Pattern Recognition and Machine Learning

Course Project Report

Project Title:

Natural Image Processing on CIFAR-10 dataset

Team Members:

Kartik Narayan
Kotha Pranith Reddy
Manav Gopal

Table of Contents

About Dataset

The learning task in this exercise is to create an image classifier using Convolutional Neural Network (CNN) architectures for the CIFAR-10 Dataset of images [1,2,6]. The data consists of 60,000 images of objects from 10 distinct categories (airplane, car, bird, cat, deer, dog, frog, horse, ship, truck). The images are 32x32 pixels with three RGB color channels [6]. You should create an image classifier by developing a Convolutional Neural Network (CNN) and training it using the packages and approaches we have been discussing in lecture. See the course website for example codes. For the test dataset, your output should be a prediction of the class of each image. This will be evaluated by the percentage of predictions that are in the correct category.

You should experiment with different choices of architectures, hyper-parameters, and training protocols to get familiar with how convolutional neural networks work and how they can be tuned to improve image classification. You are welcome to experiment with other CNN architectures, with stacking additional convolutional layers, by adjusting the width, kernel size, and depth of filters, or with performing data augmentation or by making other modifications. For inspiration, see the papers below for discussions of both historic and modern techniques [1-5].

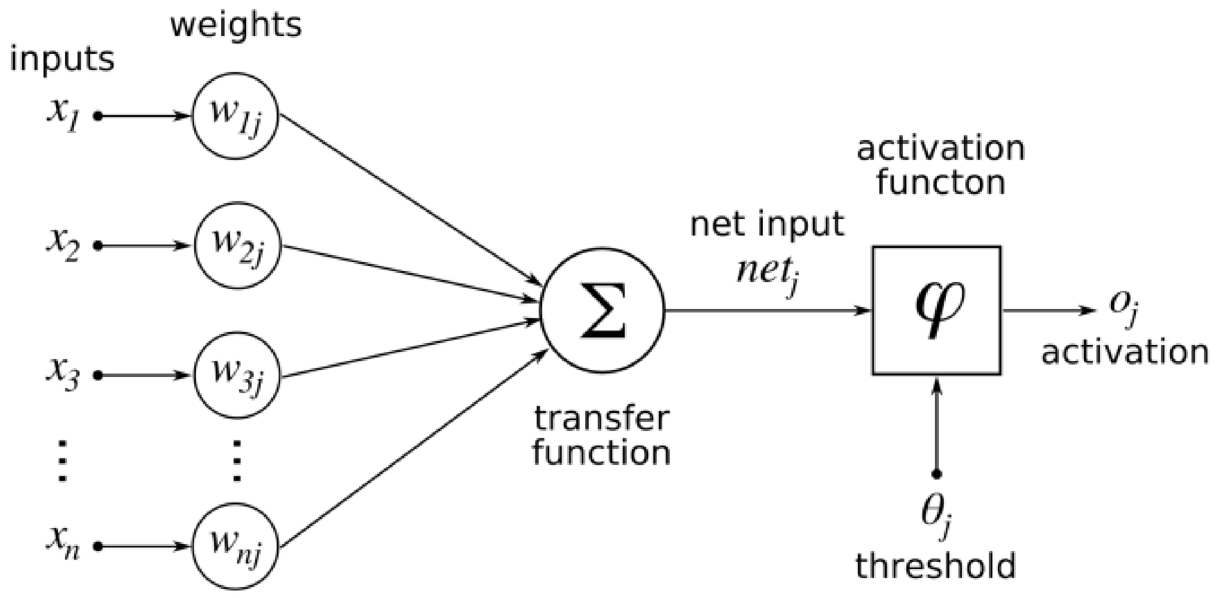
Classifier Used and Description

Convolutional Neural Network (CNN)

Convolutional neural networks are named so because they are said to be composed of multiple layers of artificial neurons. Artificial neurons are just mathematical functions that calculate the weighted sum of multiple inputs and give output as an activation value. A convolutional neural network algorithm takes an input image and assigns importance in the form of weights or biases to the objects in the image. This helps in differentiating one from the other. Weights and bias are both learnable parameters. The neural network can randomize or adjust both the values in order to get corrected output.[8]

