Convolutional Neural Network

(Using Tensor Flow, Keras and Sklearn)

Name: Mubeen Khalid Reg. no.: SP21-BSE-015

Class: BSE-6A Subject: Artificial Intelligence

# Lab Terminal

# Link to notebook:

# <https://colab.research.google.com/drive/1NDYpiI6BJoc7bmPxWGAOFw0OMUZ4INC1?usp=sharing>

# Source Code

## Model:

# Import necessary libraries

import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.datasets import cifar10

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.utils import to\_categorical

# Loading and preprocessing

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

# Normalizing pixel values

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

# encoding labels

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

# Data augmentation

datagen = ImageDataGenerator(rotation\_range=20, width\_shift\_range=0.2, height\_shift\_range=0.2, horizontal\_flip=True)

# CNN model

model = models.Sequential()

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

model.add(layers.BatchNormalization())

model.add(layers.MaxPooling2D((2, 2)))

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

model.add(layers.BatchNormalization())

model.add(layers.Flatten())

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

model.add(layers.Dropout(0.5))

model.add(layers.Dense(128, activation='relu'))

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

# Compilation

model.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=0.0001), loss='categorical\_crossentropy', metrics=['accuracy'])

# Training

batch\_size = 64

epochs = 100

history = model.fit(datagen.flow(x\_train, y\_train, batch\_size=batch\_size),

                    steps\_per\_epoch=len(x\_train) / batch\_size,

                    epochs=epochs,

                    validation\_data=datagen.flow(x\_test, y\_test, batch\_size=batch\_size))

# Evaluation

loss, accuracy = model.evaluate(x\_test, y\_test, verbose=1)

accuracy = 100\*accuracy

print(f'\nTest accuracy: {accuracy:.2f} percent')

# saving

model.save('CNN\_labFinalTF.keras')

## Classification Report:

from sklearn.metrics import classification\_report

import numpy as np

from tensorflow.keras.models import load\_model

# Load the trained model

model = load\_model('/content/CNN\_labFinalTF.keras')

# Make predictions on the test set

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

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

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

# Generate classification report

report = classification\_report(y\_true\_classes, y\_pred\_classes)

# Print the report

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

## Prediction:

import numpy as np

from tensorflow.keras.preprocessing import image

from tensorflow.keras.models import load\_model

# Load the trained model

model = load\_model('/content/CNN\_labFinalTF.keras')

img\_path = '/content/frog.jpeg'

img = image.load\_img(img\_path, target\_size=(32, 32))

img\_array = image.img\_to\_array(img)

img\_array = np.expand\_dims(img\_array, axis=0)

img\_array /= 255.0

# Make predictions

predictions = model.predict(img\_array)

# Get the predicted class label

predicted\_class = np.argmax(predictions[0])

# Label descriptions

label\_descriptions = {

    0: 'airplane',

    1: 'automobile',

    2: 'bird',

    3: 'cat',

    4: 'deer',

    5: 'dog',

    6: 'frog',

    7: 'horse',

    8: 'ship',

    9: 'truck',

}

# Get the predicted class description

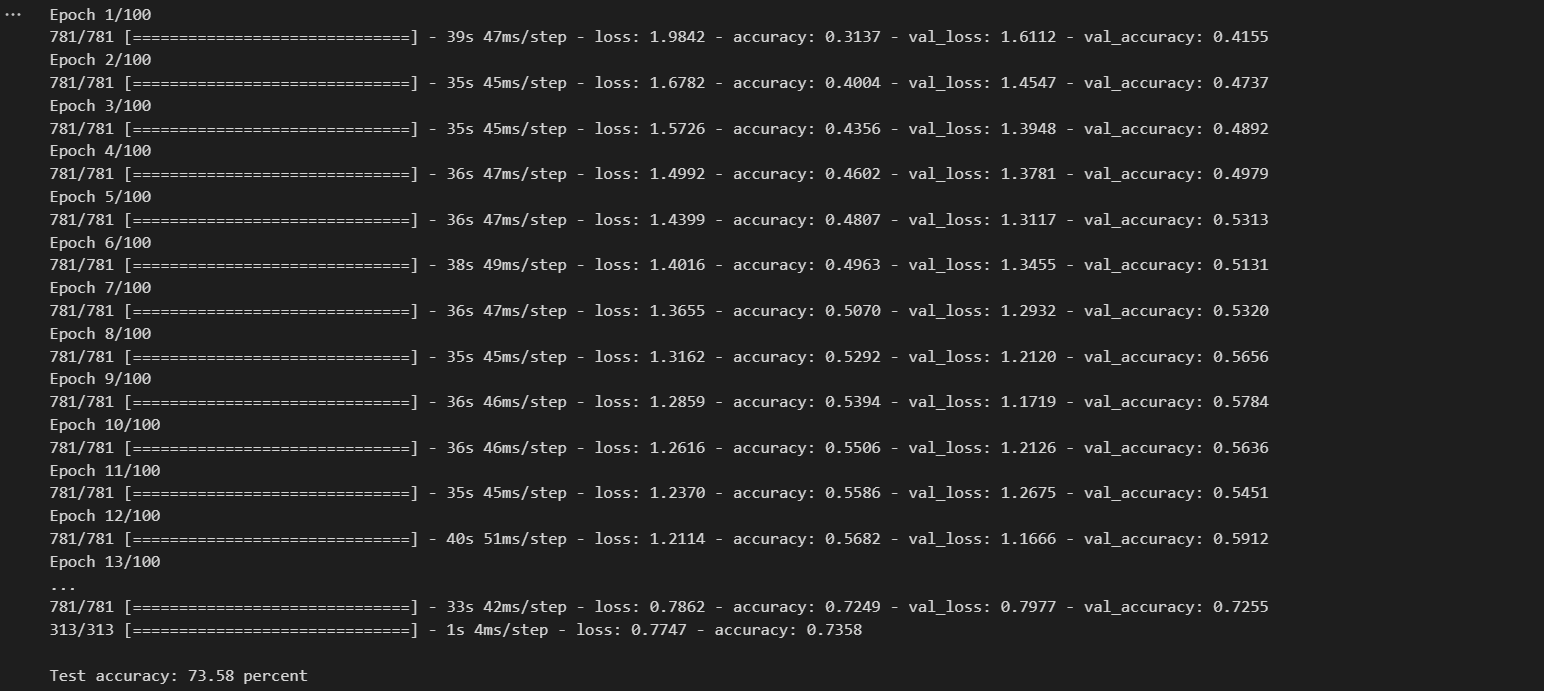
predicted\_description = label\_descriptions.get(predicted\_class, 'Unknown Class')

# Print the predicted class description

print(f'The predicted class is: {predicted\_description}')

# Output

## Accuracy:



## Input Image:

A green and blue frog on a leaf

Description automatically generated

## Prediction:

A black screen with white text

Description automatically generated

## Classification Report:

A screenshot of a computer screen

Description automatically generated