**Batch: HO-DL-1**

**Roll Number: 16010421073 Experiment Number:5**

**Name: Keyur Patel**

**Title of the Experiment: Convolutional Neural Network**

**Program:**

**Import requisite libraries using Tensorflow and Keras.**

import tensorflow as tf

from tensorflow import keras

import numpy as np

import matplotlib.pyplot as plt

**Load the selected dataset.**

(X\_train, y\_train), (X\_test, y\_test) = keras.datasets.cifar10.load\_data()

****

**Visualize and display random images belonging to each class.**

class\_names = ["airplane", "automobile", "bird", "cat", "deer","dog", "frog", "horse", "ship", "truck"]

plt.figure(figsize=(10,10))

for i in range(10):

  idx = np.where(y\_train == i)[0][0]

  plt.subplot(2, 5, i+1)

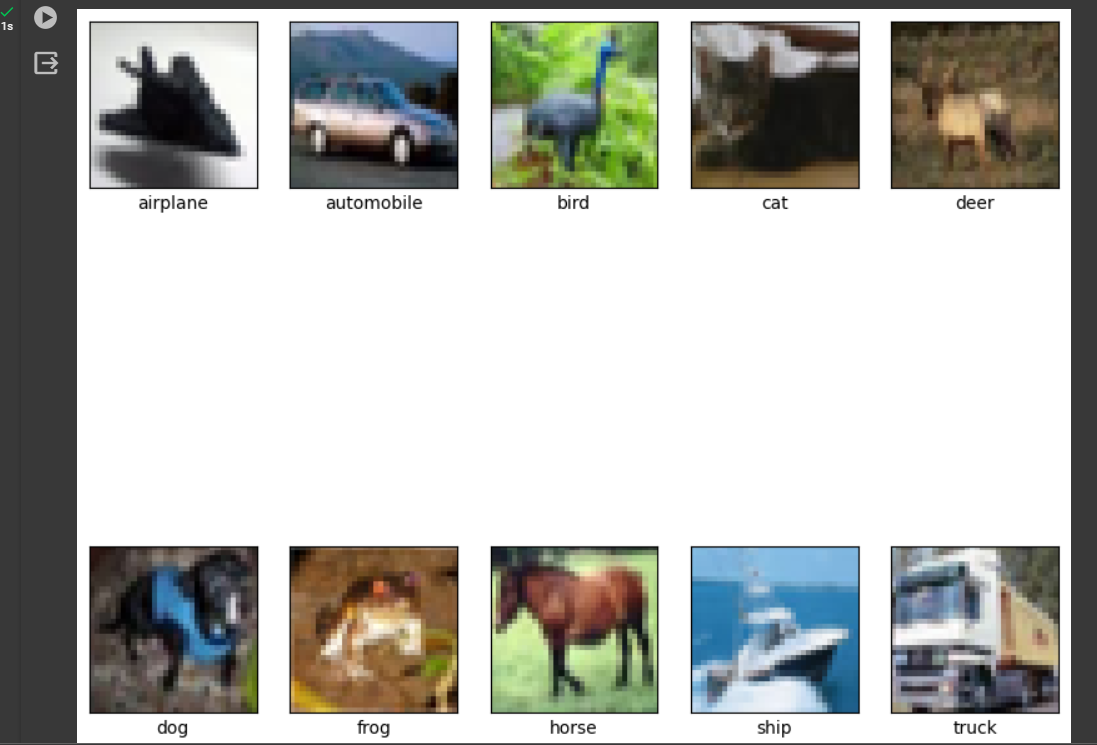
  plt.xticks([])

  plt.yticks([])

  plt.imshow(X\_train[idx])

  plt.xlabel(class\_names[i])

plt.show()