Bias makes up the difference between the function's output and its intended output. A low bias value hints that the network is making a large number of assumptions about the output whereas a high bias indicates that assumptions made are less. Weight is a parameter of a neural network that transforms the input data within the hidden layers of the network. Or we can say that weights are multiplicative factors of the filters. CNN is a series of nodes and within each node there exists a set of inputs, weights and bias values. As the input is entered, it gets multiplied by weight and the current output is either the final output or is passed to the next convolutional layer of the network. A low weight will have no significant effect on the input but a larger weight will have a larger impact on the output. When input is an image, the convolutional layers generate several activation mappings. The artificial neurons taes in a patch of pixels as its input and their color with weights. Then the values are summed up and run through the activation function.[9]

On a general note, the initial layer of the CNN does the job of detecting the most basic features such as edges and the output of this convolutional layer acts as input for the next layer. The next layer extracts some more complex features. This way layer by layer the complexity of features increases to objects, faces and much more.



Training the convolutional neural network

Adjusting weights is one of the most significant processes while training in order to extract right features from the image. This adjusting of weights is termed as training in ConvNets. Initially, CNN choses randomized weights. In the training process, with the help of large datasets of images, the convNet processes the images with its random weight values and compares the output with the correct image label. Adjustments in weight values are made if the output and the intended output do not match.[7,8]

The corrections are made using a technique called Backpropagation which does optimization of the tuning process and helps the network to decide which units need to be adjusted.

The entire training set is run multiple times and each of this run is termed as epoch. So technically quoting, after every epoch the weights are adjusted to make output better than before. After the completion of training, another significant task is testing a dataset to verify the accuracy.[8]

Testing the dataset

The test dataset is different from the training dataset. Each image of the test dataset is run through the ConvNet and the output and intended output is compared to the actual image labels. There is a chance of overfitting when there is not enough variety in the training set or if the number of epochs is quite large.[8]

Limitations

- One problem with neural networks is that they are unable to understand the relations between different objects.
- Neural networks begin breaking as soon as they have to move a bit out of the context.[8]

Importing Dataset

We had imported this cifar-10 dataset from the tensorflow.keras.dataset and then loaded the data into X_train, y_train, X_test and y_test.

Then printed the shape of those training and testing dataset which comes out to be

```
(50000, 32, 32, 3)
(50000, 1)
(10000, 32, 32, 3)
(10000, 1)
```

Preprocessing

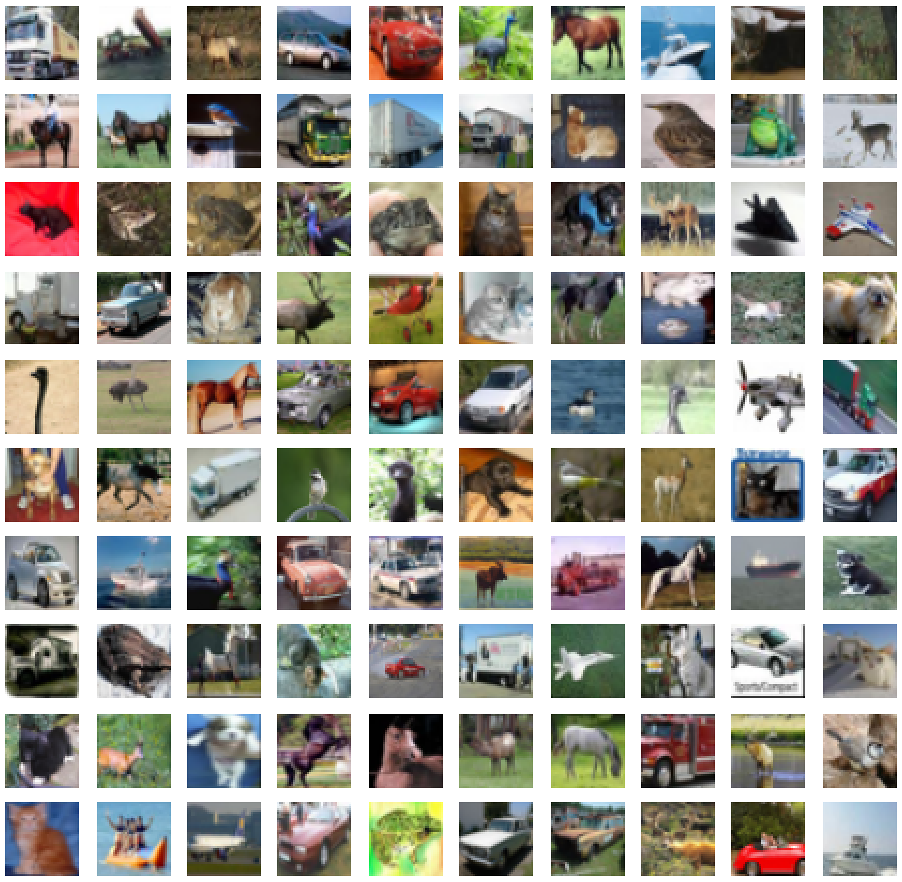
- Image Normalisation is a typical process in image processing that changes the range of pixel intensity values. Its normal purpose is to convert an input image into a range of pixel values that are more familiar or normal to the senses in order to reduce data redundancy and improve data integrity. In simpler terms, normalisation makes sure that all of your data looks and reads the same way across all records

