REPORT

Introduction

I have concentrated on choosing an image processing project that makes use of convolutional neural networks (CNN) on the first day of my online internship. It is all about image classification and prediction .

DAY 01

**Image processing**

The modification and analysis of digital pictures fall under the umbrella of this important discipline. It may be used for a wide variety of things, including object identification and autonomous cars as well as medical imaging.

**Convolutional Neural Networks for Image Processing**

CNNs have demonstrated considerable potential in image processing jobs. They are particularly suited for applications like image classification, object identification, and image segmentation because they can automatically learn hierarchical representations from pictures.

*CNN Architectures*

* LeNet
* AlexNet
* VGGNet
* GoogLeNet
* ResNet

*Reference*

<https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5>

<https://towardsdatascience.com/various-types-of-convolutional-neural-network-8b00c9a08a1b>

**Project: Google Recaptcha**

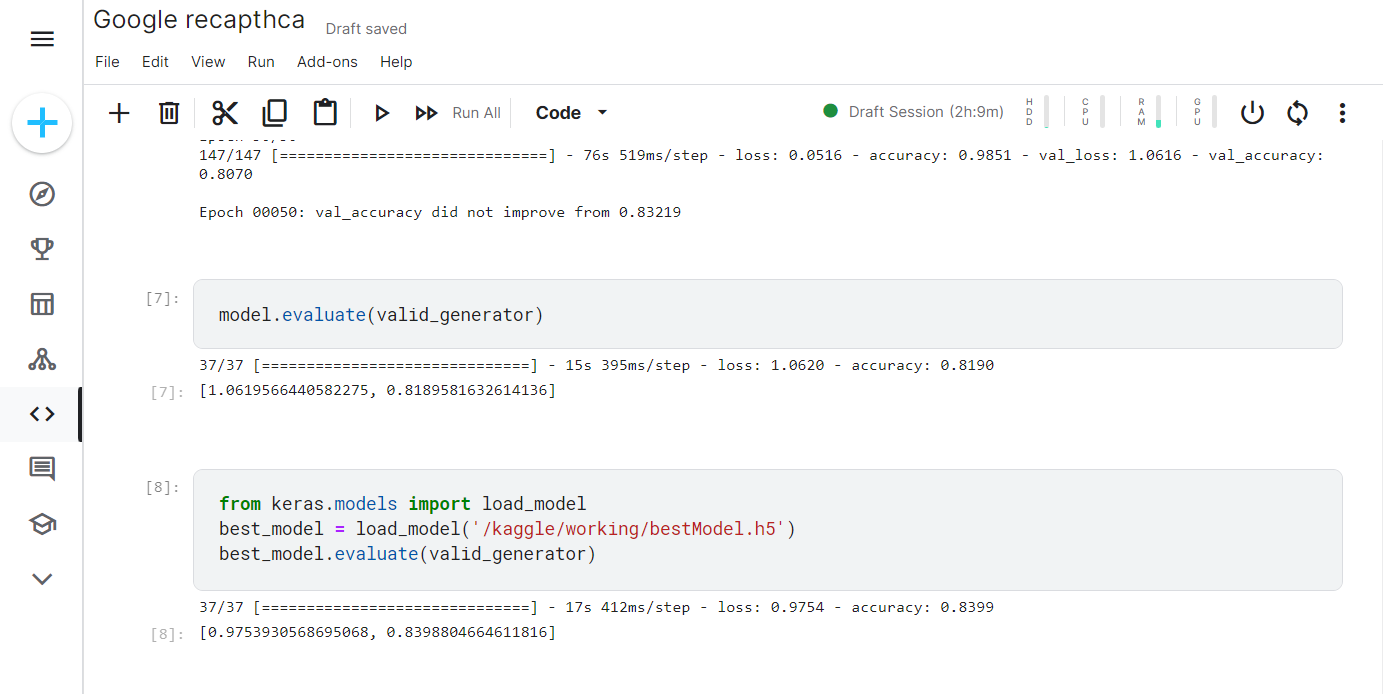
Almost **12000** images used in Google Recaptcha V2 collected by category more than 500 of which with manual markup for training object detection model such as YOLO. The dataset consists of **2** directories. The images directory contains **12** directories containing images in various categories such as: Bicycle, Bridge, Bus, Сar, Chimney, Сrosswalk, Hydrant, Motorcycle, Palm, Stair, Traffic light and Other.