****

**Develop the CNN model.**

model = keras.Sequential([keras.layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(32,32, 3)),

  keras.layers.MaxPooling2D((2, 2)),

  keras.layers.Conv2D(64, (3, 3), activation='relu'),

  keras.layers.MaxPooling2D((2, 2)),

  keras.layers.Conv2D(64, (3, 3), activation='relu'), keras.layers.Flatten(),

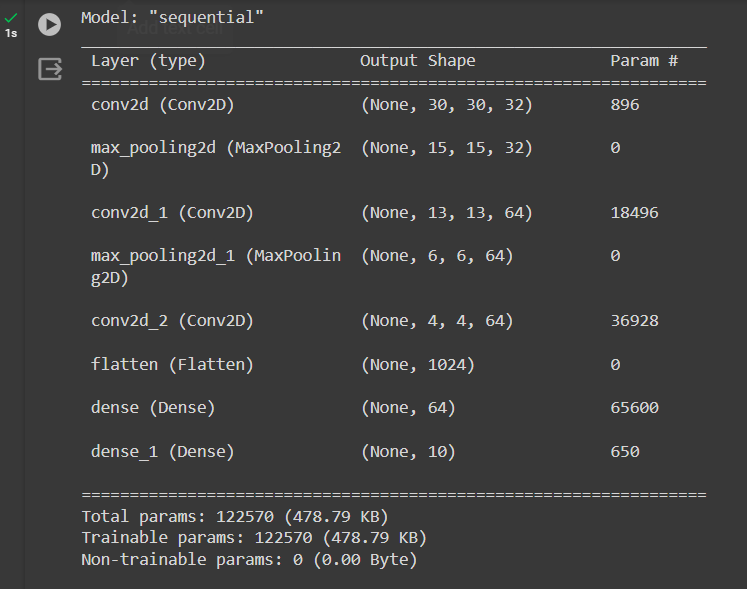
  keras.layers.Dense(64, activation='relu'), keras.layers.Dense(10)

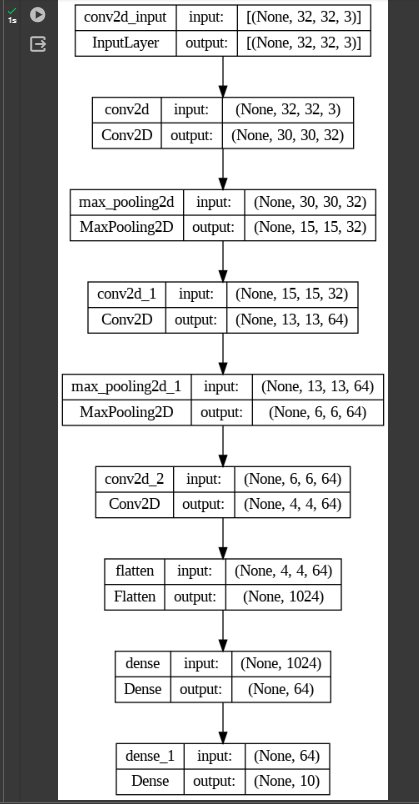
])

**Print Model Summary and display architecture diagram.**

model.summary()

keras.utils.plot\_model(model, show\_shapes=True)

