Manarat International University

Department of Computer Science and Engineering

Neural Networks & Fuzzy Systems (CSE-433)

Project Report

On

CIFAR-10(Object Recognition Challenge)

Team Name : GirlsPower

Team Members:

1. Raisa Akter (1640CSE00515)
2. Kazi Tanjee Tamanna (1640CSE00498)
3. Asrafia Malek choya (1640CSE00499)
4. Rafe Ara Eisha (1641CSE00539)

Git Repository link:

https://github.com/RaisaAkter/CIFAR-10-Project

Introduction:

The object recognition challenge which was based on CIFAR-10 dataset was our project purpose. The CIFAR-10 dataset contains of 60,000 images. Each image is 32*32 color image .There are 10 object classes & 6000 images for each class. In this project, CIFAR- 10 dataset was divided into 50,000 labeled training images & 10,000 with test image. To make it more competitive ,kaggle adds 2,90,000 junk images with 10000 test images .We further divided the training set into 40000 training samples & 10000 validation sample to select the best model & hyper parameter .The classifier trained on the 40000 images and then used to predict on the testing set and evaluate the prediction accuracy. We used CNN architecture with different hyper parameter and got the accuracy of about 84%.

Data Preprocessing:

To feed the data into the model, it is recommended to process the dataset before it sends to model because it is proved that the preprocessed data can give better performance than the raw data. So when we were loading the dataset ,we normalized the dataset by dividing by 255,and deducing the mean of the training set from both the training set , validation set and testing set .

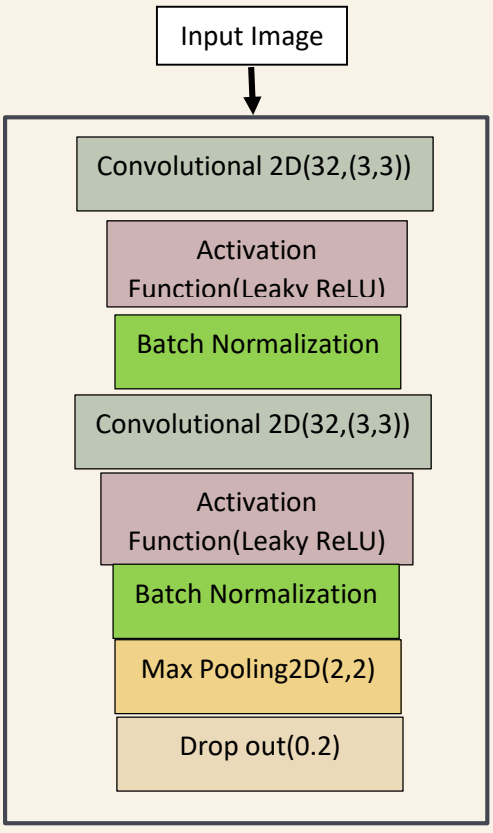
We also did some augmentation techniques to improve the performance. Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks. we used ImageDataGenerator from keras and used 4 parameter .

