ECEP 596, Autumn 2019 Homework 4: Final Project

Image Classification on CIFAR-10

Hiralben Mistry, Charu Sundaram

1. Introduction

The sole motivation of this project is to learn how to develop Convolutional Neural Network for Object detection - Image classification problem. Being new to Machine learning methodologies, we believe this project work would help us learn from scratch, the different steps involved in this course of action i.e. how to pre process a dataset, develop a model and lastly fine tune various parameters to achieve improved performance.

The main objective of this project is to develop a CNN which does the image classification on CIFAR 10 dataset for at least five classes with improved efficiency. Convolutional neural networks are deep learning algorithms that can train large datasets with millions of parameters, in the form of 2D images as input and convolve it with filters to produce the desired outputs. Various function optimization methods such as SGD, Adam along with various regularization techniques like Batch Normalization and Dropout are used to get good accuracy on image classification task. Also, we intended to fine tune a pre trained classifier VGG16 to adapt to our dataset and then compare the results with the CNN that is built from scratch. Keras which is a high-level neural networks API, written in Python and capable of running on top of TensorFlow is being used for this work. Tensorflow is run on Python3.6. All the coding, modeling and visualizations are done using Jupyter notebook.

2. Related work

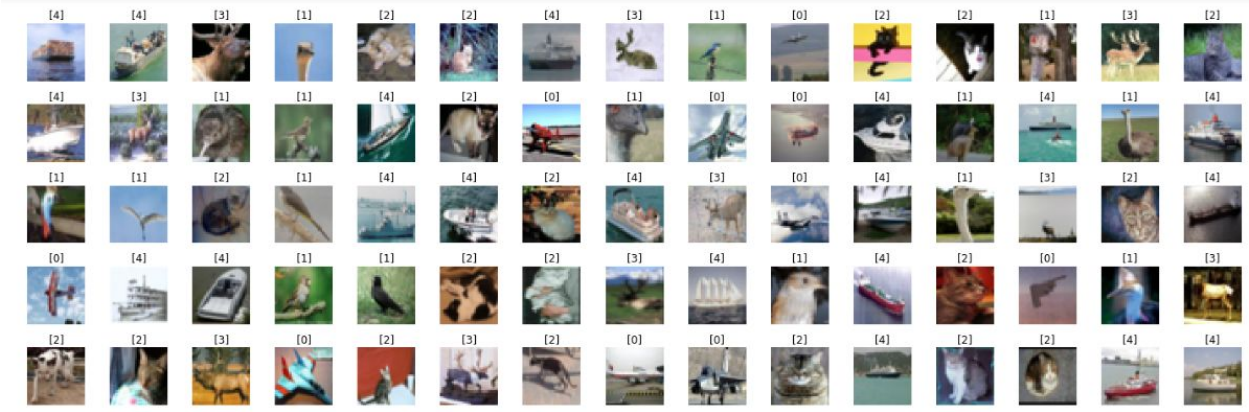
Deep learning has shown improvement in accuracy on various datasets. Some of the works have been described below:

[1] discusses about performance of a simple CNN on MINST and CIFAR-10 with less computation cost. **[2]** is an implementation of CNN with Adam optimization technique to update weights for best accuracy and uses dropout and regularization to reduce overfitting. **[3]** is implemented on MNIST and CIFAR-10 dataset and performance are evaluated based on the accuracy of the model.Real time data augmentation and dropout is used and experimented on CPU unit. **[4]** proposes a modified VGG-16 and discusses about how to take advantage of deep CNN on small datasets like CIFAR-10 with simple and proper modifications and don't need to re-design specific small networks.

We consolidated ideas from the above references and developed a simple CNN with few layers. We analyzed its performance based on the accuracy on the test data. We realized how different optimization and regularization techniques influence the results. Finally, we also developed a modified VGG-16 to adapt to our problem and compared the result from the previous CNN.

