***Architecture used in the Paper:***

***A picture containing timeline

Description automatically generated***

***Dataset details :-***

The dataset that used in the paper is (MNIST dataset). It is an acronym that stands for the Modified National Institute of Standards and Technology dataset. It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.

The dataset that used in the project is Animal\_classification\_dataset which was divided into 2 folders [‘train’,test’] , each folder contains 5 folders of the 5 classes ['elephant','farfalla','mucca','pecora','scoiattolo'] . in folder train each folder contains 160 images and in folder test each folder contains 40 images.

A collage of animals

Description automatically generated with medium confidence

***Implementation details :-***

loading the libraries and importing the necessary modules.

import numpy as np

import pandas as pd

import os

from sklearn.metrics import classification\_report

import seaborn as sn; sn.set(font\_scale=1.4)

from sklearn.utils import shuffle

import matplotlib.pyplot as plt

import cv2

import tensorflow as tf

from tqdm import tqdm

Declare variables that contain the class names and the corresponding label.

class\_names= ['elephant','farfalla','mucca','pecora','scoiattolo']

class\_names\_label={class\_name:i for i,class\_name in enumerate (class\_names)}

nb\_classes=len(class\_names)

print(class\_names\_label)

IMAGE\_SIZE=(300,300)

Define a function called load\_data which we can use to load train and test data.

def load\_data():

DIRECTORY="C:\ANIMALS"

CATEGORY=["seg\_train","seg\_test"]

output=[]

for category in CATEGORY:

path=os.path.join(DIRECTORY,category)

images=[]

labels=[]

print("loading {}".format(category))

for folder in os.listdir(path):

label=class\_names\_label[folder]

for file in os.listdir(os.path.join(path,folder)):

img\_path=os.path.join(os.path.join(path,folder),file)

image=cv2.imread(img\_path)

image=cv2.cvtColor(image,cv2.COLOR\_BGR2RGB)

image=cv2.resize(image,IMAGE\_SIZE)

images.append(image)

labels.append(label)

images=np.array(images,dtype='float32')

labels=np.array(labels,dtype='int32')

output.append((images,labels))

return output

*Call load\_data() function and save training and testing data.*

(train\_images,train\_labels),(test\_images,test\_labels)=load\_data()

train\_images,train\_labels =shuffle(train\_images,train\_labels,random\_state=25)

*Creating a Convolution Neural Network with Keras.*

model = tf.keras.Sequential([

tf.keras.layers.Conv2D(32,(3,3),activation='relu',input\_shape=(300,300,3)),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(32,(3,3),activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(32,(3,3),activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(32,(3,3),activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(32,(3,3),activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(128,activation=tf.nn.relu),

tf.keras.layers.Dense(6,activation=tf.nn.softmax)

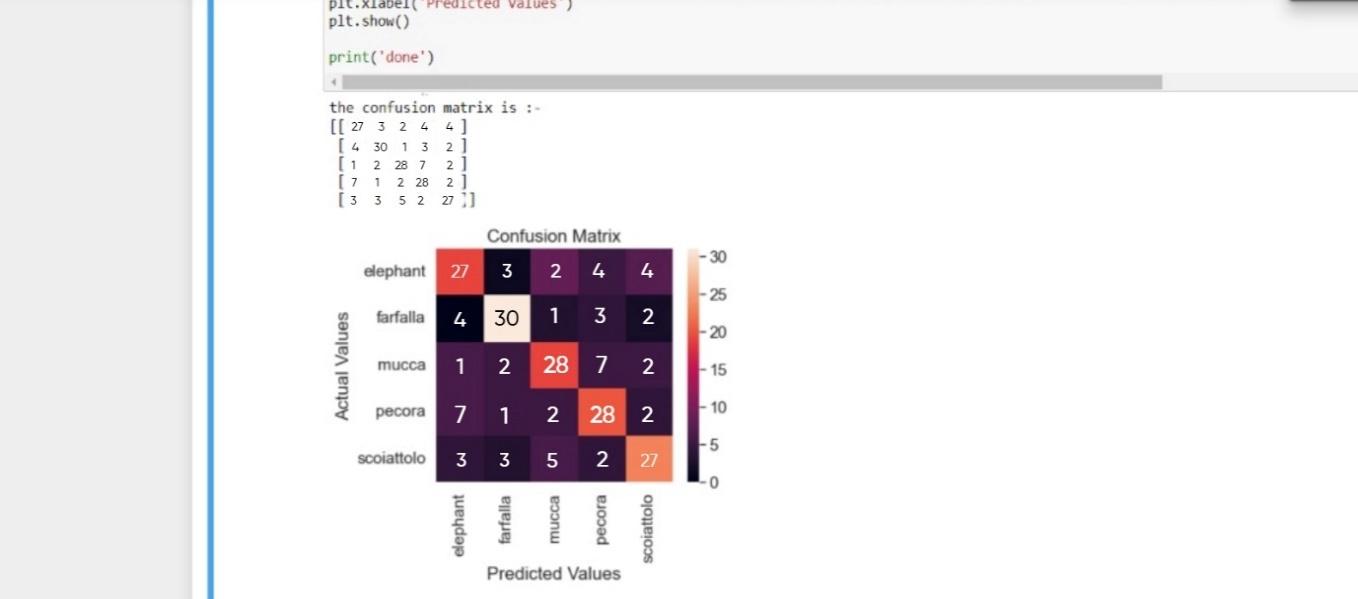
])

