CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK.

A COURSE PROJCET REPORT

BY

A.S.THARUN(RA2011026010217)

G.MEGHANATH(RA2011026010219)

N.SHIVRAM(RA2011026010237)



DEPARTMENT OF COMPUTER INTELLIGENCE

18CSE388T-ARTIFICIAL NEURAL NETWORK

2022-2023 ODD SEMESTER, NOVEMBER 2022.

SUBMITTED TO: Dr. A JACKULIN MAHARIBA

TABLE OF CONTENTS

CHAPTER CONTENTS

1. PROBLEM STATEMENT
2. INTRODUCTION
3. METHODLOGY
4. IMPLEMENTATION AND RESULTS
5. INFERENCE
6. REFERENCE

1.PROBLEM STATEMENT

* This project aims to classify the given dataset into 10 unique classes through the use of Convolutional neural network(CNN)
* The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.
* The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

2.INTRODUCTION

THE NEURAL NETWORK:

* A Convolutional Neural Network is used in this project to classify the dataset
* A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other.
* Since our aim is to classify the dataset into 10 different classes, CNN is the best available neural network for the task.

ADVANTAGES OF CNN:

* CNNs eliminate the need for manual feature extraction—the features are learned directly by the CNN.
* CNNs produce highly accurate recognition results.
* CNNs can be retrained for new recognition tasks, enabling you to build on pre-existing networks.

WORKING OF A CNN:

* A convolutional neural network can have tens or hundreds of layers that each learn to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as very simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object.
* Like other neural networks, a CNN is composed of an input layer, an output layer, and many hidden layers in between.
* These layers perform operations that alter the data with the intent of learning features specific to the data. Three of the most common layers are: convolution, activation or ReLU, and pooling.

CONVOULUTION LAYER:

Convolution puts the input images through a set of convolutional filters, each of which activates certain features from the images.

RECTIFIED LINEAR UNIT (ReLU):

ReLu layer allows for faster and more effective training by mapping negative values to zero and maintaining positive values. This is sometimes referred to as activation, because only the activated features are carried forward into the next layer.

POOLING:

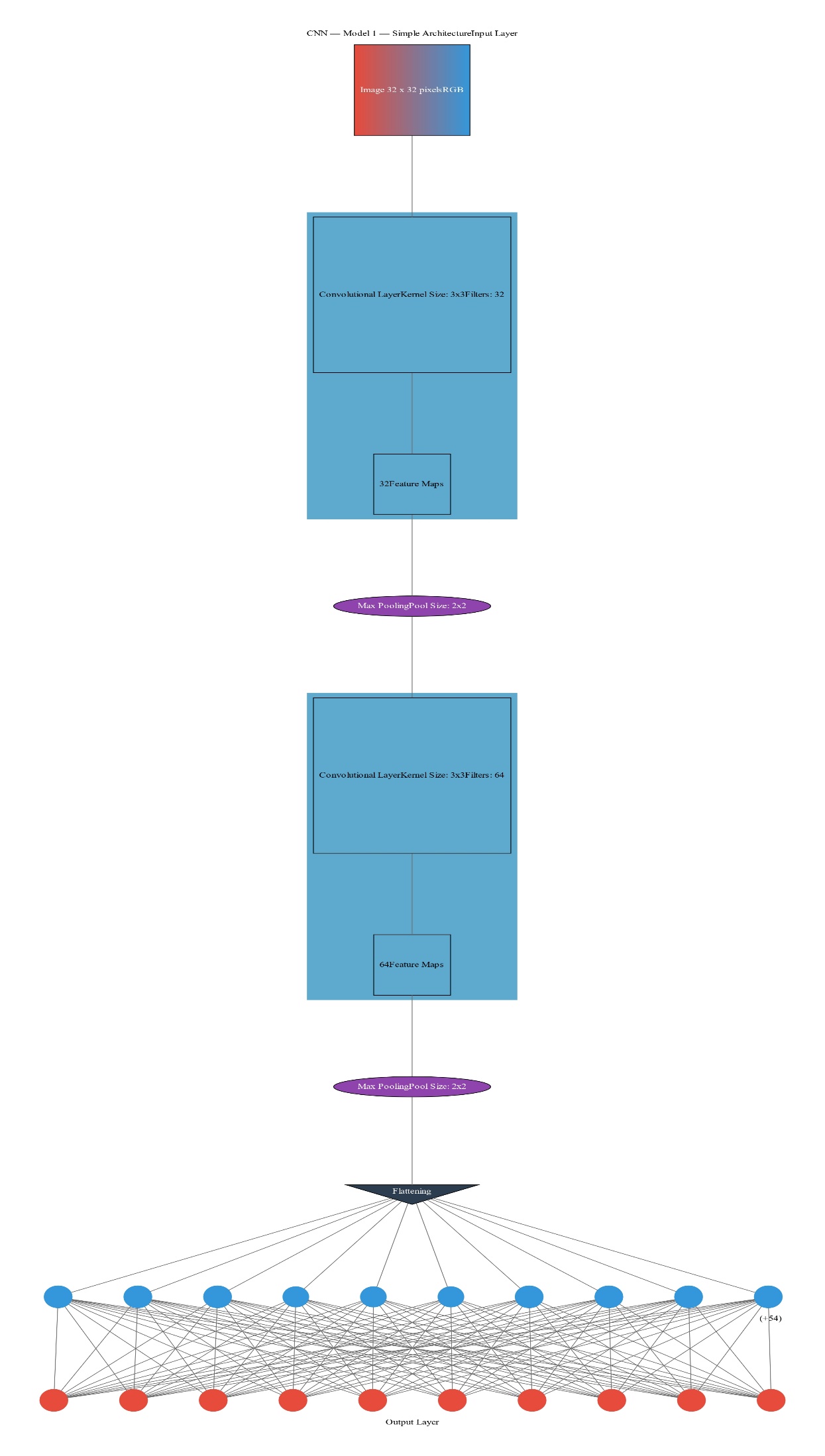
Pooling simplifies the output by performing nonlinear down sampling, reducing the number of parameters that the network needs to learn.