*Reference*

<https://www.kaggle.com/datasets/mikhailma/test-dataset>

<https://www.kaggle.com/code/ahmedhossam666/google-recapthca>

<https://www.kaggle.com/code/csujith0210/google-recapthca/edit>



Day 02

**Selecting Datasets Name/category:**

We need to identify the datasets name like bus and bike .If dataset collected is having only two classes means it is binary datasets ,if it is more than two classes then it multiclass(categorical) problem .So I taken ships and boats.

**Collecting image datasets:**

For collecting image dataset we need minimum of 300 image ,so downloading more images in easy method we need to add extension to the browser like bulk image downloader.

**Data cleaning:**

After collecting images datasets we need to clean the image so accuracy for classification will be increased like removing texted images or other than selected class .

**Uploading of Datasets :**

Uploading cleaned datasets to google drive datasets are classified into three types ->Train(200<)

->Test(50<)

->Validation(35<)

from google.colab import drive

drive.mount('/content/drive')

**Code:**

# Import libraries

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow import keras

from keras.layers import \*

from keras.models import \*

from keras.preprocessing import image

from keras.preprocessing.image import ImageDataGenerator

import os, shutil

import warnings

warnings.filterwarnings('ignore')

# Let's plot a few images

train\_path = "/content/drive/MyDrive/Datasets/Train"

validation\_path = "/content/drive/MyDrive/Datasets/Validation"

test\_path = "/content/drive/MyDrive/Datasets/Test"

image\_categories = os.listdir('/content/drive/MyDrive/Datasets/Train')

import keras.utils as image

def plot\_images(image\_categories):

# Create a figure

plt.figure(figsize=(12, 12))

for i, cat in enumerate(image\_categories):

# Load images for the ith category

image\_path = train\_path + '/' + cat

images\_in\_folder = os.listdir(image\_path)

first\_image\_of\_folder = images\_in\_folder[0]

first\_image\_path = image\_path + '/' + first\_image\_of\_folder

img = image.load\_img(first\_image\_path)

img\_arr = image.img\_to\_array(img)/255.0

# Create Subplot and plot the images

plt.subplot(4, 4, i+1)

plt.imshow(img\_arr)

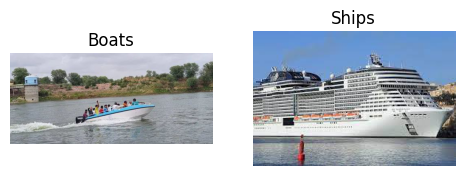
plt.title(cat)

plt.axis('off')

plt.show()

# Call the function

plot\_images(image\_categories)



# Creating Image Data Generator for train, validation and test set

# 1. Train Set

train\_gen = ImageDataGenerator(rescale = 1.0/255.0) # Normalise the data

train\_image\_generator = train\_gen.flow\_from\_directory(

train\_path,

target\_size=(150, 150),

batch\_size=32,

class\_mode='binary')

# 2. Validation Set

val\_gen = ImageDataGenerator(rescale = 1.0/255.0) # Normalise the data

val\_image\_generator = train\_gen.flow\_from\_directory(

validation\_path,

target\_size=(150, 150),

batch\_size=32,

class\_mode='binary')

# 3. Test Set

test\_gen = ImageDataGenerator(rescale = 1.0/255.0) # Normalise the data

test\_image\_generator = train\_gen.flow\_from\_directory(

test\_path,

target\_size=(150, 150),

batch\_size=32,

class\_mode='binary')

Found 546 images belonging to 2 classes.

Found 100 images belonging to 2 classes.

# Print the class encodings done by the generators

class\_map = dict([(v, k) for k, v in train\_image\_generator.class\_indices.items()])

print(class\_map)

{0: 'Boats', 1: 'Ships'}

# Build a custom sequential CNN model

model = Sequential() # model object

# Add Layers

model.add(Conv2D(filters=32, kernel\_size=3, strides=1, padding='same', activation='relu', input\_shape=[150, 150, 3]))

model.add(MaxPooling2D(2, ))

model.add(Conv2D(filters=64, kernel\_size=3, strides=1, padding='same', activation='relu'))

model.add(MaxPooling2D(2))

# Flatten the feature map

model.add(Flatten())