3. Dataset

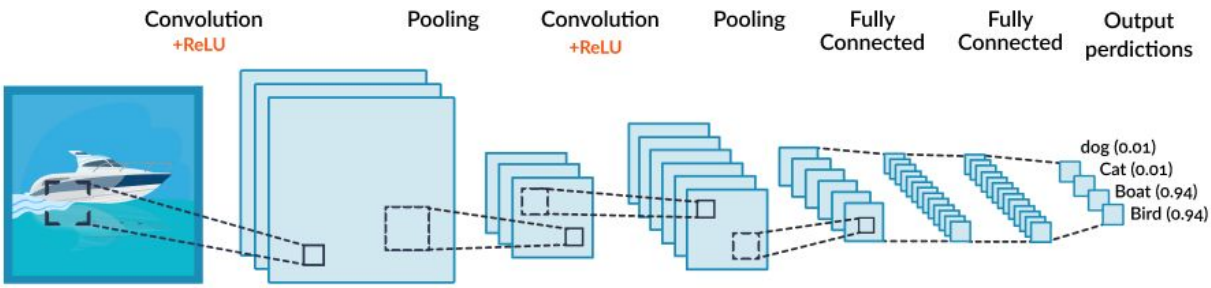
The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) with 6000 images per class. There are 50000 training images and 10000 test images.The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. For our implementation we extracted only five classes (airplane', 'bird', 'cat', 'deer', 'ship'), so ended up having 25,000 images for training data and 5,000 for test data. Also, we set aside 20% of training dataset as validation set. Finally, there were 20,000 images as training dataset, 5,000 as validation dataset and another 5,000 as test dataset. Below figure represents a sample dataset which only consist of images from the classes of our interest.



4. Architecture and Algorithms

CNN Basic Components:

Basic CNN architecture of convolutional layer, pooling layer and fully-connected layer.



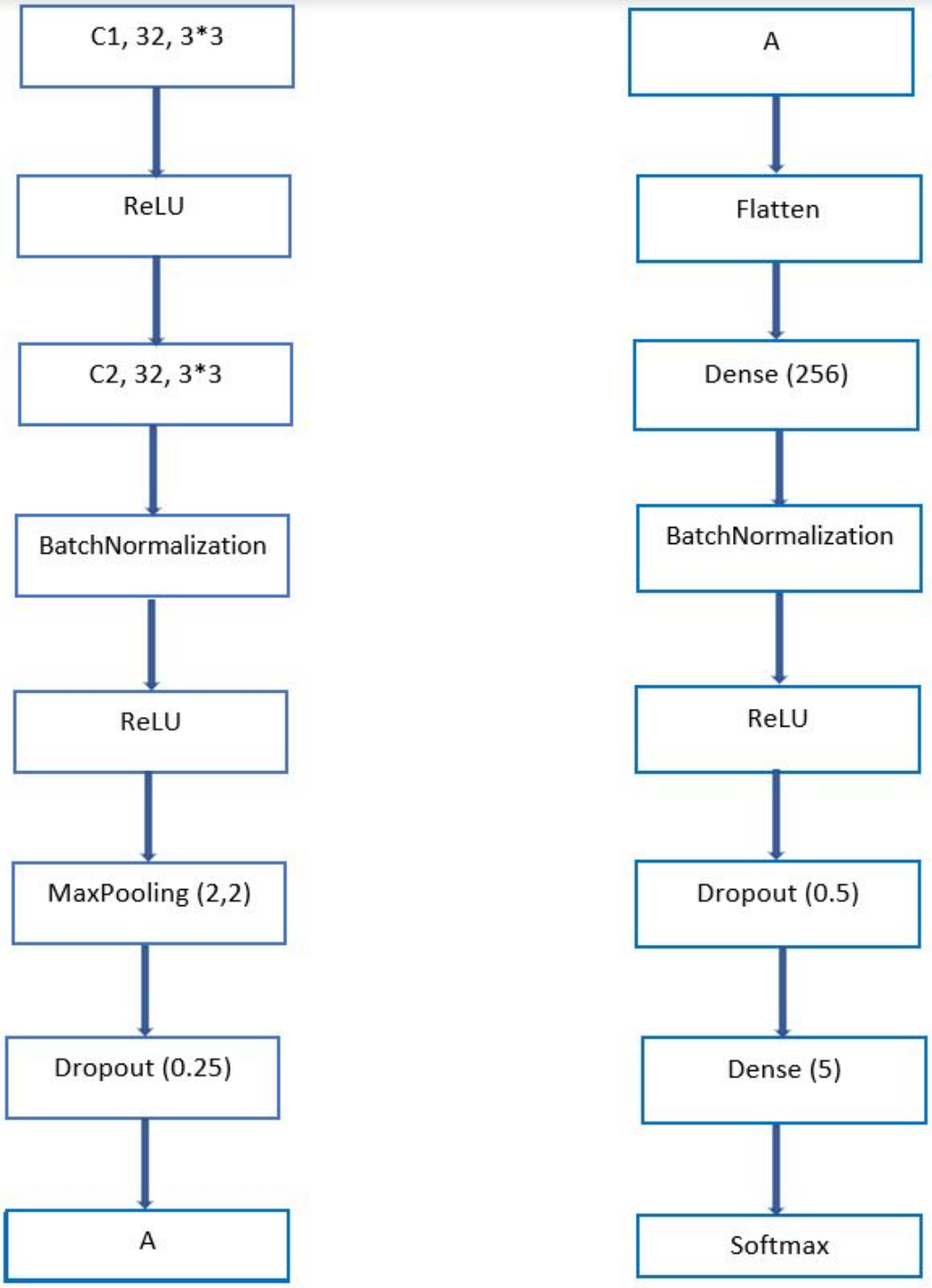
- A convolutional layer extracts features from a source image. A convolution kernel is passed over the image, inspecting a small window of pixels at a time, and a feature map is created with probabilities that each feature belongs to the required class.
- A pooling layer down samples each feature i.e. reduces the dimensionality of each feature while maintaining its most important information. Most CNN's uses "max pooling" in which the highest value is taken from each pixel rea scanned by CNN.
- A fully-connected layer flattens the features identified in the previous layers into a vector and predicts probabilities that image belongs to each one of several possible labels.

