Div: B
Roll No.: 422004
Gr. No.: 21810939
Name of Student: Aakanksha Bhondve
subject: Advanced Machine Learning

Assignment No. 4
Title: Implement Convolutioal Neural Network for Image Classification.

Theory:

1) Neural Network:

Neural networks are parallel computing devices, which is basically an attempt to make a computer model of the brain. The main objective is to develop a system to perform various computational tasks faster than the traditional systems. These tasks include pattern recognition and classification, approximation, optimization, and data clustering.

Deep Learning Algorithms:
Deep learning is based on the branch of machine learning, which is a subset of artificial intelligence. Since neural networks imitate the human brain and so deep learning will do. In deep learning, nothing is programmed explicitly. Basically, it is a machine learning class that makes use of numerous nonlinear processing units so as to perform feature extraction as well as transformation. The output from each preceding layer is taken as input by each one of the successive layers. Deep learning models are capable enough to focus on the accurate features themselves by requiring a little guidance from the programmer and are very helpful in solving out the problem of dimensionality. Deep learning algorithms are used, especially when we have a huge number of inputs and outputs. Deep learning can be defined as the method of machine learning and artificial intelligence that is intended to intimidate humans and their actions based on certain human brain functions to make effective decisions. It is a very important data science element that channels its modeling based on data-driven techniques under predictive modeling and statistics. To drive such a human-like ability to adapt and learn and to function accordingly, there have to be some strong forces which we popularly call algorithms.

Working of CNN:

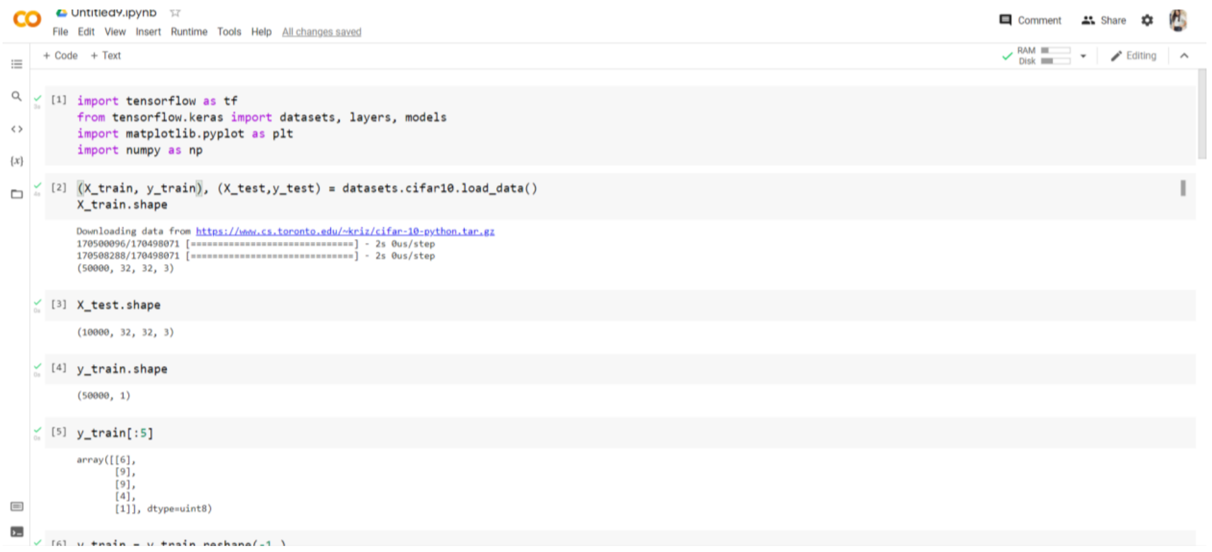
Generally, a Convolutional Neural Network has three layers, which are as follows;

- Input: If the image consists of 32 widths, 32 height encompassing three R, G, B channels, then it will hold the raw pixel([32x32x3]) values of an image.
- Convolution: It computes the output of those neurons, which are associated with input's local regions, such that each neuron will calculate a dot product in between weights and a small region to which they are actually linked to in the input volume. For example, if we choose to incorporate 12 filters, then it will result in a volume of [32x32x12].

● ReLU Layer: It is specially used to apply an activation function elementwise, like as max (0, x) thresholding at zero. It results in ([32x32x12]), which relates to an unchanged size of the volume.
● Pooling: This layer is used to perform a downsampling operation along the spatial dimensions (width, height) that results in [16x16x12] volume.
● Locally Connected: It can be defined as a regular neural network layer that receives an input from the preceding layer followed by computing the class scores and results in a 1-Dimensional array that has the equal size to that of the number of classes.

