CSCE 636 Deep Learning Project, Image Classification

Sanjay Nayak
Department of Computer Science & Engineering
Texas A & M University
College Station, TX, 77843
sanjaynayak@tamu.edu

Abstract

Deep learning helps to solve different complex problems with relative ease. Degrdation of gradient problem in deep neural networks renders training ineffective. ResNet technique solves the problem of degradation of gradient by addition of input into the output of each residual blocks. Two different block formats, standard block and residual block, were introduced for the framework of deep residual learning. This project uses modified versions of ResNet, Dual Path Network, which uses the concept of ResNet and DenseNet to classify the images. A wider network with a depth of 60 convolution layers is being used that trains on CIFAR-10 public training dataset with an accuracy of 94.65% on the public test dataset of CIFAR-10 [5].

1 Introduction

"Image Classification" is used for categorizing/clustering different images in order to gain relevant information, the concept of which is utilized in Computer Vision. The features from the images are extracted and then a model is trained on those features which then classifies the particular image. One of the way is to apply a filter to an image which forms an activation resulting in a feature map. This technique is termed as Convolution. Neural Network tries to simulate how a brain processes and takes a decision. This helps in more efficient decision making and solving of complex problems. There are different types of deep neural networks - Convolutional Neural network, Recurrent Neural Network, Autoencoder Neural Network, and Feedforward Neural Network. Convolutional Neural Network is widely used for processing of images.

Even though *Deep Convolutional Neural network* has led breakthroughs for the classification of images, there is a major problem of vanishing gradient which obstructs the convergence. This issue was addressed by using normalized initialization and intermediate normalization layers. But when the depth of the networks is increased, a degradation problem was noticed that made the training stagnate by saturating the accuracy. This also resulted in degradation of accuracy. To put it simply, a deeper network should not have accuracy higher than that of its shallower network. This problem was solved by the *Deep Residual Learning* framework. Figure 1 denotes the bottleneck residual block architecture.

Different deep learning frameworks based on the concept of ResNet [1, 3] has been developed. Through various researches, it has been observed that increasing the width of a neural network gives better or same accuracy as very deep neural networks. The main disadvantage of a very deep neural network is the amount of time it takes to train the model. Wide networks helps in the reduction of the training time without compromising the training and testing accuracy. DenseNet is one of the variant where instead of summation of the input feature map with the output feature map of the residual block, the input feature maps are concatenated with the output feature maps produced at different layers that results in better flow of information between layers [4].

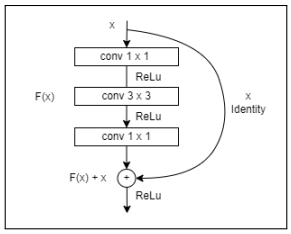

Figure 1: Deep Residual Network: Residual Bottleneck building block

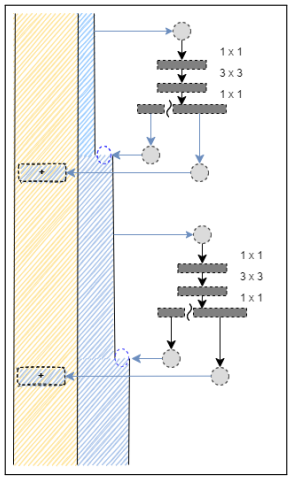

Figure 2: Dual Path Network: Block Architecture, where " $+$ " denotes element-wise addition, " $\wr$ " denotes split operation

2 Dual Path Network

Dual Path Network is another deep learning framework based on the concept of ResNet and DenseNet frameworks. This project uses the concept of Dual Path Network [2]. This section introduces the concept of Dual Path Network, and its architecture.

2.1 Architecture

Dual Path Network uses residual network as a backbone and a thin densely connected path is added to it forming the network. It is generated using stacks of blocks, where each block consists of a bottleneck structure. It consists of a 1×1 convolution layer, which is followed by a 3×3 convolution layer, and followed by a 1×1 convolution layer. The functionality of this individual block is same as that of the bottleneck block of ResNet.

The block architecture of Dual Path Network is presented in Figure 2. The " $+$ " in the architecture refers to the addition of the input to the output of the block as per the ResNet architecture of Bottleneck layer. The " $\wr$ " symbol denotes the split operation, where one part of the split is used in the addition operation of input and output like ResNet, and the other part of split is used for concatenation of input and output like DenseNet.

2.2 Mechanism

The neural model of Dual Path Network technique is made up of multiple stacks, where each stack consists of multiple bottleneck layers. Each stack layer has a particular number of input features or dimensions. The output features includes the output features to be used for ResNet and DenseNet operations. The first 1×1 convolution layer in the bottleneck converts the number of input dimension into desired number of dimension which is then passed to 3×3 convolution layer that increases the dimension to get the final output. Once the desired output dimension is generated from the last convolution layer in the block, the dimensions are split to perform two operations, addition on a

Table 1: Model Architecture

Parameters	Input Layer	Stack 1	Stack 2	Stack 3	Stack 4	Output Layer
Image Size	32×32	32×32	16×16	8×8	4×4	1×1
Number of blocks		3	4	10	3	
Dense Depth		16	32	24	128	
In Channel	3	64	128	256	512	1024
Out Channel	64	128	256	512	1024	10

Table 2: CIFAR-10 class representation

Class	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Label	0	1	2	3	4	5	6	7	8	9

part of output channels as per ResNet and concatenation of the other part of output channels as per DenseNet. This results in the increase in the dimension of the output with respect to the input, which is then passed onto the next block repeating the above operation.