* The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

SOURCE: <https://www.cs.toronto.edu/~kriz/cifar.html>

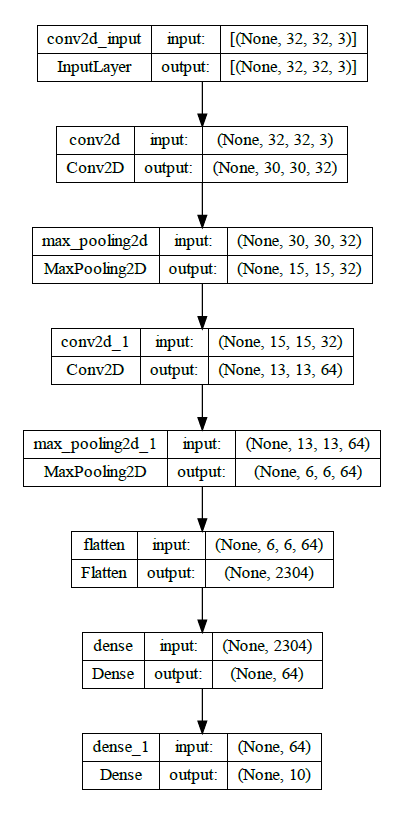
3.METHOLOGY



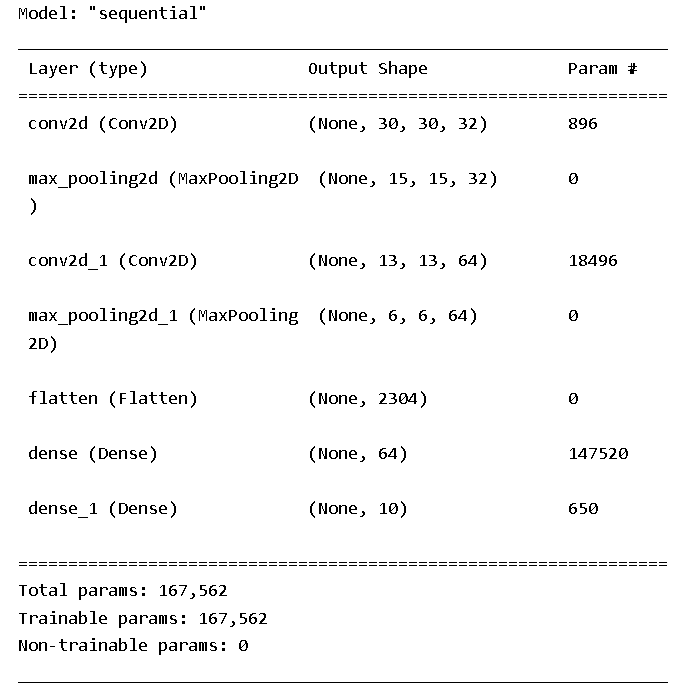
DATA FLOW:

* THE CNN has 2 sets of convulsion and pooling layers.
* A 32x32 image is given as input to the first convolution layer.
* This convolution layer gives a 32 size filter map.
* This 32 size feature map is reduced to 2x2 by the pooling layer.
* This 2x2 feature map is again given to the second convolution layer.
* This layer gives a 64x64 feature map, which is again reduced to 2x2 feature map.
* This 2x2 map is given to the input or the flattening layer, which flattens the inputs.
* The flatten layer transfer the data to dense or hidden layer with 64 neurons.
* After data processing the hidden layer transfers the data to the output layers which have 10 neurons.

