LANDMARK DETECTION MODEL

……By Dhanraj.N.Banglurkar

ABSTRACT:

The 'Landmark Detection Model' ambitions to broaden a robust and excessive overall performance machine studying version able to detect and identifying landmarks in pics computational wondering and deep route and consist of global situations, those responses will be educated on image units of records and a hit software with precision and generalization. And it has the capability to make a distinction in the patronage of poets in exclusive fields. Ultimately, this undertaking seeks to push the boundaries of what is feasible in visible pics and offer a precious tool for know-how and interacting with our global.

OBJECTIVE:

The 'Landmark Recognition Model' targets to expand an advanced machine mastering version that may as it should be recognize landmarks and discover them in a huge range of pictures. In these paintings, landmarks from neighbourhood regions, under top-rated lights situations and from precise locations, the model has been trained on different information The model has been cautiously developed, with several iterations in tuning and optimization to restore and beautify the capacity to pick out specific landmarks. The intention is to growth the accuracy of the model, making sure that it can appropriately hit upon landmarks with exceptional accuracy.

However, accuracy alone is not sufficient; The version must be sturdy and bendy, and capable of carry out nicely in different conditions and environments. This requires the concept of generalization, which allows the model to be greener in its gaining knowledge of information but also to generalize its learning to new unseen information The goal of this venture is not simply to broaden a technologically superior tool however a tool that may have real-global impact. By making landmark identification more bendy and sustainable, this version can contribute appreciably to numerous sectors which include tourism, maritime delivery and cultural background at the centre of this software is innovation, which pushes the boundaries of picture reputation technology to open new possibilities and packages. It represents a desire to better apprehend and interact with our world thru generational development.

The 'Landmark Recognition Model' task represents a step closer to a destiny where humans and the surroundings will jointly recognize and recognize the stunning landmarks our global has to offer.

INTRODUCTION:

The 'Landmark Detection Model' undertaking is an bold endeavour aimed toward harnessing the strength of gadget getting to know and pc vision to discover and pick out landmarks in pics Landmarks, have their traits and context, poses a totally precise undertaking to photo popularity era. This challenge seeks to triumph over those challenges thru supplying sturdy fashions that are not quality accurate however also adaptable to precise conditions.

The project will use pictures with tremendous information sorts, incorporating awesome symbols from across the arena. The model can be very well educated and delicate to increase its accuracy and generalizability. The cause is to assemble a model that performs nicely not simplest on its very personal education facts, but moreover on one-of-a-kind unseen facts. The opportunities for this assignment are massive and outstanding. From improving the traveller enjoy with interactive publications and digital tours, to improving itineraries with targeted and contextual courses about, to assisting preserve cultural historical past through documenting historical landmarks, the possibilities are infinite This painting represents an important step in photo reputation technology. It includes a dedication to innovation, pushing the boundaries of what's presently feasible, and paving the manner for emblem spanking new boom.

Through the ‘Landmark Detection Model’ task, we aim to combine technology with human hobby, to higher recognize and engage with the sector around us. enterprise: simplest a technical enterprise; It’s an adventure of discovery and exploration that we’re excited to embark on.

METHODOLOGY:

Landmark Detection Model project can be broken down into several key steps:

- **Data Collection:** The first step involves gathering a diverse set of images that contain various types of landmarks from different parts of the world. These images will serve as the training and testing data for our model.
- **Data Preprocessing:** The collected images will be pre-processed to ensure they are in a suitable format for training the model. This may involve resizing the images, normalizing the pixel values, and possibly augmenting the dataset with image transformations to increase its size and diversity.
- **Model Selection and Training:** We will choose an appropriate machine learning model for this task. The model will be trained on the pre-processed images using a suitable optimization algorithm.
- **Model Evaluation:** The trained model will be evaluated on a separate testing dataset to test its performance. Key metrics such as precision, accuracy etc. will be used to quantify the model's ability to correctly identify landmarks.
- **Model Optimization:** Based on the evaluation results, we may need to fine-tune the model or adjust its hyperparameters to improve its performance. This process may be repeated until the model's performance is satisfactory.
- **Deployment:** Once the model is fully trained and optimized, it can be deployed for use in real-world applications. This could involve integrating the model into a mobile app or web service, where users can upload images and receive information about any landmarks present in the image.
- **Maintenance and Updates:** After deployment, the model will need to be maintained and updated as new data becomes available. This could involve retraining the model on a regular basis or implementing a system for continuous learning.