3 Implementation

This section gives an overview of how the deep learning technique of Dual Path Network is implemented for this project.

3.1 Environment

For this project, the framework used is *PyTorch 1.6* and language used is *Python 3.9*. This configuration is used for the whole implementation of the model, its training, testing, and generating the predictions on the private test image dataset. Google Colab Pro with runtime settings of GPU in *hardware accelerator* and High-RAM in the *runtime shape* was used for the implemenatation, training, testing, and predicting.

3.2 Data Load & Data Pre-processing

CIFAR-10 [5] public training dataset consisting of 50000 images was used to train the model. For intial training, the training data was split in a ratio of 0.9 consisting of 45000 images for training and 5000 images for validation to get the best model. Each image of the CIFAR-10 is in the format of $(3 \times 32 \times 32)$. The final prediction was done on the private dataset consisting of 2000 images, which is present in the format of $(32 \times 32 \times 3)$.

For *training data pre-processing*, the image was padded with padding of 4, randomly cropped in size 32×32, and randomly flipped horizontally. Image normalization with mean and standard deviation based on channels was also done.

For *testing and prediction data pre-processing*, the image was only normalized with mean and standard deviation using the channels of the image.

CIFAR-10 data is associated with the following 10 labels corresponding to 10 classes represented by numbers $0 - 9$, which can be referred from Table 2.

3.3 Hyperparameters

The hyperparameters used for the best model can be referred from Table 3

3.4 Implementation

The Dual Path Network network implemented has 4 stack layers with each stack layer having $3, 4, 10,$ and 3 bottleneck layers respectively. Each convolutional layer is followed by a batch normalization and ReLu, non-linear activation function. For this implementation, post activation is used in

Table 3: Hyperparameters

Name	Value
Weight Decay	0.0005
Learning Rate	0.01
Momentum	0.9
Epochs	200
Batch Size	128
Activation Function	ReLu
stack layers	4
Convolution layers	60

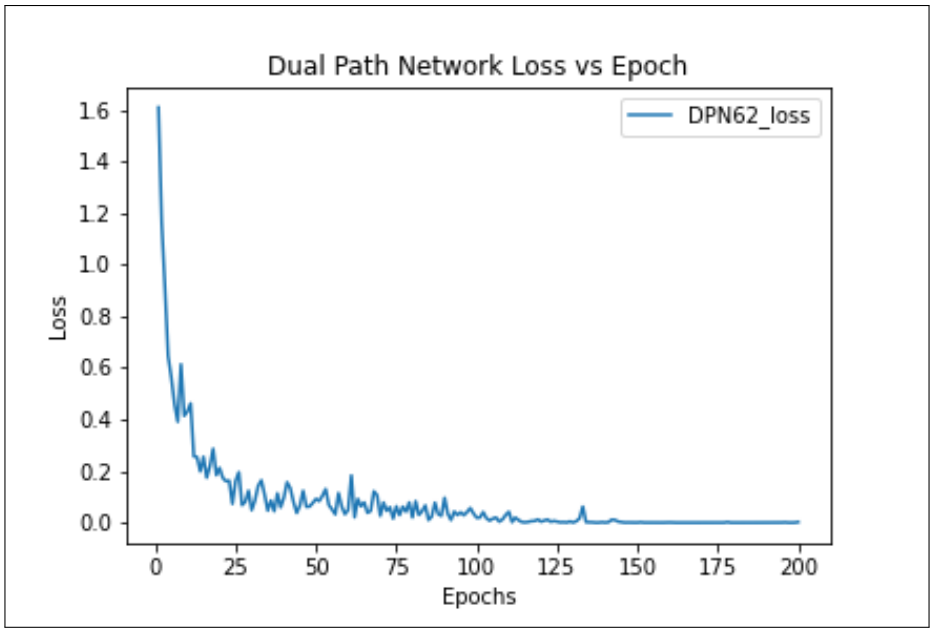

Figure 3: Dual Path Network: Loss vs Epoch

the bottleneck layers. An average pool of kernel size 4 is used followed by a linear fully connected layer. The output generates an array consisting of 10 predictions corresponding to different classes for each of the input image. SGD is used for calculation of loss and learning rate scheduler is used for varying the learning rate after each epoch. The final predictions for the private image dataset consisting of 2000 images is done using the best model as per the hyperparameters of the best model. The Dual Path Network architecture that is followed to create the DPN-62 model can be referred from Table 1.