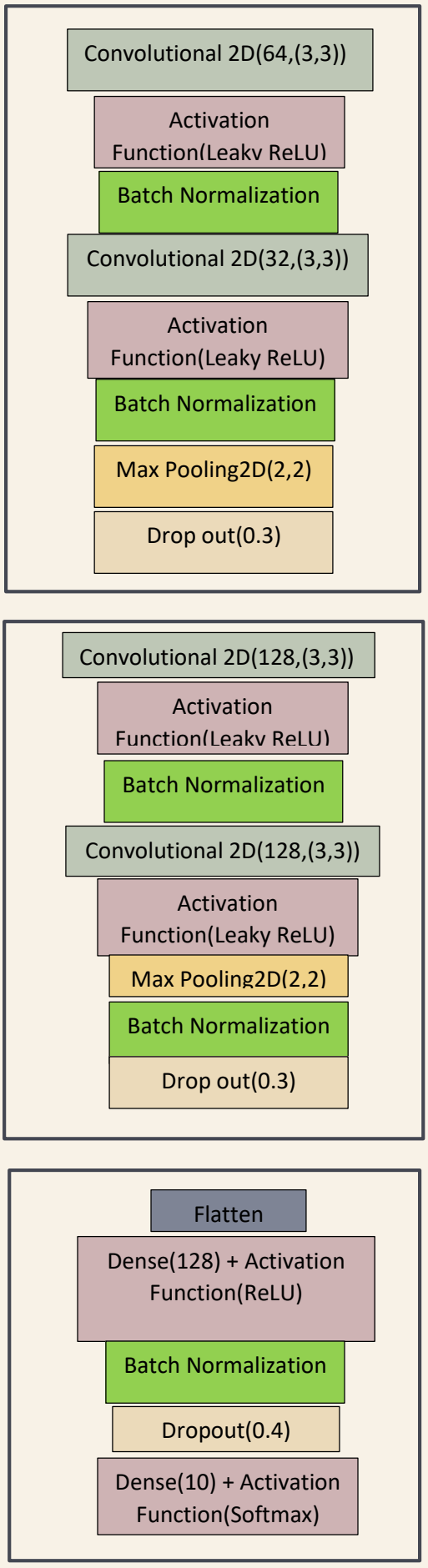
```
width_shift_range=0.1, height_shift_range=0.1, horizontal_flip=True,
                            rotation_range=15
```

Network Architecture:

CNN:

The general convolutional neural network (CNN) consists of multiple layers that transform the input image volume into a output volume holding the class scores .The several distinct type of layers are convolutional layer, activation layer, maxpool layer and fully – connected layer. The network architecture in this project is given below:




Convolutional 2D(64,(3,3))
Activation Function(Leaky ReLU)
Batch Normalization
Convolutional 2D(32,(3,3))
Activation Function(Leaky ReLU)
Batch Normalization
Max Pooling2D(2,2)
Drop out(0.3)
Convolutional 2D(128,(3,3))
Activation Function(Leaky ReLU)
Batch Normalization
Convolutional 2D(128,(3,3))
Activation Function(Leaky ReLU)
Max Pooling2D(2,2)
Batch Normalization
Drop out(0.3)
Flatten
Dense(128) + Activation Function(ReLU)
Batch Normalization
Dropout(0.4)
Dense(10) + Activation Function(Softmax)

In the first 3 layer we add 2 convolutional layer, 2 activation function layer, adds batch normalization after the activation layer, and lastly we added a MaxPooling layer and a dropout rate of different value.

In the last layer, we flattened our CNN layer, added a depth of 128 neuron with activation function. Then we added batch normalization and a dropout rate. Lastly we added output layer of dense 10, as in the dataset there was 10 class and an activation function 'softmax'.

Training Procedure:

To train the model, firstly we load the train dataset from the directory and the label. Then we split the dataset to train with one portion and to test with another portion. In this way we calculate loss and on basis of this we update the parameters.

We also applied the augmentation technique. We took 4 parameter and create augmented data from them. For training the model initially we the learning rate was 0.001. Then we used learning rate scheduler which decrease the learning rate while increasing the epochs. We used the `categorical_crossentropy` loss in our model and 'Stochastic Gradient Descent' as optimizer. Initially we set epoch=50 but we didn't get enough performance. The best performance we achieved with epoch=120. We assigned the batch size as 250. We also used the EarlyStopping method in our model and set the patience=40.

The whole training procedure was done on windows operating system, with 4gb RAM and 2.70 GHz CPU. The programming language was python and we also used different libraries for our work. We experimented with different activation function and got best performance with 'LeakyReLU' activation function.

Result:

Model Architecture	Accuracy
Two convolutional layer with epoch=1	0.10170
Same architecture with epoch=15	0.68410
New model with epoch=15	0.73190
Same model with epoch=100	0.78580
Adding data augmentation	0.79820
Added learning rate scheduler with same model	0.83900
Changed activation function, dropout rate with epoch=120	0.84390

Conclusion:

So in the project we tried to work from the first and improve the performance. We also tried to apply different techniques to increase the accuracy rate. We submitted our result in the kaggle 7 times and got 0.84390(highest). We expected better performance from our side, but we couldn't applied more algorithm or techniques because of time shortage. If we could give more time, study then the performance could be better than it is now.