  As the image values range from 0 to 255, it is good practice to normalise the pixel values so that each pixel value has a value between 0 and 1. Hence we had divided all the training and testing dataset values by 255 to have them in range 0 to 1.

- Shape of the training and testing data:

  ```
  Number of train images:  50000
  Number of test images:  10000
  ```

- Here we had printed some random images by taking values from X_train and then plotting using imshow.



Building CNN model

- Take the values of y_train in a label array and convert it into a dataframe because we don't want to alter the y_train data as we may need it in further calculation so it's always better to make a new copy whenever we want to do some changes in the basic dataset.

	labels
0	6
1	9
2	9
3	4
4	1
...	...
49995	2
49996	6
49997	9
49998	1
49999	1

- We had made a classes array which contains all the unique classes of the y_train dataset.

  ```
  classes: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  ```

- We had created a label dictionary inorder to label the classes:

  ```
  labels_dict =  {0:'airplane', 1:'automobile', 2:'bird', 3:'cat',
  4:'deer', 5:'dog', 6:'frog', 7:'horse', 8:'ship', 9:'truck'}
  ```

- We used to_categorical to transform our training data before we passed it to our model as our training data uses classes as numbers, to_categorical will transform those numbers into proper vectors for using with models. We can't simply train a classification model without that so we converted the labels column of the labels dataframe using the to_categorical function.
- Whenever we want to evaluate any model it is necessary to split the data available into training and testing sets. Before splitting the data into train and test we should make sure that the set is large enough to yield statistically meaningful results and is representative of the data set as a whole. In other words, don't pick a test set with different characteristics than the training set.

  We had splitted the training dataset (X_train) into training and testing dataset using the train_test_split function.

- The shape of the splitted data is :

  shape of training data : (47500, 32, 32, 3)

  shape of validation data : (2500, 32, 32, 3)

Data Augmentation

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.