4 Experiments

The network model is trained with different input and output channels for each layer, different number of layers in each stack layer, different learning rates, and different epochs. The loss, validation accuracy, and test accuracy were used to find the best model. Initial training was done on 45000 training images and validation was done on the rest 5000 images of training. The final model was trained with 50000 images and then tested on the public test dataset of 10000 images of CIFAR-10 for image classification.

Table 4: Model Comparisions

Model	Epochs	Accuracy (%)	Time per Epoch (seconds)
DPN-92	100	92.74	212
DPN-62	100	92.51	114
DPN-62	200	94.65	133

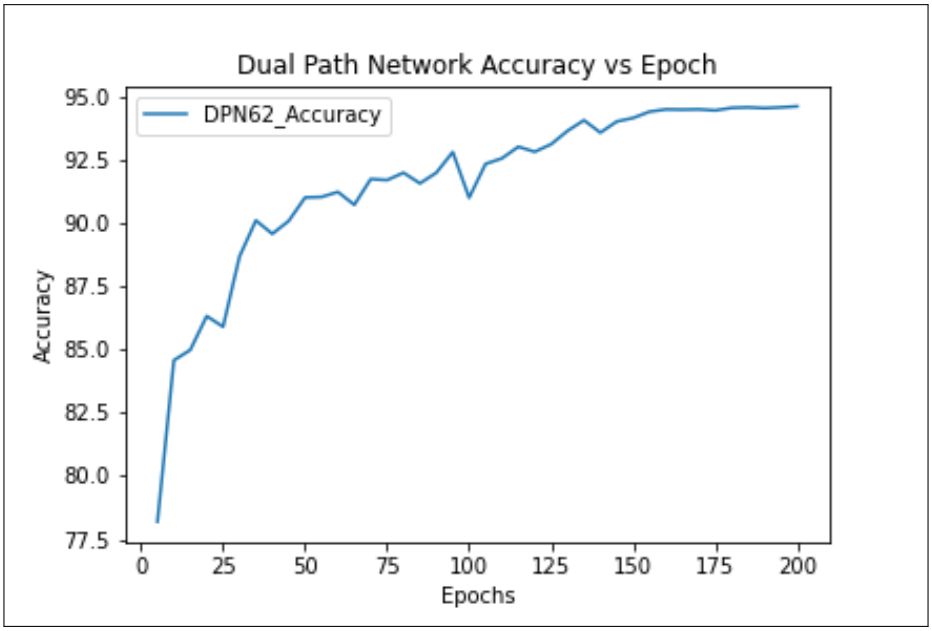

Figure 4: Dual Path Network: Test accuracy vs Epoch

5 Results

All results and graphs were calculated and plotted using the DPN-62 network model. The results indicated that on increasing the number of channels in each stack layer of the network, the loss was getting converged in lower number of epochs but the training time increased significantly. By using half of the channels in each stack layer, the loss took some more epochs for convergence but the overall training time of the model was reduced. For the best model used, the training took around 7.5 hours with each epoch taking around 133 seconds for completion.

It is also noticed that the test accuracy was comparable even when the number of channels was decreased. Figure 3 gives us the graph of loss vs epoch. It can be seen that the loss started from 1.6 and got reduced to around 0.0005 at epoch 200.

Table 4 gives us a view of how the models have performed in different epochs. It can be observed that the per epoch time gets almost doubled without any significant change in accuracy when the number of convolution layers was increased from 60 to 90.

Figure 4 provides details regarding how test accuracy changed with different epochs. Checkpoint interval of 5 epoch was used to get the data for the graph. It can be noticed that the test accuracy almost remained same from epoch 150. From epoch 150 till 200, the increment was in the range of 0.01. A final accuracy of 94.65% was achieved through the model.

6 Conclusion

From the experiments conducted, it can be concluded that on increasing the width or depth of the network, the training time gets increased significantly and increasing depth do not give a significant performance improvement. Rather, with increase in the width of the network, the loss gets converged in less epochs. Data augmentation helps in better training of the model as it provides randomization.

One of the main reasons of using Dual Path Network in this project is it is an interesting concept that uses two different concepts, ResNet and DenseNet, and uses the benefit from both of them.

References

[1] I. Bello, W. Fedus, X. Du, E. D. Cubuk, A. Srinivas, T.-Y. Lin, J. Shlens, and B. Zoph. Revisiting resnets: Improved training and scaling strategies, 2021.

[2] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks, 2017.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[4] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2018.

[5] A. Krizhevsky. Learning multiple layers of features from tiny images, 2009.

A Appendix

Figure 5 represents 200 randomly sampled training images from CIFAR-10 dataset, denoting the types of images used for training the model.

Figure 5: Randomly sampled 200 CIFAR-10 training images

B Appendix

Table 5 represents the epoch loss and accuracy in that epoch on the public test dataset of the DPN-62 model.

Table 5: Epoch loss & accuracy

Epoch	Loss	Accuracy (%)
10	0.431234	84.59
20	0.212238	86.34
40	0.097001	89.60
80	0.020342	92.02
120	0.004022	92.85
160	0.000990	94.53
180	0.000378	94.59
200	0.002423	94.65