Algorithm:

This section details about the model developed from scratch which gave the best accuracy among our experiments excluding the transfer learning model experiment.

- Batch size = 32 , Number of epochs = 20
- Optimizer = Adam, Learning rate = 0.0006
- Data Augmentation = True with rotation, width shift, height shift and horizontal flip.
- Dropout = True
- Batch normalization = True

Conv1 → Activation → Conv2 → BatchNormalization → Activation → MaxPooling → Dropout → Flatten → Dense → BatchNormalization → Activation → Dropout → Dense → SoftMax → Result



The above figure shows the architecture of the model. The 32x32 input image is given to the model where the first convolutional layer learns 32 features through a 3x3 filtre. After the activation, another convolutional layer is stacked up that learns 32 features with a 3x3 filter. Then BatchNormalization is applied to speed up the process. ReLU activation is added after that and forwarded to max pooling layer. Then a dropout of 0.25 is implemented. The output is flattened and we have a fully

connected layer followed by BatchNormalization and activation. Then a dropout of 0.5 is carried out and a dense layer to number of output classes i.e. 5 and passed to softmax layer for the final output. The accuracy of the model is 81.74% with data augmentation included on a CPU unit on test dataset.

4. Experiments, Performance Evaluation and results

Below are the five steps in the neural network model life-cycle that we used in all our experiments.

1. Define Network.
2. Compile Network.
3. Fit Network.
4. Evaluate Network.
5. Make Predictions.

The model is made to fit on the training and validation set and the comparison of loss and accuracy over each epoch is analyzed. Finally the model is evaluated on the test dataset and the accuracy percentage on the predictions is observed. The confusion matrix is used to analyze how many right or wrong predictions resulted out of the model with respect to each image categories.

Experiment # 1 Base Model

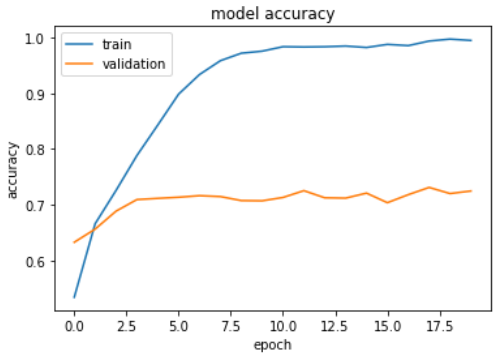
Parameters: Epochs = 20, Layers = 2, Batch size = 32, Activation = ReLU, Optimizer = SGD, Learning rate = 0.01

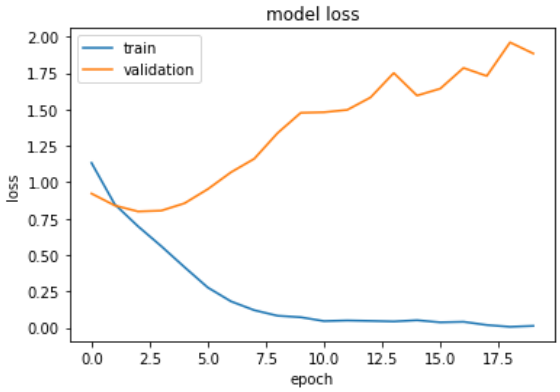
Model:

