Mini-Project 1: Residual Network Design

Mohammad Shamoon
ms12736@nyu.edu

Umesh Deshmukh
urd7172@nyu.edu

Aradhya Alamuru
aa9405@nyu.edu

Abstract

In mini project 1, a custom ResNet has been designed and implemented, testing on the CIFAR-10 dataset under a constraint of 5 million parameter count, while implementing data augmentation and introducing normalization, resulting in an accuracy of 87.5% on testing data.

1 Introduction

One of the most commonly used models for image classification tasks are ResNets. In mini project 1, the task is to design and train a ResNet model for CIFAR-10 image classification. The goal is to maximize accuracy while keeping the size of the ResNet model under 5 million parameters.

2 Implementation

2.1 Dataset and Preprocessing

The CIFAR-10 dataset has 10 classes, 50,000 training images and 10,000 testing data images. In preprocessing, the mean and the standard deviation on the dataset is calculated, across three channels, i.e. RGB. Random Crop (32 x 32) and Random Horizontal Flip methodologies are used to augment the data. The data is normalized using the mean and the standard deviation that was calculated previously. The data is then split into training data and validation data in the ratio of 80:20.

2.2 Residual Network

Based on the plain network, shortcut connections are inserted, turning the network into the residual version. x and y are the input and output vectors of the layers considered. Function $F(x, W_i)$ represents the residual mapping to be learned.

$$y = F(x, W_i) + x$$

W_s is used to match the input and output dimensions, where it does not add any additional parameters or computational complexity.

$$y = F(x, W_i) + W_s x$$

When the shortcuts go across feature maps of two sizes, they are performed with a stride of 2.

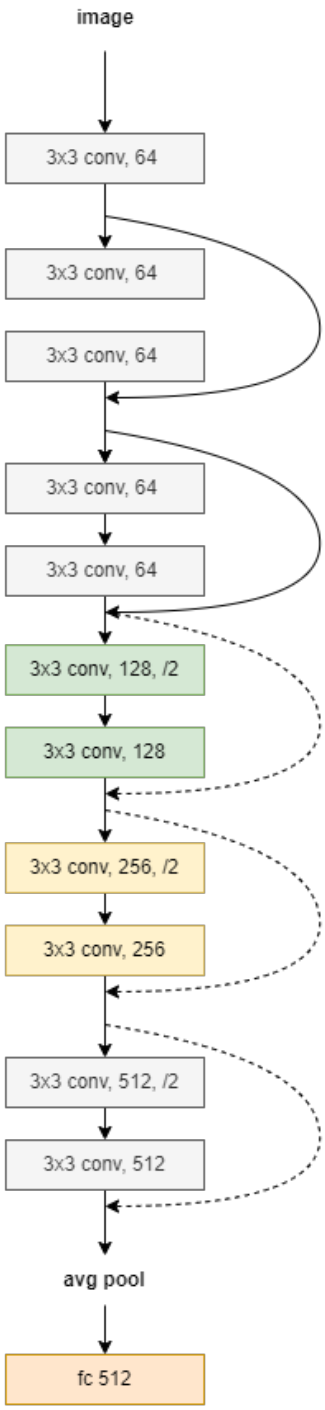
2.3 Optimization

ADAM optimizer is used with a learning rate of 0.001 and β_1 of 0.9 and β_2 of 0.999.

Dropout regularization scheme is used in the fully connected layer to avoid over-fitting. If dropout layers are not present, the first batch of training samples influences the learning in a disproportionally high manner.

The batchsize of 128 used for training, testing and validation of data. Training is performed for 20 epochs.

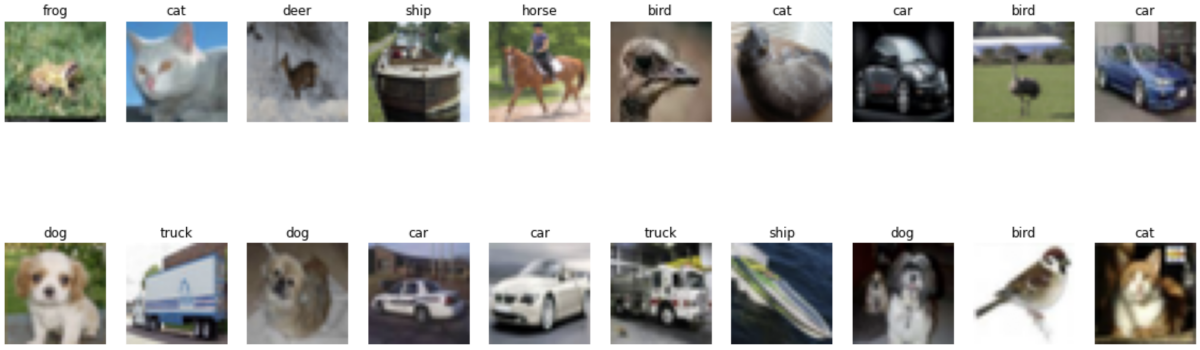
Figure 1: Architecture of the ResNet



2.4 Code

The Python code for the ResNet implementation and the model weights can be found on the GitHub Link. [1]

Figure 2: CIFAR-10 Dataset



3 Methodology

The original ResNet-18 model has 11 million parameters and has four layers, and each layer has 2 basic blocks. Our approach was designed with a single block per layer. Resulting in network width with 4.7 million trainable parameters.