****

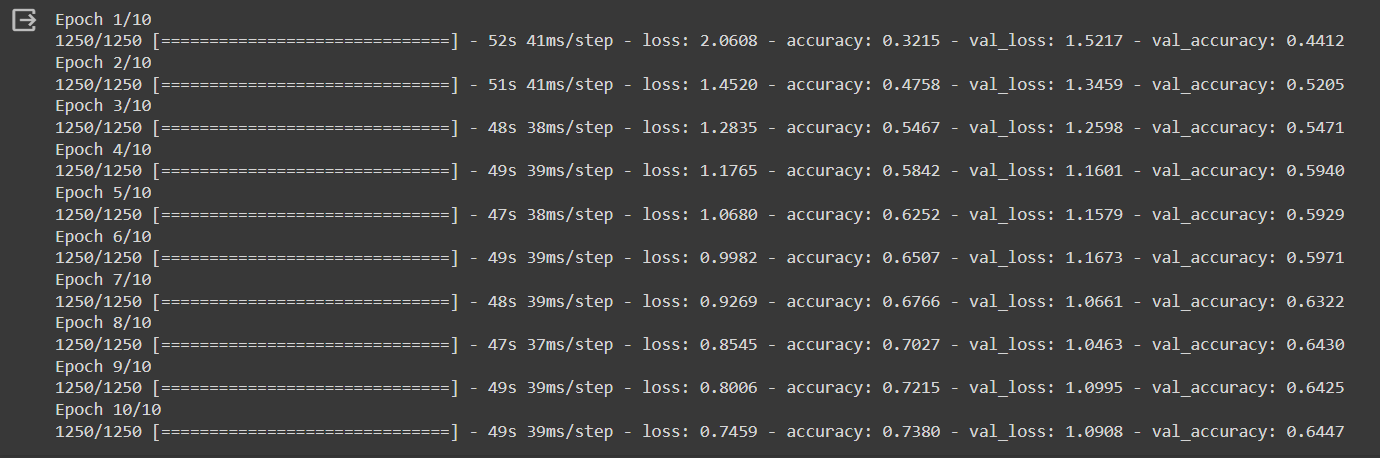
****

**Compile and fit the model on train dataset.**

model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True), metrics=['accuracy'])

history = model.fit(X\_train, y\_train, epochs=10, validation\_split=0.2)

****

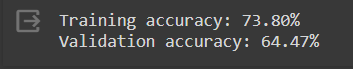
**Calculate training and the cross-validation accuracy.**

train\_acc = history.history['accuracy'][-1]

val\_acc = history.history['val\_accuracy'][-1]

print("Training accuracy: {:.2f}%".format(train\_acc\*100))

print("Validation accuracy: {:.2f}%".format(val\_acc\*100))

****

**Redefine the model by using appropriate regularization technique to prevent overfitting.**

from tensorflow.keras import regularizers

reg\_model = keras.Sequential([keras.layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(32,32, 3),

  kernel\_regularizer=regularizers.l2(0.001)),

  keras.layers.MaxPooling2D((2, 2)),

  keras.layers.Conv2D(64, (3, 3), activation='relu',

  kernel\_regularizer=regularizers.l2(0.001)),

  keras.layers.MaxPooling2D((2, 2)),

  keras.layers.Conv2D(64, (3, 3),

  activation='relu',

  kernel\_regularizer=regularizers.l2(0.001)),

  keras.layers.Flatten(),

  keras.layers.Dense(64, activation='relu',kernel\_regularizer=regularizers.l2(0.001)),

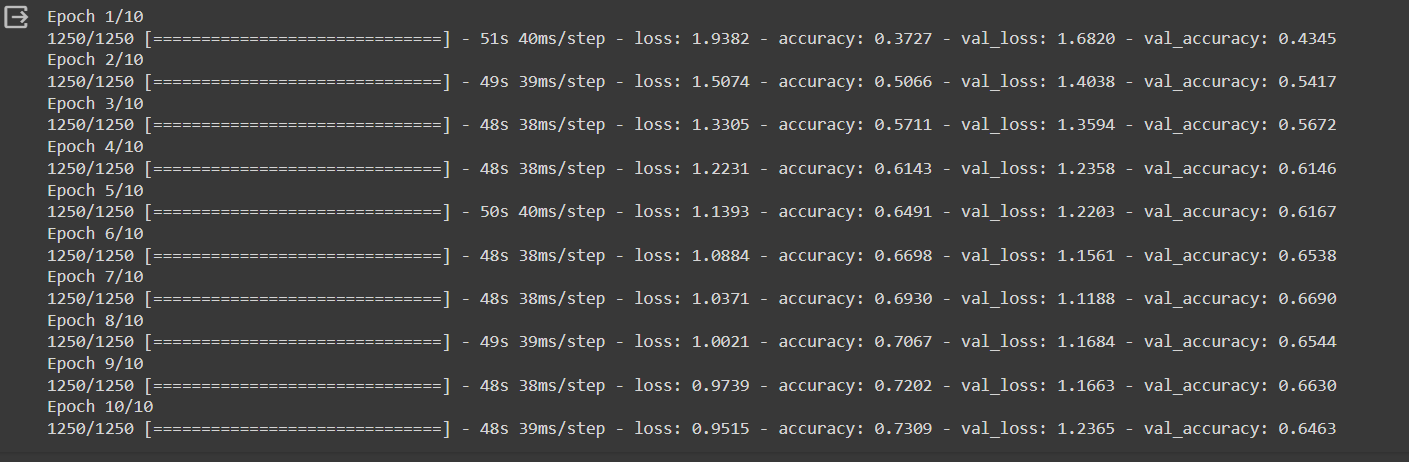
  keras.layers.Dense(10)