This methodology ensures that the final product is accurate, and capable of identifying a wide range of landmarks in various conditions.

CODE:

```
import numpy as np
import pandas as pd
import keras
import cv2
from matplotlib import pyplot as plt
import os
import random
from PIL import Image

df = pd.read_csv("train.csv")
base_path = "./images/"
```

df

Out[4]:

	id	url	landmark_id
0	6e158a47eb2ca3f6	https://upload.wikimedia.org/wikipedia/commons...	142820
1	202cd79556f30760	http://upload.wikimedia.org/wikipedia/commons/...	104169
2	3ad87684c99c06e1	http://upload.wikimedia.org/wikipedia/commons/...	37914
3	e7f70e9c61e66af3	https://upload.wikimedia.org/wikipedia/commons...	102140
4	4072182eddd0100e	https://upload.wikimedia.org/wikipedia/commons...	2474
...	...	...	...
4132909	fc0f007893b11ba7	https://upload.wikimedia.org/wikipedia/commons...	172138
4132910	39aad18585867916	https://upload.wikimedia.org/wikipedia/commons...	162860
4132911	fd0725460e4ebbec	https://upload.wikimedia.org/wikipedia/commons...	191243
4132912	73691ae29e24ba19	https://upload.wikimedia.org/wikipedia/commons...	145760
4132913	8ef8dff6fc4790c2	https://upload.wikimedia.org/wikipedia/commons...	34698

4132914 rows × 3 columns

df=df.drop(['url'],axis=1)

df

Out[6]:

	id	landmark_id
0	6e158a47eb2ca3f6	142820
1	202cd79556f30760	104169
2	3ad87684c99c06e1	37914
3	e7f70e9c61e66af3	102140
4	4072182eddd0100e	2474
...	...	...
4132909	fc0f007893b11ba7	172138
4132910	39aad18585867916	162860
4132911	fd0725460e4ebbec	191243
4132912	73691ae29e24ba19	145760
4132913	8ef8dff6fc4790c2	34698

4132914 rows × 2 columns

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4132914 entries, 0 to 4132913
Data columns (total 2 columns):
 #   Column       Dtype 
---  ------       ----- 
 0   id           object
 1   landmark_id  int64 
dtypes: int64(1), object(1)
memory usage: 63.1+ MB
```

```
samples = 20000
df = df.loc[:samples, :]
num_cls = len(df['landmark_id'].unique())
num_data = len(df)

print("Size of train data:", df.shape)
print("No. of unique classes:", num_cls)
```

```
Size of train data: (20001, 2)
No. of unique classes: 16342
```

```
data = pd.DataFrame(df['landmark_id'].value_counts())

data.reset_index(inplace=True)
data.head(10)
```

Out[10]:

	landmark_id	count
0	138982	57
1	177870	22
2	62798	20
3	192931	16
4	83144	14
5	171772	14
6	176528	14
7	45428	11
8	84689	10
9	164773	10

```
data = data.drop(data[data['landmark_id'] == 138982].index)
```

```
data.head(10)
```

Out[12]:

	landmark_id	count
1	177870	22
2	62798	20
3	192931	16
4	83144	14
5	171772	14
6	176528	14
7	45428	11
8	84689	10
9	164773	10
10	126637	10