Then to improve it, the number of blocks in the first layer was increased to 2. This resulted in an improvement in accuracy.

Due to the constraint on total number of parameters to be under 5 million, the number of blocks in the other layers cannot be increased.

The training dataset is the sample of data that is used to fit the model. The testing dataset is used to provide unbiased evaluation of model fit while tuning trainable parameters of the model. As the model is trained and tested, it becomes more biased towards the testing data. The evaluation becomes more biased as skill on the validation dataset is indirectly incorporated into the model.

Validation provides the model with a good generalization strategy, which is used primarily to provide an unbiased evaluation of a final model fit on the training dataset.

Data augmentation is chosen and implemented to provide variability to the data set to improve the robustness of the model. For the data augmentation, geometric transformation of image flip with a probability of 0.5 and random crop is used. This has been proven to improve the accuracy in CIFAR-10 dataset and ImageNet dataset.

Introduced normalization by implementing batch normalization and dropout, where the set of activations in a layer are normalized. This is a standard technique in the preprocessing of pixel values. The following are the values for mean and standard deviation.

Channels	**Red**	**Green**	**Blue**
Mean	0.49139968	0.48215827	0.44653124
Standard Deviation	0.24703233	0.24348505	0.26158768

Image of size 32x32 is provided as an input to the convolutional layer with a kernel size of 3, stride of 1 and padding 1, with 64 feature maps. This will then be passed into the batch normalizations. This is then passed through a ReLU layer and subsequently through 4 residual layers with feature map size of 64, 128, 256, 512 respectively, and then finally through a fully connected layer of 512 neurons, which produces an output of 10 classes.

While training the model on the CIFAR-10 dataset, the training accuracy, validation accuracy, overall training accuracy and overall validation accuracy through out the training session are calculated and the model is saved when the validation loss is lesser than the previously calculated validation loss of the best model.

4 Results

The custom ResNet model designed and trained on the CIFAR-10 dataset, resulted in a test accuracy of 87.5%.

