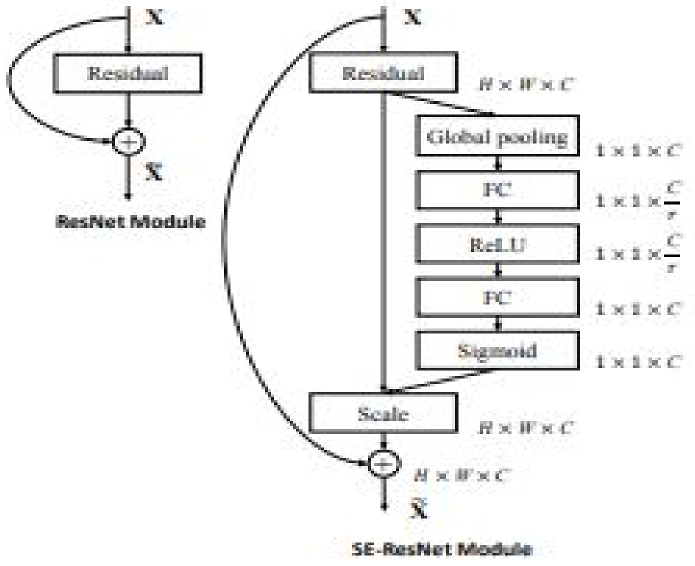

Figure 1: Original ResNet (left) and Squeeze and Excitation block(right)

SE blocks are architectural units that are designed to improve the representational power of a network by enabling it to perform dynamic channel-wise feature recalibration. This project uses Squeeze and Excitation ratio of 0.25

For optimizer this project is using Stochastic Gradient Descent with nesterov momentum set to true and momentum value set as 0.9

3 Data Augmentation

Real-life images are sometimes different only in terms of position and angle rotation or the relative size of the object. Data Augmentation techniques are used to add variability to the dataset such that the model can learn these real life images better. This project uses random augmentation from multiple combinations of cropping, rotation, flipping, changing sharpness, contrast and color and some other translations. The random augmentation performs better and gives a boost to accuracy of the model. Some of the policies mentioned in Daniel Ho et al [3] paper on Population based Augmentation gives good baseline policies for the implementation.

3.1 Cutout

Cutout is a very commonly used regularization technique and involves removing portions of image at random points. This effectively simulates the occluded image cases and helps in improving the learning capacity of the model. Using this has significantly improved the accuracy of this model. This project creates 3 Cutouts of each having a size of 4 is made in all of the training images.

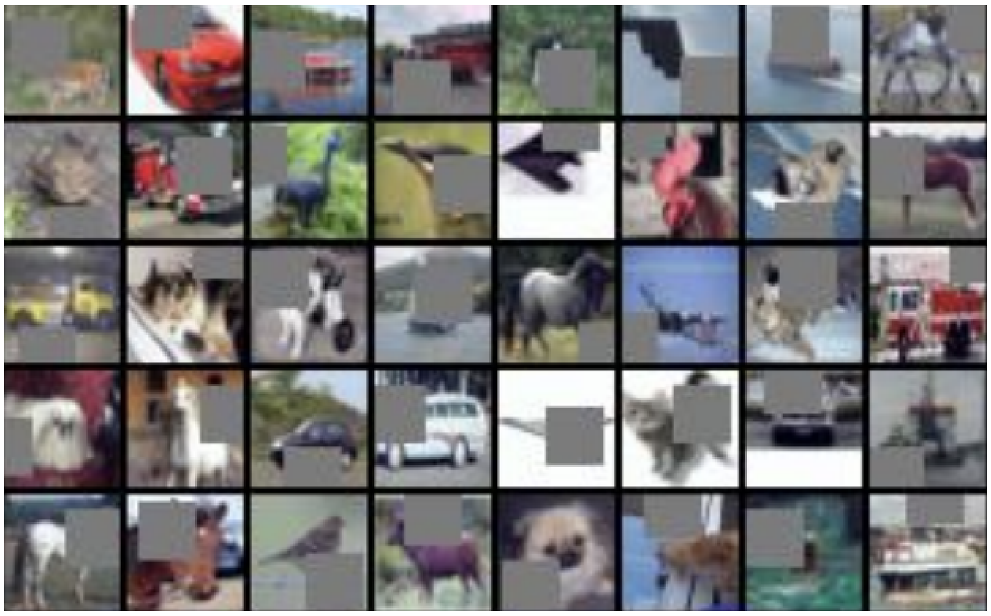

Figure 2: Example of cutout for CIFAR10 dataset[4]

4 Architectural Improvements

4.1 Exponential Moving Averages

The moving averages are computed using exponential decay. The averages of previous observations are taken to compute the moving average. Moving average is used to stabilize the training. It gives more weightage to neighbours when computing averages. It does not however improve the accuracy much.