model.compile(optimizer='adam',loss='sparse\_categorical\_crossentropy',metrics=['accuracy'])

history=model.fit(train\_images,train\_labels,batch\_size=50,epochs=50,validation\_split=0.2)

*Results and visualizations:-*

*The confusion\_matrix:*



*Accuracy curve:*

*Accuracy= 0.70*

Graphical user interface

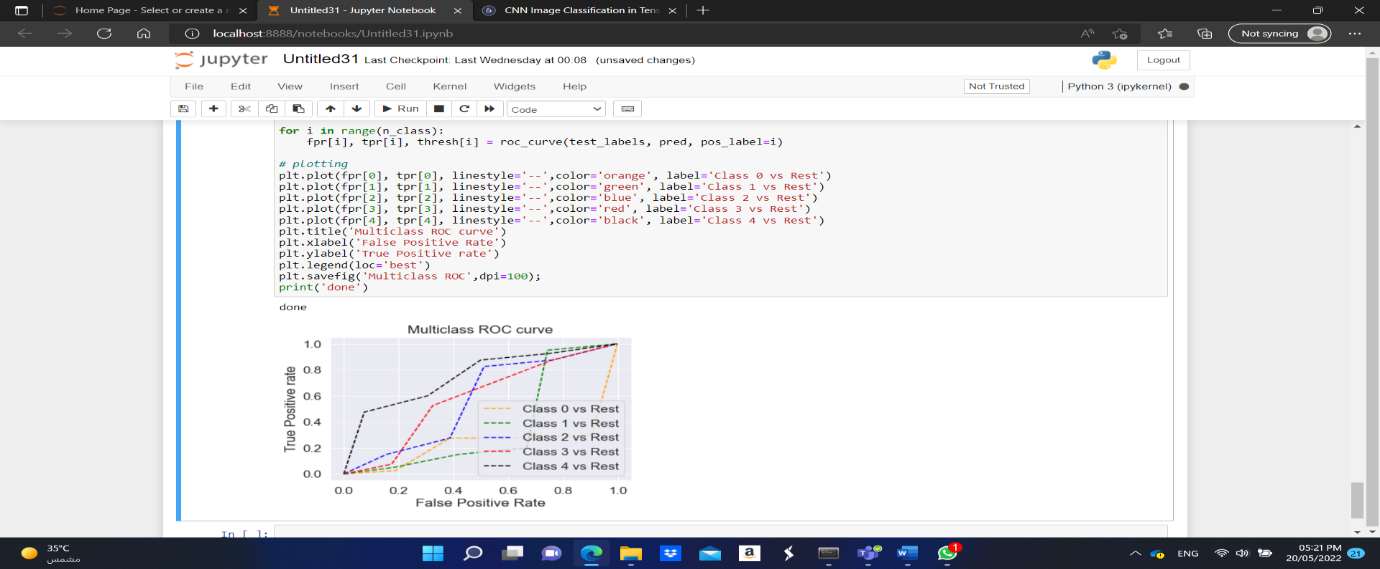
Description automatically generated

*The loss curve:*

Graphical user interface, application

Description automatically generated

*The ROC\_curve:*

**