Implementation link:
https://colab.research.google.com/drive/19LvtYslxo1yLemepgX9A_ejm9fqdco3a

Code & Output:



```
[1] import tensorflow as tf
    from tensorflow.keras import datasets, layers, models
    import matplotlib.pyplot as plt
    import numpy as np
```

```
[2] (X_train, y_train), (X_test,y_test) = datasets.cifar10.load_data()
    X_train.shape
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 2s 0us/step
170508288/170498071 [==============================] - 2s 0us/step
(50000, 32, 32, 3)
```

```
[3] X_test.shape
```

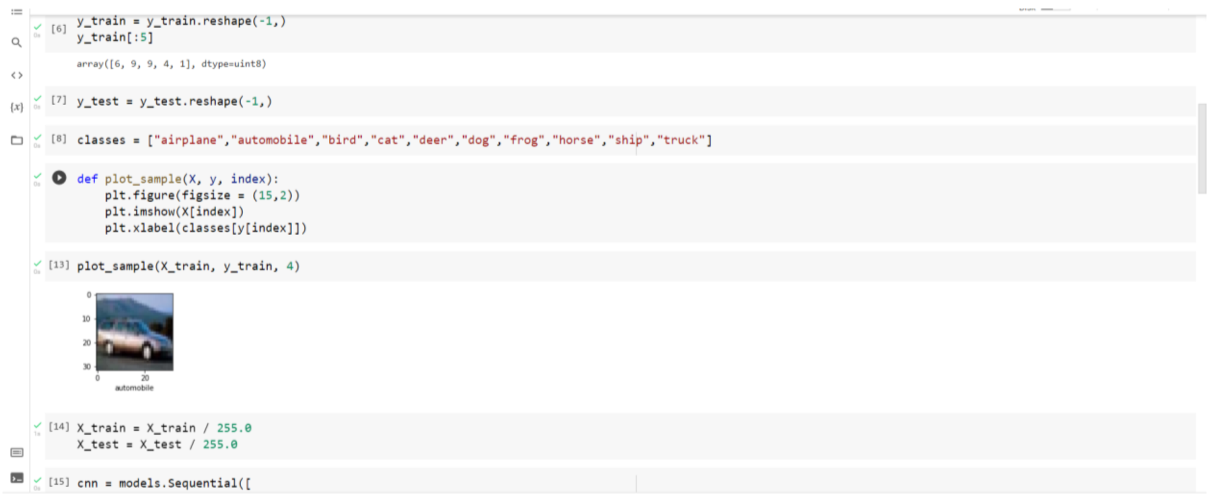
```
(10000, 32, 32, 3)
```

```
[4] y_train.shape
```

```
(50000, 1)
```

```
[5] y_train[:5]
```

```
array([[6],
       [9],
       [9],
       [4],
       [1]], dtype=uint8)
```



```
[6] y_train = y_train.reshape(-1,)
    y_train[:5]
```

```
array([6, 9, 9, 4, 1], dtype=uint8)
```

```
[7] y_test = y_test.reshape(-1,)
```

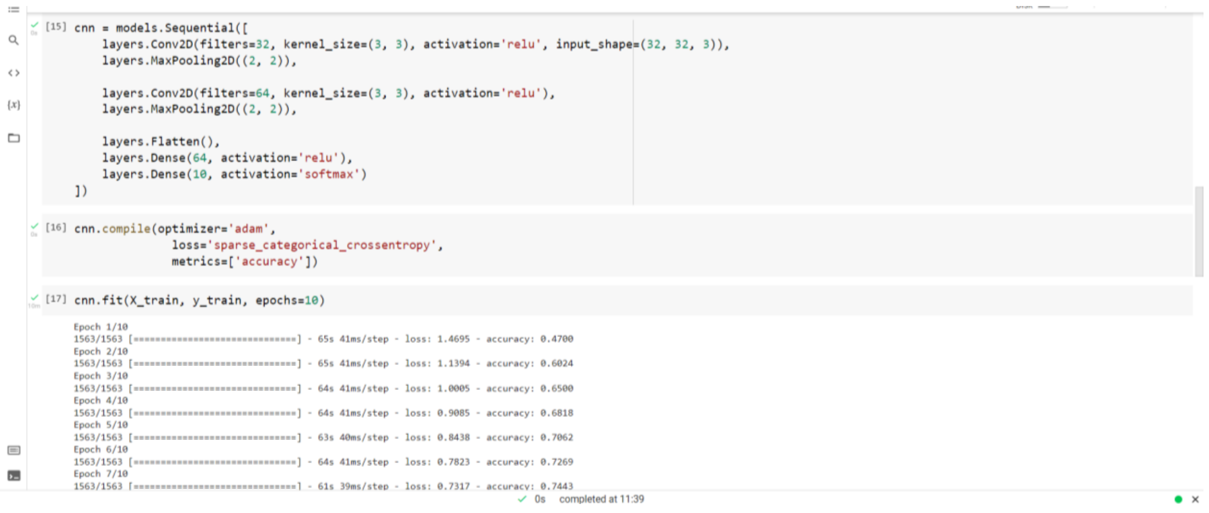
```
[8] classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```