])

**Fit the data on the regularized model.**

reg\_model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True), metrics=['accuracy'])

reg\_history = reg\_model.fit(X\_train, y\_train, epochs=10, validation\_split=0.2)

****

**Calculate and plot loss function and accuracy using suitable loss function.**

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.legend()

plt.title('Loss Function')

plt.show()

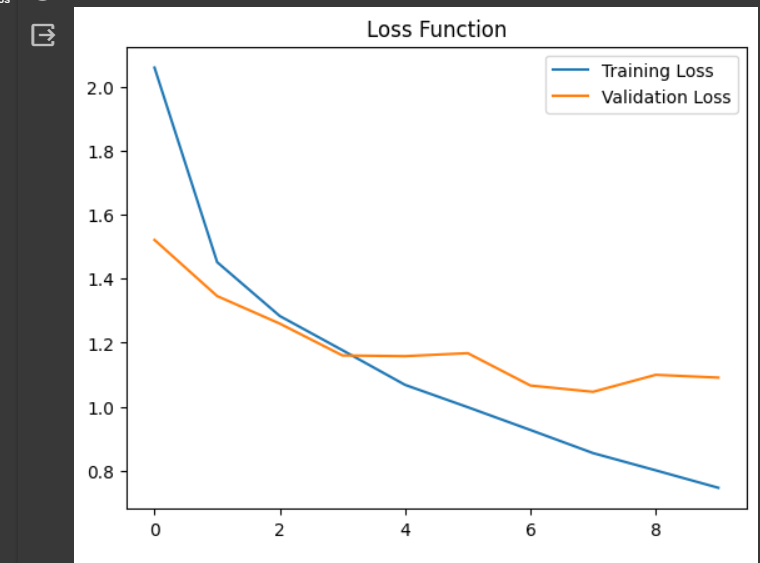
plt.plot(history.history['accuracy'], label='Training Accuracy')

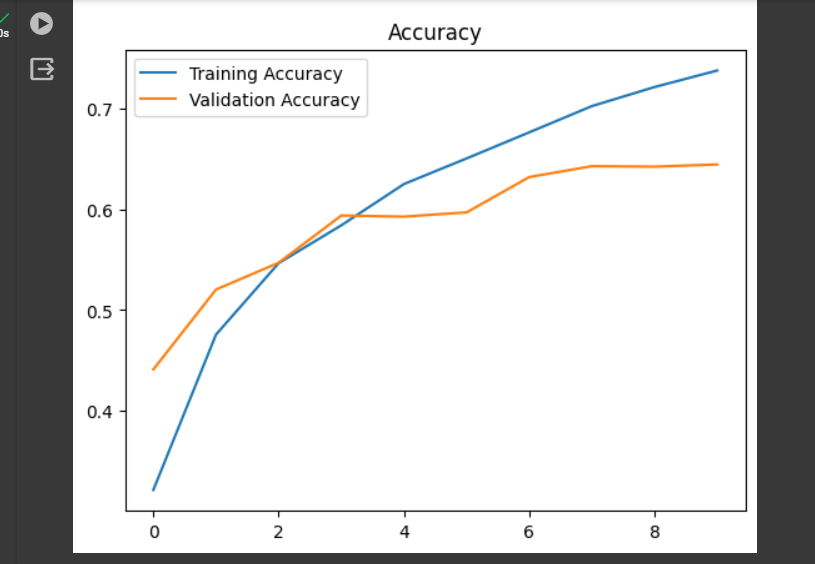
plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.legend()