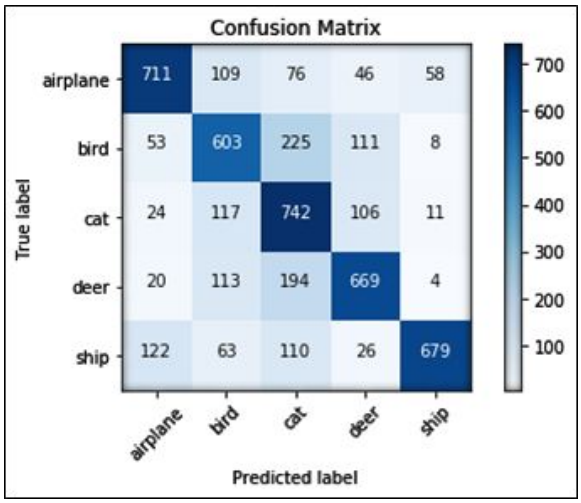
This base model consists of two convolutional layers. Because the number of layers is low, the filter size is also set to a low number 32. The resolution of CIFAR-10 dataset is only 32x32 and hence a small kernel size 3x3 is used. A non-linear activation function Rectified Linear Unit is used which has the advantage of having a constant gradient during back propagation. A logarithmic loss function is used with the stochastic gradient descent (SGD) optimization algorithm configured with a large momentum and weight decay start with a learning rate of 0.01.

- Convolutional input layer, 32 feature maps with a size of 3x3, ReLU
- Convolutional input layer, 32 feature maps with a size of 3x3, ReLU
- Max Pool layer with size 2x2
- Flatten layer
- Fully connected layer with 256 units and ReLU activation.
- Fully connected output layer with 5 units and softmax activation.

Performance:







Result :

By studying the loss and accuracy curves, we can conclude that the accuracy of the training set is high but lagging on the validation set. Hence, this model is a overfitting one. The accuracy of the model is **71.54%.**

Experiment # 2 Base Model with Regularization

Parameters: Epochs = 20, Layers = 2, Batch size = 32, Activation = ReLU, Optimizer = SGD, Learning rate = 0.01

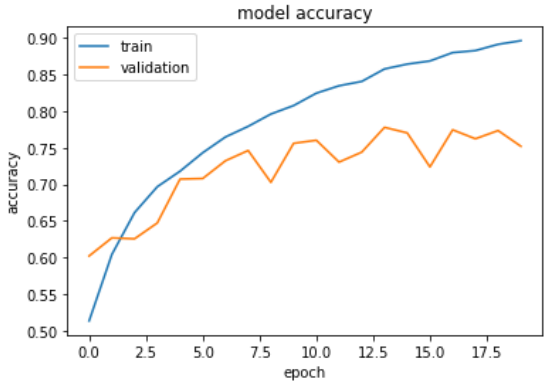
Model:

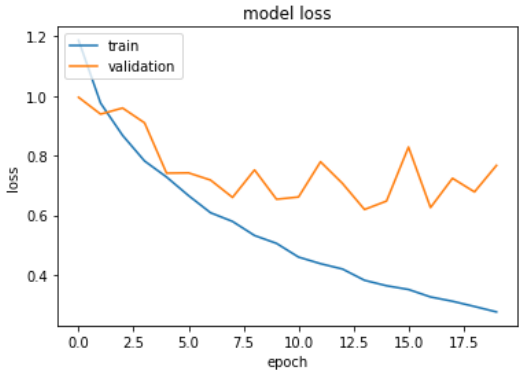
In this model, regularization techniques like Batch normalization and Dropout are included. Distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and makes it notoriously hard to train models with saturating nonlinearities. To increase

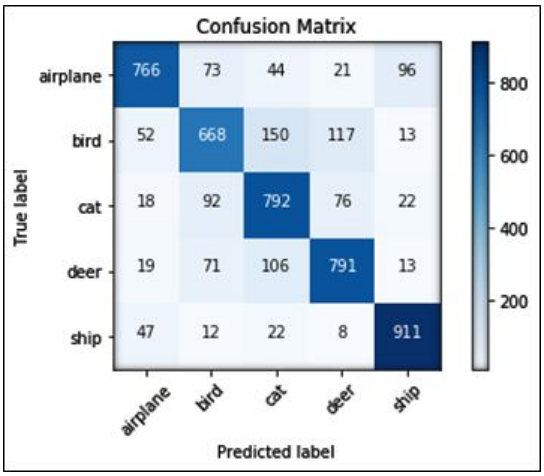
the stability of a neural network and to speed up, batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. When a large feedforward neural network is trained on a small training set, it typically performs poorly on test data. This "overfitting" is greatly reduced by randomly omitting half of the feature detectors on each training case. This is achieved by using Dropout layer.