Data augmentation is used for improving deep learning robustness. For example, it can be applied on images for classification solutions. It is often necessary when confronted with a massive domain of use like an open world as it improves the robustness of a neural network.

Different data augmentation can lead to very different stability performance for the neural network. The more the neural network can handle noise in its dataset the better the data augmentation had a positive impact during the training phase.

Model

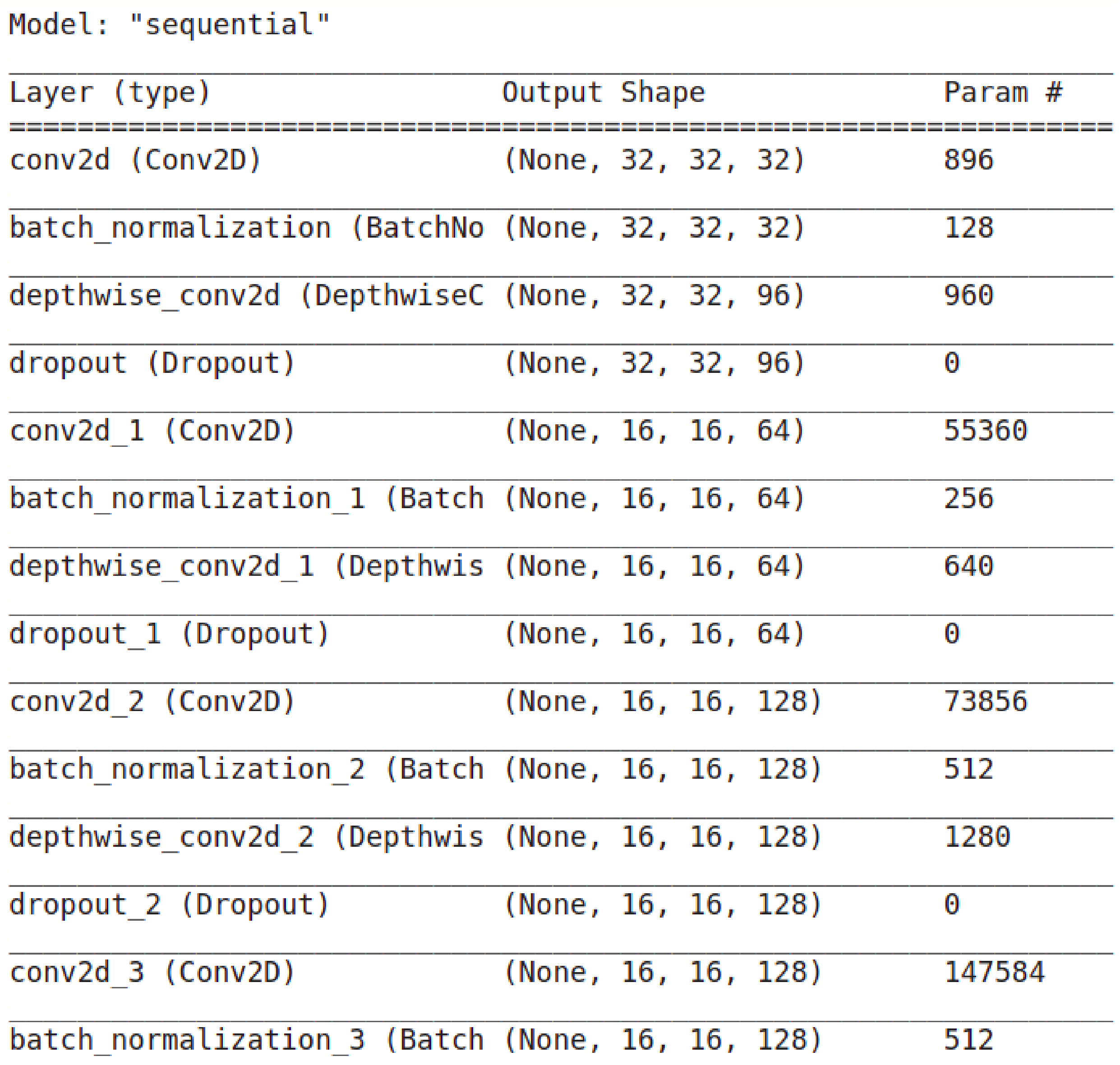
We tried different classifier models for this project, three of which we decided to keep which are Random Forest, Multi layered perceptron and Convolutional Neural network.

Building CNN sequential model

The sequential model is a linear stack of layers. The model needs to know what input shape it should expect. For this reason, the first layer in a sequential model (and only the first, because following layers can do automatic shape inference) needs to receive information about its input shape. We can pass arguments like batch_size and input shape in the first layer. We can add layers to the model and can tune parameters like units and activation_function to explore the possible accuracies. We also have an option to choose the output activation function.

In our code we have used one input layer, one output layer and four hidden layers. The input shape is passed as (32,32,3) and the activation function for the hidden layer is set as relu. The output activation function used in our sequential CNN model is softmax activation function. We are also normalising the data after the output of each hidden layer.

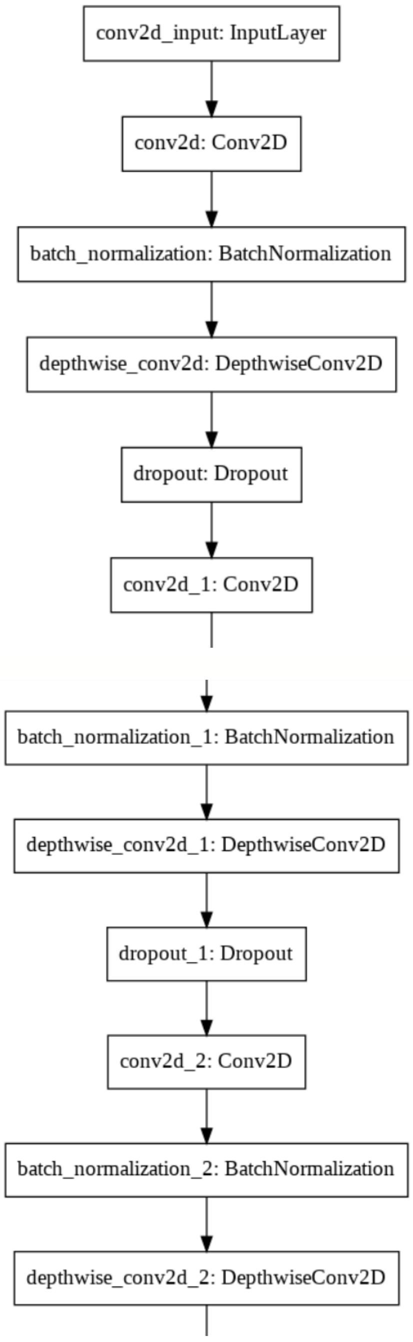
Sequential Model summary



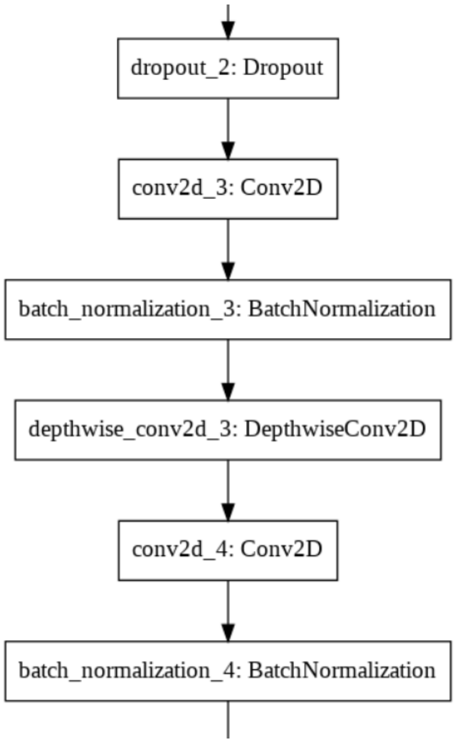
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896       
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 32)        128       
_________________________________________________________________
depthwise_conv2d (DepthwiseC (None, 32, 32, 96)        960       
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 96)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 64)        55360     
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 64)        256       