plt.title('Accuracy')

plt.show()

****

****

**Display classification Report for un-regularized CNN model.**

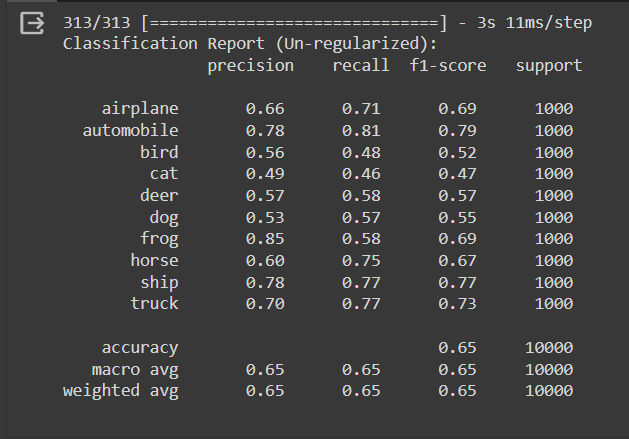
from sklearn.metrics import classification\_report

y\_pred = model.predict(X\_test)

y\_pred\_classes = np.argmax(y\_pred, axis=1)

print("Classification Report (Un-regularized):\n",

classification\_report(y\_test, y\_pred\_classes, target\_names=class\_names))

****

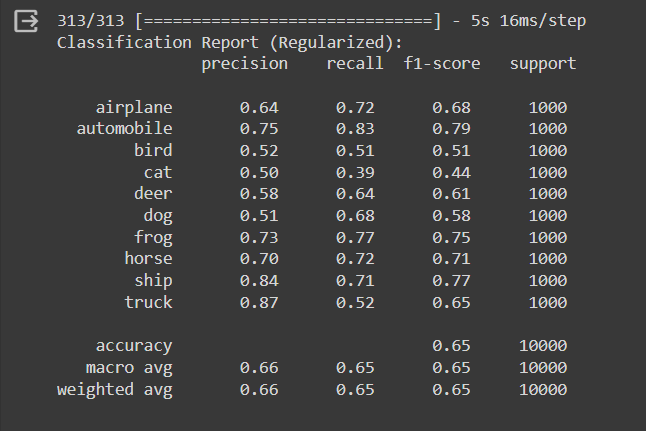
**Display classification Report for regularized CNN model.**

y\_pred = reg\_model.predict(X\_test)

y\_pred\_classes = np.argmax(y\_pred, axis=1)

print("Classification Report (Regularized):\n",

classification\_report(y\_test, y\_pred\_classes, target\_names=class\_names))

****

**Comment on output.**

The CIFAR-10 dataset was used to train a model, and the results indicate that the regularization techniques employed were effective in preventing overfitting. This can be observed through the lower regularization loss and higher accuracy when compared to the non-regularized version of the model.

The classification report reveals that the model's performance varies across different classes, with some classes being classified more accurately than others. This suggests that further improvements could be made, such as acquiring additional training data or refining the model architecture, to enhance the overall classification performance.

**CO3 :** Assimilate fundamentals of Convolutional Neural Network.

**Conclusion:** We have successfully implemented a Convolutional Neural Network (CNN) model.