- Convolutional input layer, 32 feature maps with a size of 3x3, ReLU
- Convolutional input layer, 32 feature maps with a size of 3x3, ReLU
- BatchNormalization
- ReLU activation
- Max Pool layer with size 2x2
- Dropout of 25%
- Flatten layer
- Fully connected layer with 256 units and ReLU activation.
- BatchNormalization
- ReL activation
- Dropout of 50%
- Fully connected output layer with 5 units and softmax activation.

Performance:







Result :

The accuracy and loss curves of the training and validation set are getting closer. By adding batch normalization and dropout layers, overfitting is substantially reduced and the accuracy is also increased. The accuracy of the model is **73.60%.**

Experiment # 3 Model with increased convolution layers, dropout and Adam optimizer

Parameters: Epochs = 20, Layers = 4, Batch size = 32, Activation = ReLU, Optimizer = Adam, Learning rate = 0.001

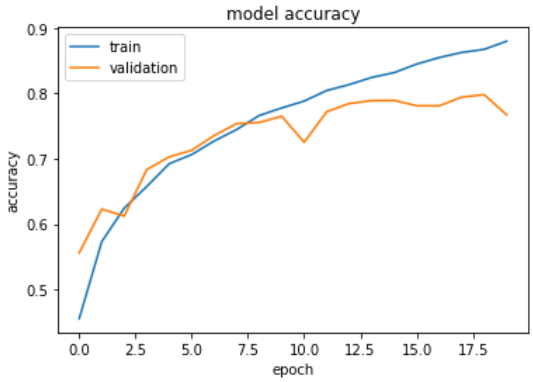
Model:

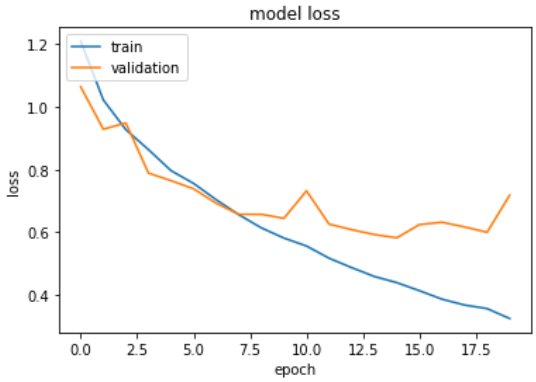
The change implemented in this model is having more convolutional layers i.e. 4 with filter size of 64 and 128. Also, Adam optimizer with learning rate of 0.001 is used. Stochastic gradient descent maintains a single learning rate for all weight updates and the learning rate does not change during training whereas Adam combines the advantages of two extensions of stochastic gradient descent which are Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients and Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing).

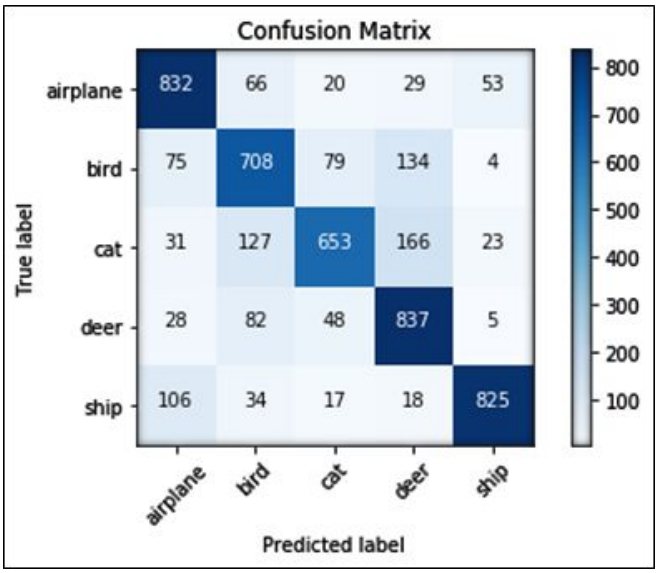
- Convolutional input layer, 64 feature maps with a size of 3x3, ReLU
- Convolutional input layer, 64 feature maps with a size of 3x3, ReLU
- Max Pool layer with size 2x2
- Dropout of 40%
- Convolutional input layer, 128 feature maps with a size of 3x3, ReLU
- Convolutional input layer, 128 feature maps with a size of 3x3, ReLU
- Max Pool layer with size 2x2