```
def plot_sample(X, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])
```

```
[13] plot_sample(X_train, y_train, 4)
```

```
[14] X_train = X_train / 255.0
     X_test = X_test / 255.0
```

```
[15] cnn = models.Sequential([
```



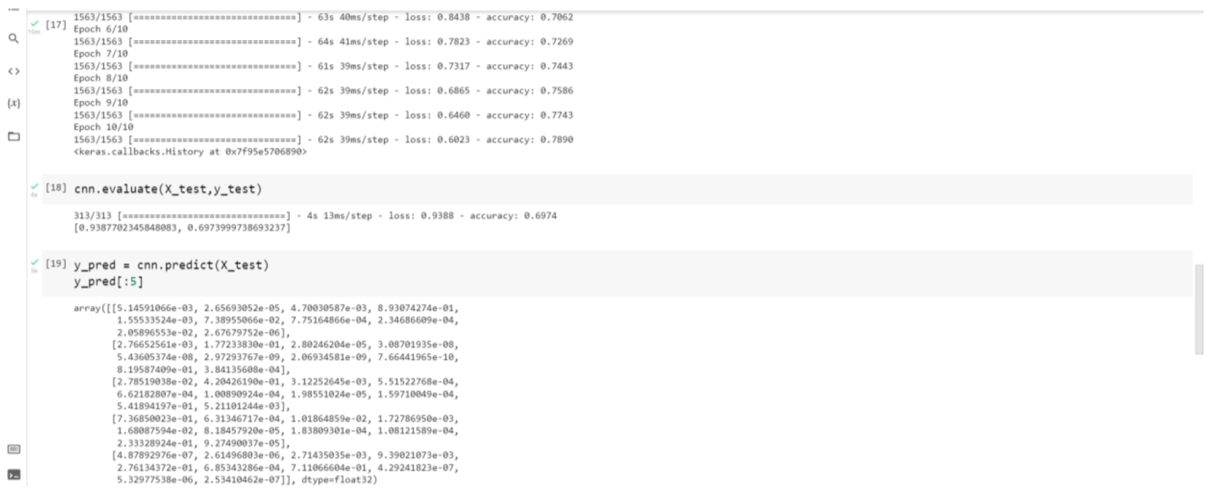
```
[15] cnn = models.Sequential([
         layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(32, 32, 3)),
         layers.MaxPooling2D((2, 2)),

         layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
         layers.MaxPooling2D((2, 2)),

         layers.Flatten(),
         layers.Dense(64, activation='relu'),
         layers.Dense(10, activation='softmax')
     ])
```

```
[16] cnn.compile(optimizer='adam',
                   loss='sparse_categorical_crossentropy',
                   metrics=['accuracy'])
```