Model	Accuracy
Test Loss	0.5639
Test Accuracy	87.5
Validation Accuracy	86.55

Figure 3: Train vs Validation Loss

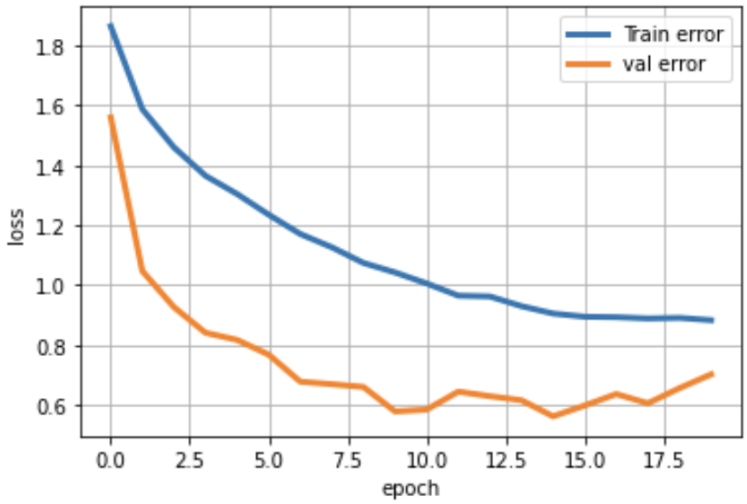


Figure 4: Train vs Validation Accuracy

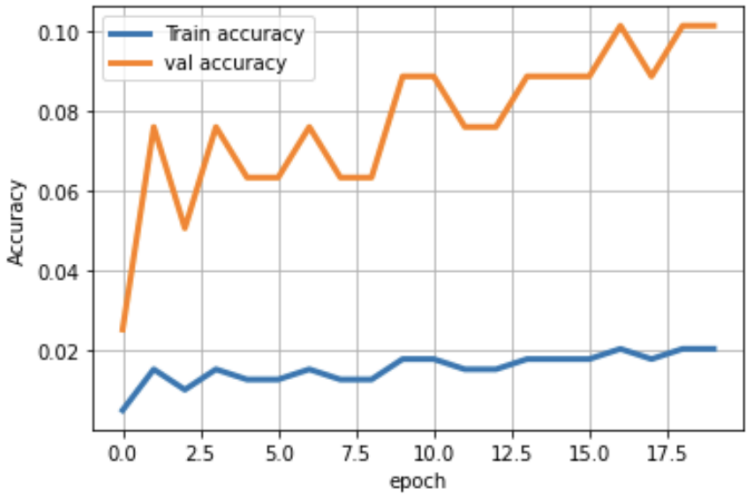


Figure 5: Trainable Parameters

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,728
       BatchNorm2d-2           [-1, 64, 32, 32]             128
            Conv2d-3           [-1, 64, 32, 32]          36,864
       BatchNorm2d-4           [-1, 64, 32, 32]             128
            Conv2d-5           [-1, 64, 32, 32]          36,864
       BatchNorm2d-6           [-1, 64, 32, 32]             128
        BasicBlock-7           [-1, 64, 32, 32]               0
            Conv2d-8           [-1, 64, 32, 32]          36,864
       BatchNorm2d-9           [-1, 64, 32, 32]             128
           Conv2d-10           [-1, 64, 32, 32]          36,864
      BatchNorm2d-11           [-1, 64, 32, 32]             128
       BasicBlock-12           [-1, 64, 32, 32]               0
           Conv2d-13          [-1, 128, 16, 16]          73,728
      BatchNorm2d-14          [-1, 128, 16, 16]             256
           Conv2d-15          [-1, 128, 16, 16]         147,456
      BatchNorm2d-16          [-1, 128, 16, 16]             256
           Conv2d-17          [-1, 128, 16, 16]           8,192
      BatchNorm2d-18          [-1, 128, 16, 16]             256
       BasicBlock-19          [-1, 128, 16, 16]               0
           Conv2d-20            [-1, 256, 8, 8]         294,912
      BatchNorm2d-21            [-1, 256, 8, 8]             512
           Conv2d-22            [-1, 256, 8, 8]         589,824
      BatchNorm2d-23            [-1, 256, 8, 8]             512
           Conv2d-24            [-1, 256, 8, 8]          32,768
      BatchNorm2d-25            [-1, 256, 8, 8]             512
       BasicBlock-26            [-1, 256, 8, 8]               0
           Conv2d-27            [-1, 512, 4, 4]       1,179,648
      BatchNorm2d-28            [-1, 512, 4, 4]           1,024
           Conv2d-29            [-1, 512, 4, 4]       2,359,296
      BatchNorm2d-30            [-1, 512, 4, 4]           1,024
           Conv2d-31            [-1, 512, 4, 4]         131,072
      BatchNorm2d-32            [-1, 512, 4, 4]           1,024
       BasicBlock-33            [-1, 512, 4, 4]               0
           Linear-34                  [-1, 10]           5,130
================================================================
Total params: 4,977,226
Trainable params: 4,977,226
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 9.06
Params size (MB): 18.99
Estimated Total Size (MB): 28.06
----------------------------------------------------------------
```

5 Conclusions

ResNet on CIFAR-10 dataset has been implemented with a final accuracy of 87.5% on test dataset. This accuracy can be increased by removing the constraint on parameters and also by increasing the depth or width of Residual Network.

Hyperparameters	**Values**
Epochs	20
Residual Layers (N)	4
Residual Blocks in Residual Layer i (B_i)	4
Channels in Residual Layer i / Feature maps (C_i)	$[64, 128, 256, 512]$
Convolutional Kernel Size in Residual Layer i (P_i)	4

6 Citations

[1] https://github.com/fatardy/DeepLearning-MiniProject1

[2] Loshchilov, Ilya, and Frank Hutter. *"Sgdr: Stochastic gradient descent with warm restarts."* arXiv preprint arXiv:1608.03983 (2016).

[3] Hinton, Geoffrey E., et al. *"Improving neural networks by preventing co-adaptation of feature detectors."* arXiv preprint arXiv:1207.0580 (2012).

[4] Shorten, C., Khoshgoftaar, *"T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019)."* https://doi.org/10.1186/s40537-019-0197-0

[5] Nitish S, Geoffrey H, Alex K, Ilya S, Ruslan S. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.

[6] Jonathan T, Ross G, Arjun J, Yann L, Christoph B. Efficient object localization using convolutional networks. In: CVPR'15. 2015.

[7] Sergey I, Christan S. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML; 2015.

[8].https://stackoverflow.com/questions/66678052/how-to-calculate-the-mean-and-the-std-of-cifar10-data

[9].https://jamesmccaffrey.wordpress.com/2020/08/07/displaying-cifar-10-images-using-pytorch/

[10].https://linuxtut.com/en/feb9f49cdd8560b97667/

[11].https://discuss.pytorch.org/t/inject-dropout-into-resnet-or-any-other-network/66322