_________________________________________________________________
depthwise_conv2d_1 (Depthwis (None, 16, 16, 64)        640       
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 64)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 128)       73856     
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 128)       512       
_________________________________________________________________
depthwise_conv2d_2 (Depthwis (None, 16, 16, 128)       1280      
_________________________________________________________________
dropout_2 (Dropout)          (None, 16, 16, 128)       0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 128)       147584    
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 128)       512       
_________________________________________________________________
depthwise_conv2d_3 (Depthwis (None, 16, 16, 128)       256       
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 256)         295168    
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 256)         1024      
_________________________________________________________________
depthwise_conv2d_4 (Depthwis (None, 8, 8, 256)         2560      
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 512)         131584    
_________________________________________________________________
batch_normalization_5 (Batch (None, 4, 4, 512)         2048      
_________________________________________________________________
depthwise_conv2d_5 (Depthwis (None, 4, 4, 512)         1024      
_________________________________________________________________
dropout_3 (Dropout)          (None, 4, 4, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 8192)              0         
_________________________________________________________________
dense (Dense)                (None, 2048)              16779264  
_________________________________________________________________
dropout_5 (Dropout)          (None, 2048)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1049088   
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130      
=================================================================
Total params: 18,549,130
Trainable params: 18,546,890
Non-trainable params: 2,240
_________________________________________________________________
```