# Add the fully connected layers

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

model.add(Dropout(0.25))

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

model.add(Dense(2, activation='sigmoid'))

# print the model summary

model.summary()

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_4 (Conv2D) (None, 150, 150, 32) 896

max\_pooling2d\_4 (MaxPooling (None, 75, 75, 32) 0

2D)

conv2d\_5 (Conv2D) (None, 75, 75, 64) 18496

max\_pooling2d\_5 (MaxPooling (None, 37, 37, 64) 0

2D)

flatten\_2 (Flatten) (None, 87616) 0

dense\_6 (Dense) (None, 128) 11214976

dropout\_2 (Dropout) (None, 128) 0

dense\_7 (Dense) (None, 128) 16512

dense\_8 (Dense) (None, 1) 129

=================================================================

Total params: 11,251,009

Trainable params: 11,251,009

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Compile and fit the model

early\_stopping = keras.callbacks.EarlyStopping(patience=5) # Set up callbacks

model.compile(optimizer='Adam', loss='binary\_crossentropy', metrics='accuracy')

hist = model.fit(train\_image\_generator,

epochs=100,

verbose=1,

validation\_data=val\_image\_generator,

steps\_per\_epoch = 15000//32,

validation\_steps = 3000//32,

callbacks=early\_stopping)

Epoch 20/20

18/18 [==============================] - ETA: 0s - loss: 0.0180 - accuracy: 0.9963

WARNING:tensorflow:Early stopping conditioned on metric `val\_loss` which is not available. Available metrics are: loss,accuracy

18/18 [==============================] - 29s 2s/step - loss: 0.0180 - accuracy: 0.9963

# Plot the error and accuracy

h = hist.history

plt.style.use('ggplot')

plt.figure(figsize=(10, 5))

plt.plot(h['loss'], c='red', label='Training Loss')

plt.plot(h['val\_loss'], c='red', linestyle='--', label='Validation Loss')

plt.plot(h['accuracy'], c='blue', label='Training Accuracy')

plt.plot(h['val\_accuracy'], c='blue', linestyle='--', label='Validation Accuracy')

plt.xlabel("Number of Epochs")

plt.legend(loc='best')

plt.show()



# Testing the Model

test\_image\_path = '/content/drive/MyDrive/Datasets/Test/Boats/images39.jpg'

def generate\_predictions(test\_image\_path, actual\_label):

    # 1. Load and preprocess the image

    test\_img = image.load\_img(test\_image\_path, target\_size=(150, 150))

    test\_img\_arr = image.img\_to\_array(test\_img)/255.0

    test\_img\_input = test\_img\_arr.reshape((1, test\_img\_arr.shape[0], test\_img\_arr.shape[1], test\_img\_arr.shape[2]))

    # 2. Make Predictions

    predicted\_label = np.argmax(model.predict(test\_img\_input))

    predicted\_vegetable = class\_map[predicted\_label]

    plt.figure(figsize=(4, 4))

    plt.imshow(test\_img\_arr)

    plt.title("Predicted Label: {}, Actual Label: {}".format(predicted\_vegetable, actual\_label))

    plt.grid()

    plt.axis('off')

    plt.show()

# call the function

generate\_predictions(test\_image\_path, actual\_label='Boats')