MODEL LAYERS:



MODEL SUMMARY:



The number of trainable parameters are: 167,562.

4.Implementation and result

Importing Libraries:

import matplotlib.pyplot as plt

import numpy as np

import tensorflow as tf

from tensorflow.keras import datasets, layers, models

Data processing:

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

X\_train.shape

X\_test.shape

y\_train[:5]

y\_train = y\_train.reshape(-1,)

y\_train[:5]

y\_test = y\_test.reshape(-1,)

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]])

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

CNN MODEL:

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')

])

cnn.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

cnn.fit(X\_train, y\_train, epochs=10)

from keras.models import load\_model

cnn.save('annproject1.h5')

from keras.models import load\_model

loaded\_model = load\_model("annproject1.h5")

from sklearn.metrics import confusion\_matrix , classification\_report

import numpy as np

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

y\_pred\_classes = [np.argmax(element) for element in y\_pred]

print("Classification Report: \n", classification\_report(y\_test, y\_pred\_classes))

loaded\_model.evaluate(X\_test,y\_test)

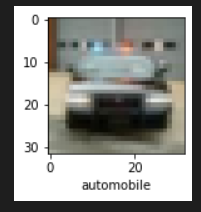
y\_pred = loaded\_model.predict(X\_test)

y\_pred[:5]

y\_classes = [np.argmax(element) for element in y\_pred]

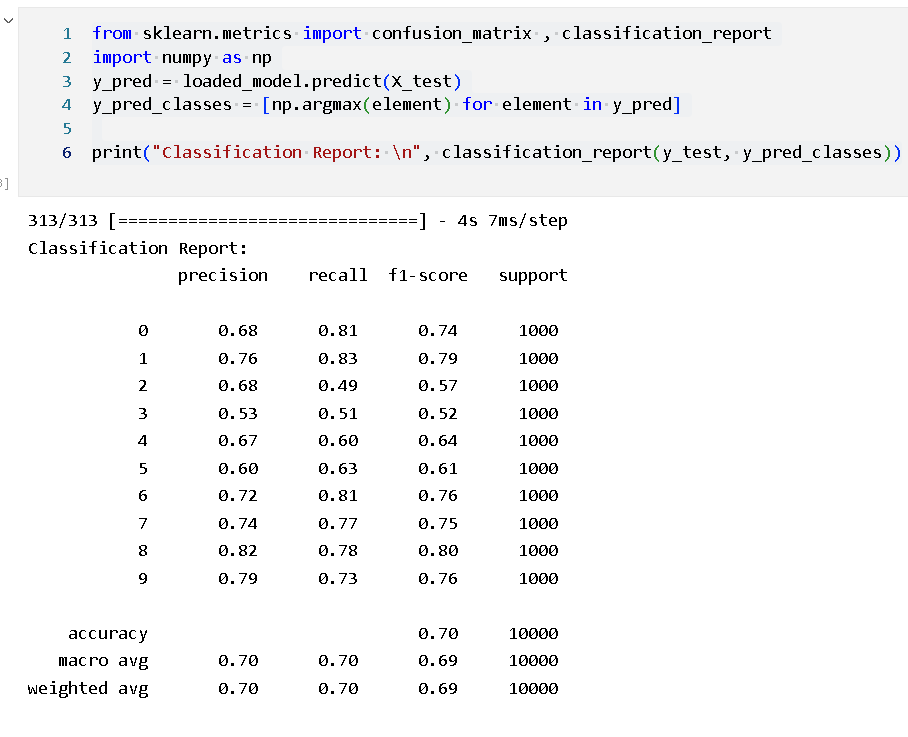
OUTPUT:

plot\_sample(X\_test, y\_test,66)

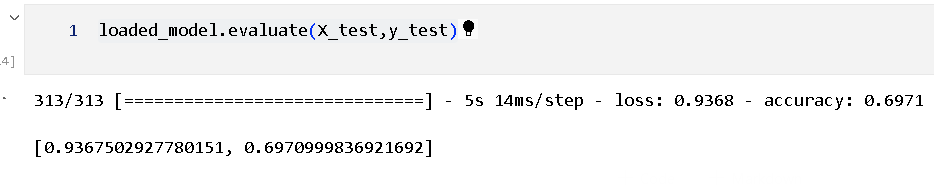


classes[y\_classes[66]]