- Dropout of 40%
- Flatten layer
- Fully connected layer with 1024 units and ReLU activation.
- Fully connected layer with 1024 units and ReLU activation
- Fully connected output layer with 5 units and softmax activation.

Performance:







Result :

The accuracy and loss curves of the training and validation set are getting closer and smooth. By experimenting with adam optimizer, the accuracy is also increased. The accuracy of the model is **76.40%.**

Experiment # 4 Base Model with Data Augmentation, Regularization and Adam optimizer

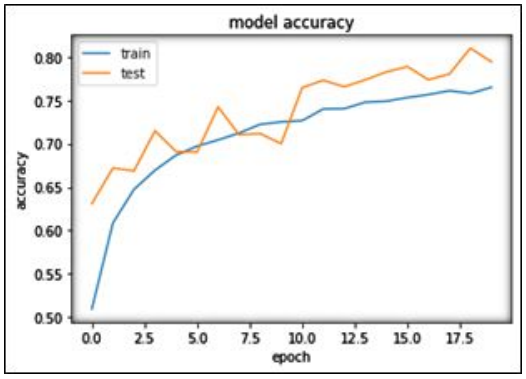
Parameters: Epochs = 20, Layers = 2, Batch size = 32, Activation = ReLU, Optimizer = Adam, Learning rate = 0.0006

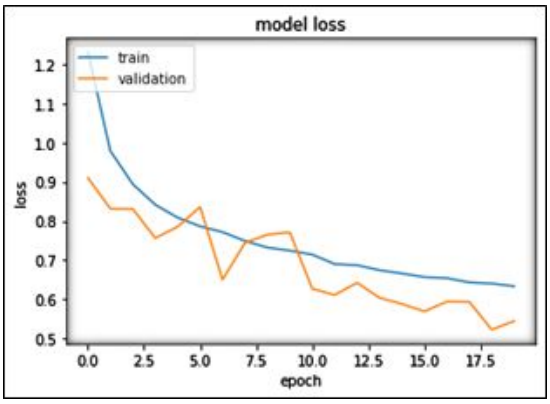
Model:

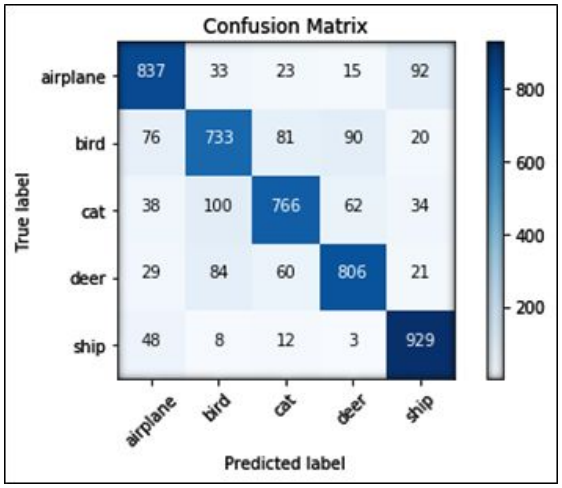
In this experiment, data augmentation is implemented using ImageDataGenerator class in keras which defines image data preparation and augmentation. This class generates batches of tensor image data with real-time data augmentation. The data will be looped over in batches. In this implementation, random rotation of 10 deg, width shift of 0.1, height shift of 0.1 and horizontal flip are used to generate augmented data. This increases the size of the training dataset without need of real data. Also, the learning rate of Adam optimizer is set to 0.0006.

- Convolutional input layer, 32 feature maps with a size of 3x3, ReLU
- Convolutional input layer, 32 feature maps with a size of 3x3, ReLU
- BatchNormalization
- ReLU activation
- Max Pool layer with size 2x2
- Dropout of 25%
- Flatten layer
- Fully connected layer with 256 units and ReLU activation.
- BatchNormalization
- ReL activation
- Dropout of 50%
- Fully connected output layer with 5 units and softmax activation.