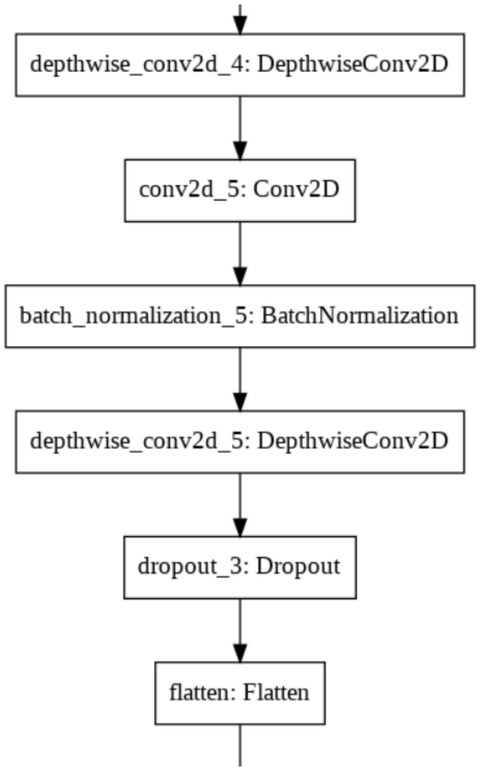
CNN Model Compilation

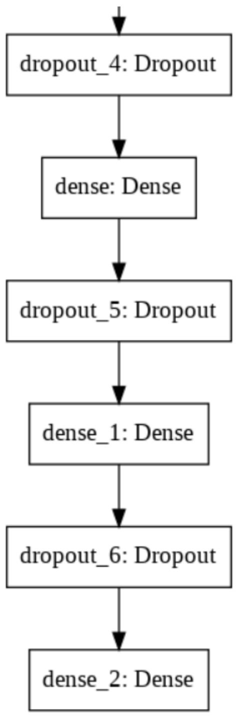
- Compiled the model using optimiser as ‘adam’ and loss as categorical_crossentropy.
- Plotted the model:



conv2d_input: InputLayer
conv2d: Conv2D
batch_normalization: BatchNormalization
depthwise_conv2d: DepthwiseConv2D
dropout: Dropout
conv2d_1: Conv2D
batch_normalization_1: BatchNormalization
depthwise_conv2d_1: DepthwiseConv2D
dropout_1: Dropout
conv2d_2: Conv2D
batch_normalization_2: BatchNormalization
depthwise_conv2d_2: DepthwiseConv2D


dropout_2: Dropout
conv2d_3: Conv2D
batch_normalization_3: BatchNormalization
depthwise_conv2d_3: DepthwiseConv2D
conv2d_4: Conv2D
batch_normalization_4: BatchNormalization


depthwise_conv2d_4: DepthwiseConv2D
conv2d_5: Conv2D
batch_normalization_5: BatchNormalization
depthwise_conv2d_5: DepthwiseConv2D
dropout_3: Dropout
flatten: Flatten



Fitting model

Taken the number of epochs as 10, a batch size as 64 and fitted the model as follows:

model.fit(datagen.flow(X_train, y_train, batch_size=batch_size), validation_data=(X_val, y_val), callbacks=[mcp_save], epochs=num_of_epochs)

Then evaluate the model using model.evaluate over the validation set and then use the accuracy_score to print the accuracy whose result is in the conclusion part.

Building Random Forest Model

- Done all the preprocessing of the data and then used the grid search to fit the best parameters among the parameters = {'n_estimators':[10,50,100]}
- Best parameter result of grid search {'n_estimators': 100}.
- Used random forest classifier with no of estimators as 100, then done a 5 fold cross val score on the scoring basis as accuracy and then fitted the model with X_train and y_train dataset
- Predicted the model over the testing dataset and printed the cross validation score and classification report whose values are given in the conclusion part.

Building MLP model

- Done all the preprocessing of the data and used the grid search to fit the best parameters among the `parameter_space = {'hidden_layer_sizes': [(50,50,50), (100,)],'activation': ['tanh', 'relu'],'alpha': [0.0001, 0.05]}` and used the cv=2.
- Took a Mlp classier with max_iter=300 and random state=42 and then fitted the training dataset.
- Used a five fold cross validation over the classifier with scoring base as accuracy.
- Then fitted the model and then Predicted it over the testing set and printed the cross validation score and classification report whose values are given in the conclusion part.