Day 03

**Execution of given datasets:**

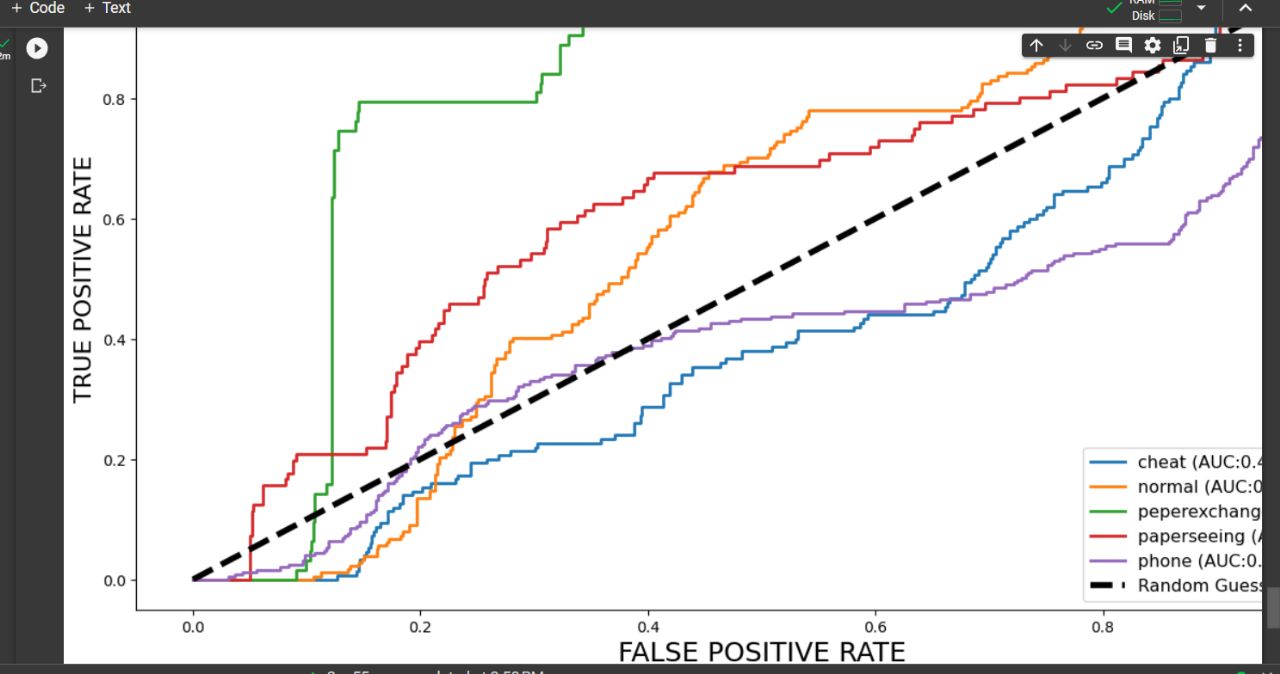
Datasets are given with cleaned datasets we needed to execute and run the output .The given datasets named ‘Examdataset’ ,it consists of train ,validation validation2 datasets .

**EXAMDATASET :**

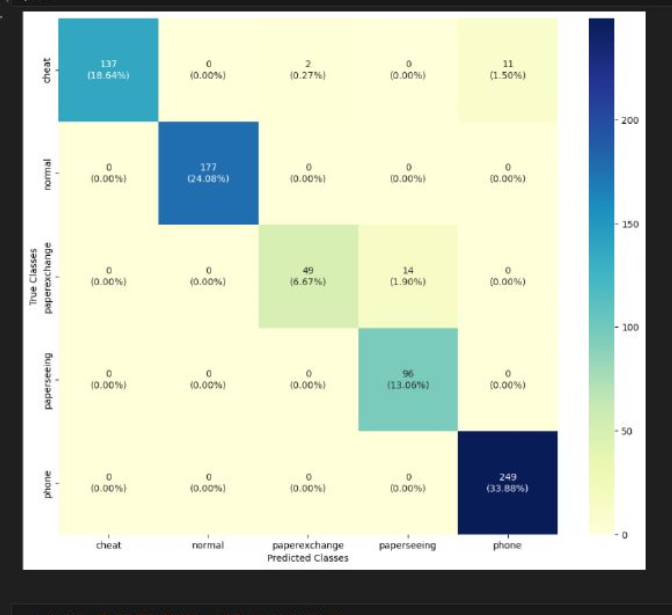
Running the code(resnet50) using vs code for reducing number of errors .

In train datasets it has 5 classes(categorical) Cheat,Normal,Paperexchanging,Paperseeing,Phone

**Screenshots:**



**Confusion matrix**

****

Day 04

confusion matrix

download anaconda

block size

epoch

accuricy

learn range

Day 05-07

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model or Algorithm | Epoch | Accuracy | precision | Recall | F-1 score | Reason for epoch stopping | Learning Rate | Batch  Size |
| VGG16 | 100 | 0.908648 | 0.902249 | 0.890445 | 0.891675 | Epochs run successfully | 0.01 | 64 |
| Sequential CNN | 100 | 0.910359 | 0.831537 | 0.889383 | 0.95844 | Epochs run successfully | 0.01 | 64 |
| ResNet51 | 100 |  |  |  |  | Key error: failed to format | 0.01 | 64 |