Performance:







Result :

The usage of data augmentation and Adam with a learning rate of 0.0006 increased the performance. The accuracy of the model is **81.50%.**

Experiment # 5 Transfer learning from VGG16

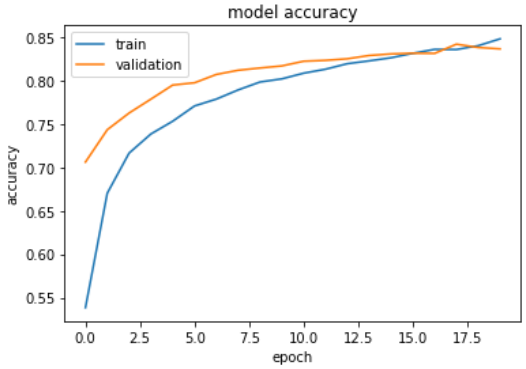
Model:

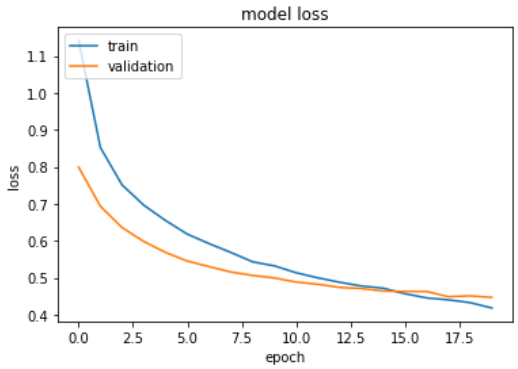
In this experiment, the well known model VGG16 which has been trained on Imagenet dataset is taken and fine tuning is done to suit our problem statement. Very deep models like VGG16 can also be used to fit small dataset like CIFAR10 as long as the input image is big enough so that it won't vanish as the model go deeper.The transfer learning from VGG16 along with some fine tuning improved the performance. The first step of transfer learning would be not including the last layer of the model that makes predictions about 1000 classes in Imagenet. To maintain the weights of the layer from VGG16, keras has a parameter called “trainable” in each layer. If this parameter is set to false then the weights remain unchanged. We go over each layer and select which layers we want to train. In this implementation only first three blocks of VGG16 is taken to reduce the training time as we were running on a CPU unit. Then a Global average pooling, BatchNormalization, two fully connected layers of 256 units, dropout of 60% and a fully connected output layer with 5 units and softmax activation is included. Since we are using VGG16 for extracting features, the minimum compatible input image size is 48x48 but CIFAR10 image resolution is 32x32, so we had to resize the whole dataset. Also, Adam optimizer of learning rate 0.0001 is used.

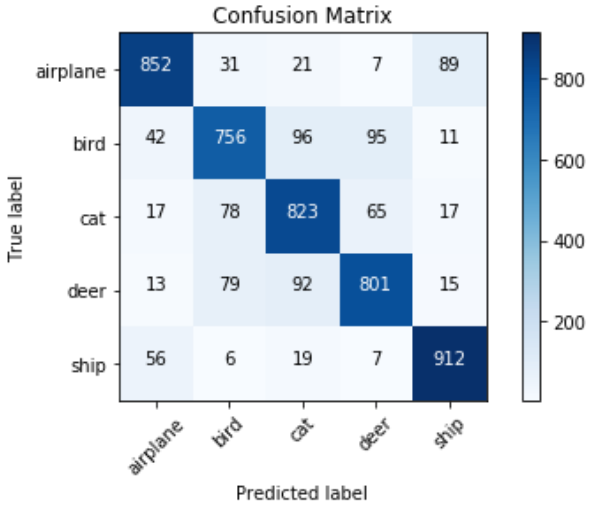
- Convolutional input layer, 64 feature maps
- Convolutional input layer, 64 feature maps
- Max Pool layer with size 2x2
- Convolutional input layer, 128 feature maps

- Convolutional input layer, 128 feature maps
- Max Pool layer with size 2x2
- Convolutional input layer, 256 feature maps
- Convolutional input layer, 256 feature maps
- Convolutional input layer, 256 feature maps
- Max Pool layer with size 2x2
- BatchNormalization
- Fully connected layer with 256 units and ReLU activation
- Fully connected layer with 256 units and ReLU activation
- Dropout of 60%
- Fully connected layer with 5 unis and Softmax activation