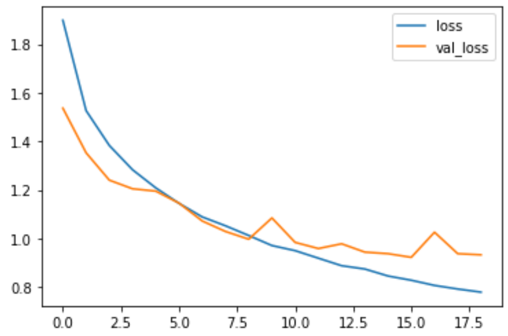
Result / Conclusion

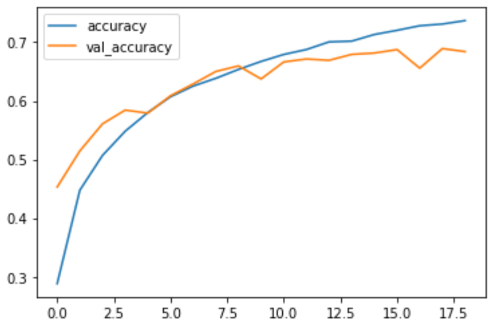
CNN Sequential Model Without Data Augmentation

- The loss and the accuracy without data augmentation for the CNN model came out to be:

```
['loss', 'accuracy']
[0.9331421852111816, 0.683899998664856]
```

- The plot for loss versus validation loss and the accuracy vs validation accuracy is as follows:





- Classification Report:

```
               precision    recall  f1-score   support
```

```
           0       0.73      0.70      0.72      1000
           1       0.81      0.83      0.82      1000
           2       0.59      0.57      0.58      1000
           3       0.46      0.55      0.50      1000
           4       0.57      0.70      0.63      1000
           5       0.71      0.44      0.54      1000
           6       0.68      0.83      0.75      1000
           7       0.80      0.70      0.75      1000
           8       0.77      0.81      0.79      1000
           9       0.84      0.71      0.77      1000

    accuracy                           0.68     10000
   macro avg       0.69      0.68      0.68     10000
weighted avg       0.69      0.68      0.68     10000
```

- Confusion Matrix:

```
array([[700,  21,  74,  26,  30,   2,  13,  14, 100,  20],
       [ 23, 830,  12,   8,   2,   3,  27,   4,  40,  51],
       [ 57,   9, 569,  79, 118,  49,  80,  21,  10,   8],
       [ 14,  13,  76, 549, 113,  72, 106,  33,  13,  11],
       [ 21,   2,  61,  70, 700,   5,  76,  48,  15,   2],
       [ 12,   7,  71, 295,  80, 438,  44,  44,   5,   4],
       [  5,   0,  46,  61,  43,   8, 827,   4,   4,   2],
       [ 18,   4,  32,  60, 111,  36,  19, 705,   1,  14],
       [ 66,  29,  17,  30,  12,   3,  14,   0, 810,  19],
       [ 41, 111,  11,  26,  11,   5,  13,  13,  58, 711]])
```

- Taken any random image and predicted that by using this classifier:

```
True class: horse
Predicted class: horse
```

CNN Sequential Model With Data augmentation

- The loss and accuracy of the sequential CNN model comes out to be as follows with data augmentation:

```
['loss', 'accuracy']
[0.5847573280334473, 0.803600013256073]
```

- The accuracy score comes out to be 0.8209 with data augmentation.

Random Forest Model

- The mean score of 5 fold cross validation comes out to be: 45.746.
- The classification report of random forest classifier:

```
              precision    recall  f1-score   support

           0       0.54      0.56      0.55      1000
           1       0.52      0.54      0.53      1000
           2       0.37      0.33      0.35      1000
           3       0.34      0.29      0.31      1000
           4       0.40      0.39      0.39      1000
           5       0.42      0.39      0.40      1000
           6       0.46      0.56      0.50      1000
           7       0.51      0.44      0.47      1000
           8       0.58      0.61      0.59      1000
           9       0.49      0.55      0.52      1000

    accuracy                           0.47     10000
   macro avg       0.46      0.47      0.46     10000
weighted avg       0.46      0.47      0.46     10000
```

Multi layered perceptron model

- The mean score of 5 fold cross validation comes out to be: 55.235.
- We had made some changes in order to increase the accuracy but it didn't work for us, it took a lot of time, still the classifier failed to fit.