```
[17] cnn.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
1563/1563 [==============================] - 65s 41ms/step - loss: 1.4695 - accuracy: 0.4700
Epoch 2/10
1563/1563 [==============================] - 65s 41ms/step - loss: 1.1394 - accuracy: 0.6024
Epoch 3/10
1563/1563 [==============================] - 64s 41ms/step - loss: 1.0005 - accuracy: 0.6500
Epoch 4/10
1563/1563 [==============================] - 64s 41ms/step - loss: 0.9085 - accuracy: 0.6818
Epoch 5/10
1563/1563 [==============================] - 63s 40ms/step - loss: 0.8438 - accuracy: 0.7062
Epoch 6/10
1563/1563 [==============================] - 64s 41ms/step - loss: 0.7823 - accuracy: 0.7269
Epoch 7/10
1563/1563 [==============================] - 61s 39ms/step - loss: 0.7317 - accuracy: 0.7443
```

✓ 0s completed at 11:39



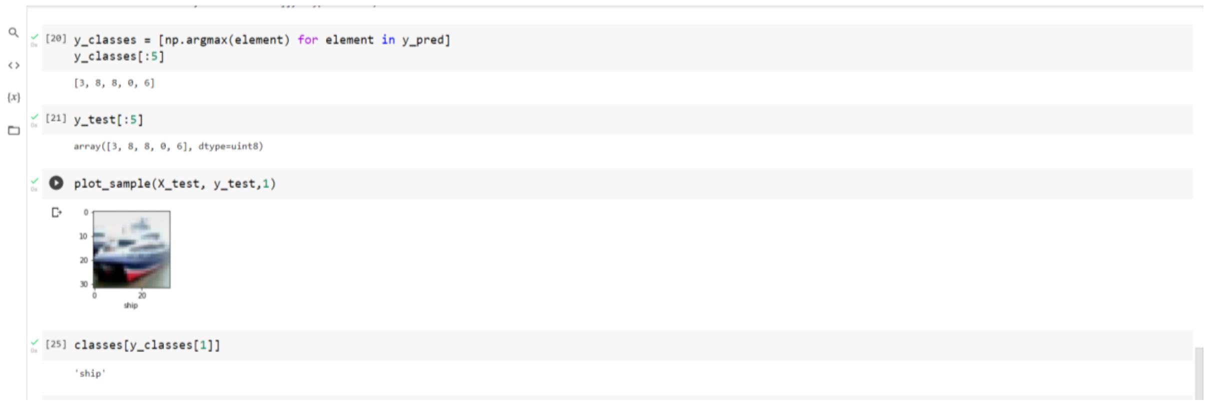
```
[17] 1563/1563 [==============================] - 63s 40ms/step - loss: 0.8438 - accuracy: 0.7062
     Epoch 6/10
     1563/1563 [==============================] - 64s 41ms/step - loss: 0.7823 - accuracy: 0.7269
     Epoch 7/10
     1563/1563 [==============================] - 61s 39ms/step - loss: 0.7317 - accuracy: 0.7443
     Epoch 8/10
     1563/1563 [==============================] - 62s 39ms/step - loss: 0.6865 - accuracy: 0.7586
     Epoch 9/10
     1563/1563 [==============================] - 62s 39ms/step - loss: 0.6460 - accuracy: 0.7743
     Epoch 10/10
     1563/1563 [==============================] - 62s 39ms/step - loss: 0.6023 - accuracy: 0.7890
     <keras.callbacks.History at 0x7f95e5706890>
```

```
[18] cnn.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 4s 13ms/step - loss: 0.9388 - accuracy: 0.6974
[0.9387702345848083, 0.6973999738693237]
```

```
[19] y_pred = cnn.predict(X_test)
     y_pred[:5]
```

```
array([[5.14591066e-03, 2.65693052e-05, 4.70030587e-03, 8.93074274e-01,
        1.55533524e-03, 7.38955066e-02, 7.75164866e-04, 2.34686609e-04,
        2.05896553e-02, 2.67679752e-06],
       [2.76652561e-03, 1.77233830e-01, 2.80246204e-05, 3.08701935e-08,
        5.43605374e-08, 2.97293767e-09, 2.06934581e-09, 7.66441965e-10,
        8.19587409e-01, 3.84135608e-04],
       [2.78519038e-02, 4.20426190e-01, 3.12252645e-03, 5.51522768e-04,
        6.62182807e-04, 1.00890924e-04, 1.98551024e-05, 1.59710049e-04,
        5.41894197e-01, 5.21101244e-03],
       [7.36850023e-01, 6.31346717e-04, 1.01864859e-02, 1.72786950e-03,
        1.68087594e-02, 8.18457920e-05, 1.83809301e-04, 1.08121589e-04,
        2.33328924e-01, 9.27490037e-05],
       [4.87892976e-07, 2.61496803e-06, 2.71435035e-03, 9.39021073e-03,
        2.76134372e-01, 6.85343286e-04, 7.11066604e-01, 4.29241823e-07,
        5.32977538e-06, 2.53410462e-07]], dtype=float32)
```



```
[20] y_classes = [np.argmax(element) for element in y_pred]
     y_classes[:5]
```

```
[3, 8, 8, 0, 6]
```

```
[21] y_test[:5]
```

```
array([3, 8, 8, 0, 6], dtype=uint8)
```

```
plot_sample(X_test, y_test,1)
```

```
[25] classes[y_classes[1]]
```

```
'ship'
```