Performance:







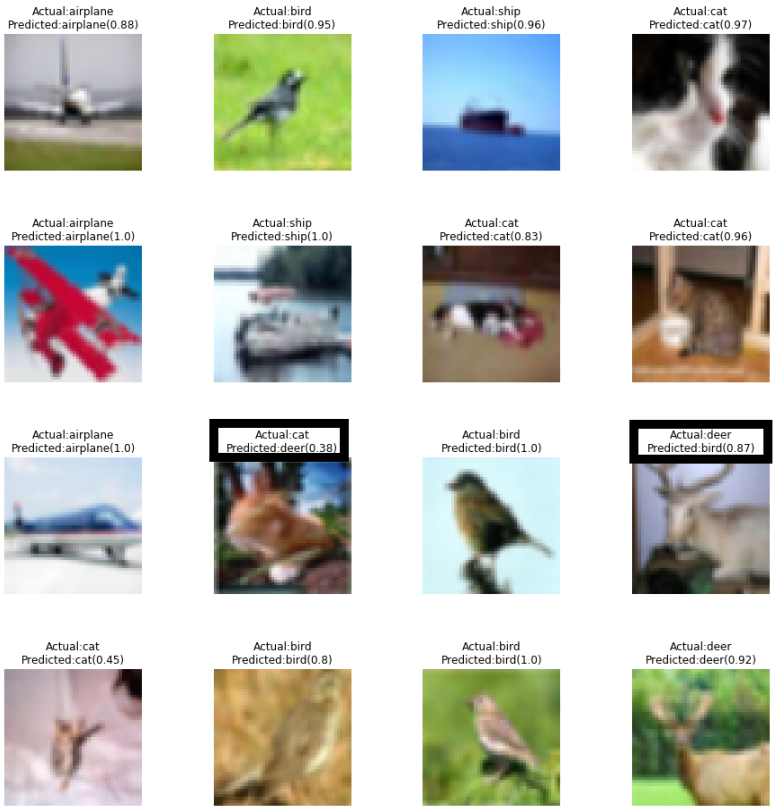
Result :

The loss and accuracy curves of validation and train set are almost similar and smooth when compared to previous experiments. The accuracy of the model is **82.82%.**

Comparison table:

Experiment	Model	Accuracy
1	Base Model (2 conv layer with 32 filters)	71.54%
2	Base Model with regularization	73.60%
3	Model with 4 layers and Adam optimizer	76.40%
4	Base Model with DataAug, Regularization and Adam	81.50%
5	Transfer learning from VGG16 and fine tune	82.82%

Below is the sample prediction plot obtained from transfer learning from VGG16 model. The false predictions are highlighted for easy observation.



5. Future work

Few possible future extensions of the work that is presented here would be

- Implement image classification on all the10 classes of objects in CIFAR10 unlike our work is only on 5 categories.
- Do an experiment with more convolutional layers with data augmentation and regularization techniques whereas our implementation only involves two convolutional layers with data augmentation.
- Instead of extracting only three blocks from VGG16 for implementing transfer learning, use all the five blocks and observe the accuracy of the model.
- Create a dataset of your own that is useful for your personal application and evaluate the best model in hand on that dataset for accuracy.
- Use dilated convolution and evaluate the performance of the model.
- Experiment with different learning rates as we have tried very minimal attempts.

6. Summary and conclusion

In this work we tried to build a CNN from scratch for image classification and fine tuned the hyper parameters for improved accuracy. We analyzed the performance of each model with loss and accuracy curves on training and validation set and finally evaluated the model depending on the accuracy on test data. Different regularization and optimization techniques with varying learning rates have been implemented. Also, we did transfer learning from VGG16 and tuned it to adapt to our test case. We realized shallow models are sufficient for dataset with less image size like CIFAR10. The shallow model with data augmentation, regularization and Adam optimizer showed almost similar accuracy to the value obtained from transfer learning.

References

[1] T. Guo, J. Dong, H. Li and Y. Gao, "Simple convolutional neural network on image classification," *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*, Beijing, 2017.

[2] R. Doon, T. Kumar Rawat and S. Gautam, "Cifar-10 Classification using Deep Convolutional Neural Network," *2018 IEEE Punecon*, Pune, India, 2018.

[3] R. Chauhan, K. K. Ghanshala and R. C. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, Jalandhar, India, 2018

[4] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Kuala Lumpur, 2015.