4.2 Dropout

Apart from earlier defined cutout, there is one more regularization techniques that helps in improving the learning of the model. In this case instead of spatial section drop, some number of units(neurons) are dropped. Cutout helps in preventing over-fitting. Dropout ratio of 0.25 is used for this model. For testing, during every iteration ¼ of total neurons are dropped for each layer for a training sample. The same thing happens with activations too. Figure[2] gives an example of network where some of the activations and units are dropped randomly.

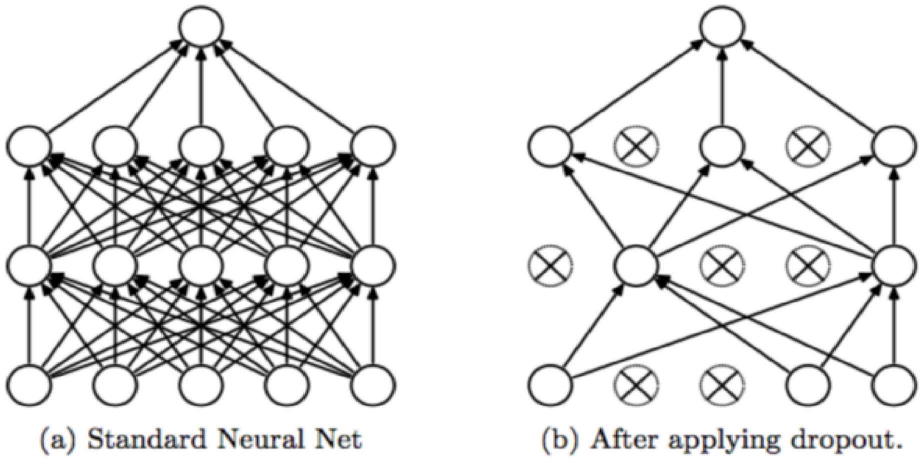

Figure 3: Example of dropout.[5]

4.3 Cosine Learning Rate

It is imperative to find a good value for the learning rate for the model on the training dataset. Also, this should not remain constant, which stagnates the learning quite early in training. This project uses Cosine Annealing for changes to learning rate. Cosine Annealing starts with a higher learning rate and is then sharply reduced to a very micro value and is reset again after some interval. This step happens again and again. These warm restarts are set after each 50 iterations.

5 Results

This model is trained on Pytorch version 1.7.0. Experiments with multiple hyperparameters

Table 1: Hyperparameter table

learning rate	weight decay	Reduction ratio	Stem width	cutout length	cutout holes
0.001	0.0001	4	64	8	1
0.1	**0.00005**	**16**	**128**	**4**	**3**
0.01	0.00005	4	64	8	1

Experiments were done also to compare Multi-Step Learning Rate and Cosine Learning Rate and the latter performs much better. This improved the accuracy by **2.1%** Increasing the stem width from 64 to 128 gives **+1.5%** accuracy.

Out of these the model with **learning rate = 0.01, weight decay = 0.00005, stem width = 128, cutout length = 4 and cutout holes = 3** performs better than other models.

This model reached **Training accuracy of 94.32** and **test accuracy of 91.11** when ran for **230 epochs**. I've used **Kaggle** to train these models and **each epoch takes about a minute**.

6 Conclusion

This project shows that some of the architectural improvements could improve the original architecture efficiency. There are multiple combinations which serves this purpose. This project is in no way the best out of all those, but it tries to learn on one such combinations.

7 References

[1] Irwan Bello, William Fedus, Xianzhi Du, Ekin D. Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, Barret Zoph arXiv 2103.07579 Revisiting ResNets: Improved Training and Scaling Strategies

[2] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu arXiv 1709.01507v4 Squeeze-and-Excitation Networks

[3] Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, Xi Chen (2019) Population Based Augmentation arXiv 1905.05393v1 Efficient Learning of Augmentation Policy Schedules

[4] Terrance DeVries, Graham W. Taylor https://arxiv.org/abs/1708.04552v2 *Improved Regularization of Convolutional Neural Networks with cutout.*

[5] https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5