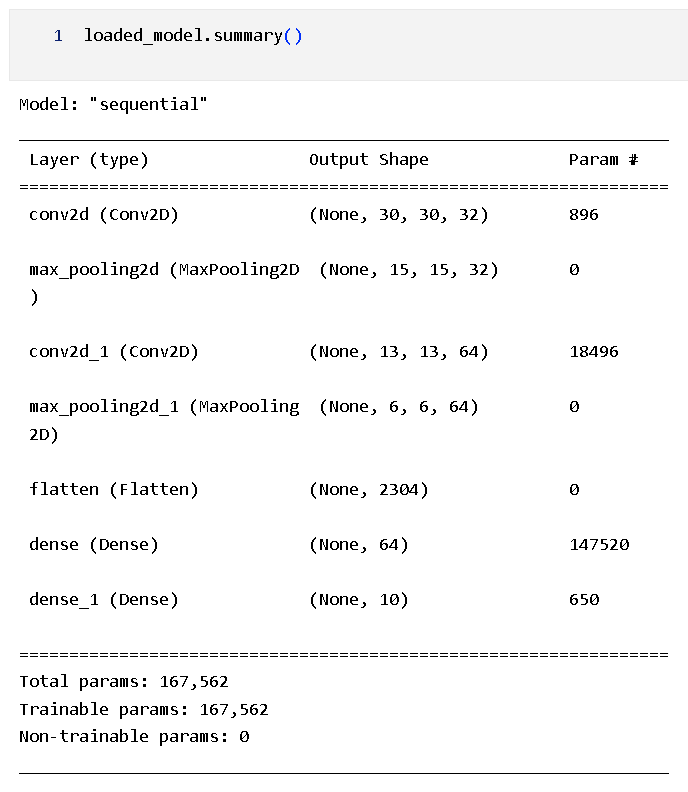
'automobile'



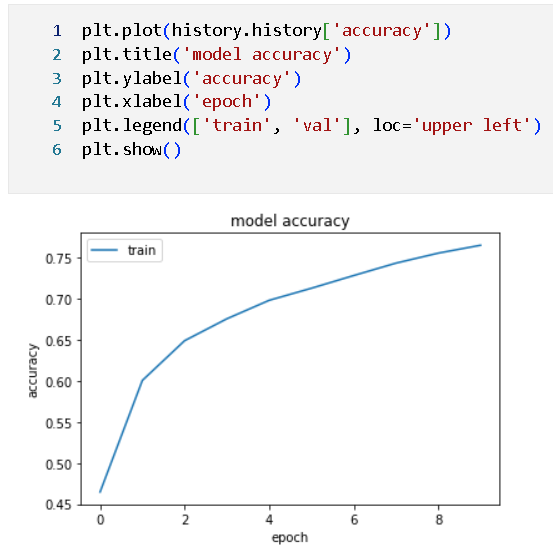
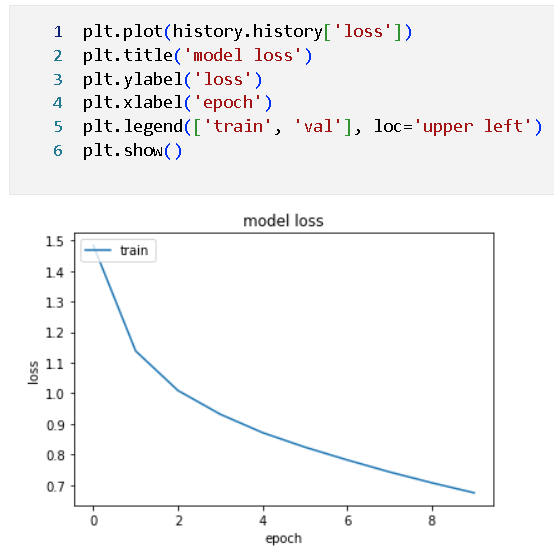
The above table depicts the accuracy of the model on the training sets.



The above diagram shows the accuracy obtained by the model on the testing sets. The accuracy achieved by the model is 69.7%.



The above table shows, the number of layers and the number of neurons present in those layers. The total number of trainable parameters are 167,562.

The above graph shows the variation of accuracy and loss with the increase in the number of epochs.

INFERENCE

The model was trained to classify the given dataset using the convolutional neural network algorithm. The model was trained on multiple iterations of the dataset to train the parameters and achieve the desired accuracy levels.

REFERENCES

Dataset: <https://www.cs.toronto.edu/~kriz/cifar.html>

CNN : <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

CNN : <https://www.geeksforgeeks.org/introduction-convolution-neural-network/>

TensorFlow guide : <https://www.tensorflow.org/guide>