Name – Akshat Bhargava

Roll no 22563

ANN – SL 2

Prac 11

Code –

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers, models

import matplotlib.pyplot as plt

import numpy as np

import os

# Load and preprocess dataset (CIFAR-10 for demonstration)

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

# Normalize pixel values to be between 0 and 1

x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

# Convert labels to categorical format

y\_train, y\_test = keras.utils.to\_categorical(y\_train, 10), keras.utils.to\_categorical(y\_test, 10)

# Data Augmentation

data\_augmentation = keras.Sequential([

layers.RandomFlip("horizontal"),

layers.RandomRotation(0.1),

layers.RandomZoom(0.1),

])

# Build CNN Model

def build\_model():

model = models.Sequential([

data\_augmentation,

layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 3)),

layers.BatchNormalization(),

layers.MaxPooling2D((2, 2)),

layers.Dropout(0.25),

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

layers.BatchNormalization(),

layers.MaxPooling2D((2, 2)),

layers.Dropout(0.25),

layers.Conv2D(128, (3, 3), activation='relu'),

layers.BatchNormalization(),

layers.MaxPooling2D((2, 2)),

layers.Dropout(0.25),

layers.Flatten(),

layers.Dense(256, activation='relu'),

layers.BatchNormalization(),

layers.Dropout(0.5),

layers.Dense(10, activation='softmax')

])

return model

# Compile Model

model = build\_model()

model.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.001),

loss='categorical\_crossentropy',

metrics=['accuracy'])

# Train the Model

history = model.fit(x\_train, y\_train, epochs=20, batch\_size=64, validation\_data=(x\_test, y\_test))

# Plot Accuracy & Loss

fig, axs = plt.subplots(2, 1, figsize=(10, 8))

# Accuracy Graph

axs[0].plot(history.history['accuracy'], label='Train Accuracy')

axs[0].plot(history.history['val\_accuracy'], label='Validation Accuracy')

axs[0].set\_title('Model Accuracy')

axs[0].set\_ylabel('Accuracy')

axs[0].set\_xlabel('Epoch')

axs[0].legend()

# Loss Graph

axs[1].plot(history.history['loss'], label='Train Loss')

axs[1].plot(history.history['val\_loss'], label='Validation Loss')

axs[1].set\_title('Model Loss')

axs[1].set\_ylabel('Loss')

axs[1].set\_xlabel('Epoch')

axs[1].legend()

plt.tight\_layout()

plt.show()

# Evaluate Model

final\_loss, final\_acc = model.evaluate(x\_test, y\_test)

print(f"Final Test Accuracy: {final\_acc:.4f}")

Output –