<u>NOTE:</u> Sequential CNN model with data augmentation gives the best results and accuracy.

Challenges Faced During Project

- Choosing the best classifier was a challenge for us as all the classifiers do not give better results and we cannot be biased that only this classifier will work in this project so we have to search for different possibilities.
- Increasing the accuracy is also a great challenge of any project and so was in our case, we tried various methods to increase the accuracy like adding data augmentation, changing the parameters of the sequential classifier used, increasing and decreasing the number of different layers, standardising the data etc.

Learning from the Project

- The project provided us a deep knowledge of Neural networks and how we can effectively use various Neural nets in order to find the best accuracy.
- We get through how adding different layers in a neural net helps.
- This project taught us to deal with the image classification problems and what are the different classifiers we can use for them.
- We also learned how we can resolve errors encountered during the run.
- In order to increase the accuracy we searched and explored various method in that domain which leads us to gain a lot of information

Contribution of Team Members

We tried to distribute the work among all the three members as much as possible and here are the estimated contributions of each group members:

- Manav Gopal- Designed the first basic Machine learning pipeline using CNN model whole accuracy was very low which was further enhanced by other team members which was helped by me and made the document of the project by reading about CNN model from different online sources and documenting the classifier. Identified the learning from the project.
- Kotha Pranith Reddy- Compared different classifiers in order to find out the best classifier which gives the maximum accuracy. Used MLP classifier and Random forest classifier and tried to increase the accuracy as much possible by trying different classifiers. Also worked on increasing the accuracy. Identified the challenges faced in the project.

- Kartik Narayan- Done the preprocessing and visualisation of the CIFAR-10 dataset and how we can transform and manipulate them to use in different models. Tried out different methods to increase the accuracy of the classifiers like putting different layers in the CNN dataset and using data augmentation. Also helped in making the documentation on latex.

Acknowledgement

We would like to express our sincere gratitude to several individuals for supporting us throughout this course and projects. First, we wish to express our sincere gratitude to our supervisors, Dr. Richa Singh, Dr.Yashaswi Verma and Dr. Romi Banerjee for their enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped us tremendously at all times to understand the course throughout and develop interest in machine learning. Their immense knowledge, profound experience and professional expertise in Data Quality Control has enabled us to complete this research successfully. Secondly, also we would like to express our gratitude towards the TA's of this course who continuously helped us to build a proper intuition of the course. Without the support and guidance of TA's and professors, this project would not have been possible. We could not have imagined having a better supervisor in my study.

References

[1] Handwritten digit recognition with a back-propagation network, Y. LeCun, In Proc. Advances in Neural Information Processing Systems, (1990).

[2] ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky, A., Sutskever, I. and Hinton, G. E., NIPS: Neural Information Processing Systems, (2012).

[3] Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, K. Fukushima, Biological Cybernetics, 36, 4, April (1980).

[4] Receptive fields of single neurons in the cat's striate cortex, Hubel, D.H.; Wiesel, T.N., J Physiol. 148 (3), (1959).

[5] Deep learning, LeCun, Y., Bengio, Y., and Hinton, G., Nature, 521, May, (2015).

[6] CIFAR10 Dataset: Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009. Images collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton of the Canadian Institute For Advanced Research, available at: https://www.cs.toronto.edu/~kriz/cifar.html.

[7] A Comprehensive Guide to Convolutional Neural Networks-the ELI5 way, Sumit Saha, Towards data science, Dec, (2018)

[8] Convolutional Neural Network- From Wikipedia,the free encyclopedia

[9] Application of the residue number system to reduce hardware costs of the convolutional neural network implementation, Mathematics and Computer in Simulation, 232-243, M.V Valueva, N.N Nagornov, G.V Valuev, N.I Chervyakov, Nov, (2020)