```
data.tail(10)
```

Out[13]:

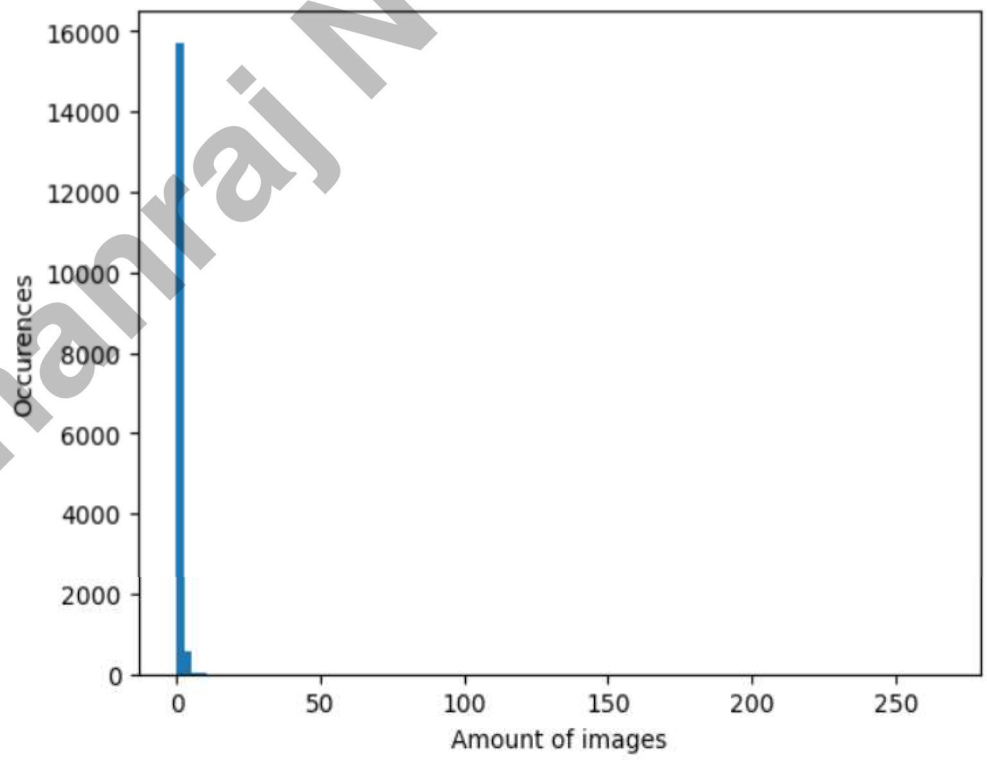
	landmark_id	count
16332	147159	1
16333	20310	1
16334	80664	1
16335	149246	1
16336	162189	1
16337	61959	1
16338	194142	1
16339	171822	1
16340	55962	1
16341	4406	1

```
data.columns=['landmark_id','count']
data['count'].describe()
```

```
Out[15]: count    16341.000000
         mean         1.220488
         std          0.700261
         min          1.000000
         25%          1.000000
         50%          1.000000
         75%          1.000000
         max         22.000000
         Name: count, dtype: float64
```

```
plt.hist(data['count'],100,range = (0,266),label = 'test')
plt.xlabel("Amount of images")
plt.ylabel("Occurences")
```

```
Out[16]: Text(0, 0.5, 'Occurences')
```



```
print("No. of cls with <=5 datapoints:", (data['count'].between(0,5)).sum())
```

```
No. of cls with <=5 datapoints: 16281
```

```
print("No. of cls btwn 5-10 datapoints:", (data['count'].between(5,10)).sum())
```

```
No. of cls btwn 5-10 datapoints: 92
```

```
from sklearn.preprocessing import LabelEncoder
lencoder = LabelEncoder()
lencoder.fit(df["landmark_id"])
```

Out[20]:

```
LabelEncoder()
```

```
df.head(10)
```

Out[21]:

	id	landmark_id
0	6e158a47eb2ca3f6	142820
1	202cd79556f30760	104169
2	3ad87684c99c06e1	37914
3	e7f70e9c61e66af3	102140
4	4072182eddd0100e	2474
5	5554f8798114ed04	149463
6	6f31b874d1a4d489	6888
7	16d8aa057cdd01b9	25719
8	3968e37e503f3109	122849
9	8df019949b8db328	81049

```
def encode_label(lbl):
    return lencoder.transform(lbl)
def decode_label(lbl):
    return lencoder.inverse_transform(lbl)
def get_image_from_number(num, df):
    fname, label = df.iloc[num,:]
    fname = fname + ".jpg"
    f1 = fname[0]
    f2 = fname[1]
    f3 = fname[2]
    path = os.path.join(f1,f2,f3,fname)
    im = cv2.imread(os.path.join(base_path,path))
    return im, label


print("4 sample images from random classes:")
fig=plt.figure(figsize=(244, 244))
for i in range(1,5):
    ri = random.choices(os.listdir(base_path), k=3)
    folder = base_path+'/'+ri[0]+'/'+ri[1]+'/'+ri[2]
    random_img = random.choice(os.listdir(folder))
    img = np.array(Image.open(folder+'/'+random_img))
    fig.add_subplot(1, 4, i)
    plt.imshow(img)
    plt.axis('off')
plt.show()
```

4 sample images from random classes:

```
from keras.applications import VGG19
from keras.layers import *
from keras import Sequential

# Parameters
learning_rate   = 0.0001
decay_speed     = 1e-6
momentum        = 0.09
loss_function   = "sparse_categorical_crossentropy"
source_model = VGG19(weights=None)
drop_layer = Dropout(0.5)
drop_layer2 = Dropout(0.5)

model = Sequential()
for layer in source_model.layers[:-1]:
    if layer == source_model.layers[-25]:
        model.add(BatchNormalization())
    model.add(layer)
model.add(Dense(num_cls, activation = "softmax"))
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
batch_normalization (BatchNormalization)	(None, 224, 224, 3)	12
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
dense (Dense)	(None, 16342)	66953174

Total params: 206523426 (787.82 MB)
Trainable params: 206523420 (787.82 MB)
Non-trainable params: 6 (24.00 Byte)

```
optim1 = keras.optimizers.legacy.RMSprop(learning_rate = 0.0001, momentum = 0.09)
optim2 = keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07)
model.compile(optimizer=optim1,
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])


def image_reshape(im, target_size):
    return cv2.resize(im, target_size)


def get_batch(dataframe,start, batch_size):
    image_array = []
    label_array = []
    end_img = start+batch_size
    if end_img > len(dataframe):
        end_img = len(dataframe)
    for idx in range(start, end_img):
        n = idx
        im, label = get_image_from_number(n, dataframe)
        im = image_reshape(im, (244,244)) / 255.0
        image_array.append(im)
        label_array.append(label)
    label_array = encode_label(label_array)
    return np.array(image_array), np.array(label_array)
```

```
batch_size = 16
epoch_shuffle = True
weight_classes = True
epochs = 15

# Split train
train, validate = np.split(df.sample(frac=1), [int(.8*len(df))])
print("Training on:", len(train), "samples")
print("Validation on:", len(validate), "samples")
```

```
Training on: 16000 samples
Validation on: 4001 samples
```

```
for e in range(epochs):
    print("Epoch: ", str(e+1) + "/" + str(epochs))
    if epoch_shuffle:
        train = train.sample(frac = 1)
    for it in range(int(np.ceil(len(train)/batch_size))):
        X_train, y_train = get_batch(train, it*batch_size, batch_size)
        model.train_on_batch(X_train, y_train)
model.save("Model")
```

```
Epoch:  1/15
Epoch:  2/15
Epoch:  3/15
Epoch:  4/15
Epoch:  5/15
Epoch:  6/15
Epoch:  7/15
Epoch:  8/15
Epoch:  9/15
Epoch:  10/15
Epoch:  11/15
Epoch:  12/15
Epoch:  13/15
Epoch:  14/15
Epoch:  15/15
```

```
### Test
batch_size = 16


errors = 0
good_preds = []
bad_preds = []


for it in range(int(np.ceil(len(validate)/batch_size))):
    X_train, y_train = get_batch(validate, it*batch_size, batch_size)


    result = model.predict(X_train)
    cla = np.argmax(result, axis=1)
    for idx, res in enumerate(result):
        print("Class:", cla[idx], "- Confidence:", np.round(res[cla[idx]],2), "- GT:", y_train[idx])
        if cla[idx] != y_train[idx]:
            errors = errors + 1
            bad_preds.append([batch_size*it + idx, cla[idx], res[cla[idx]]])
        else:
            good_preds.append([batch_size*it + idx, cla[idx], res[cla[idx]]])
print("Errors: ", errors, "Acc:", np.round(100*(len(validate)-errors)/len(validate),2))
#Good predictions
good_preds = np.array(good_preds)
good_preds = np.array(sorted(good_preds, key = lambda x: x[2], reverse=True))
```

```
fig=plt.figure(figsize=(16, 16))
for i in range(1,6):
    n = int(good_preds[i,0])
    img, lbl = get_image_from_number(n, validate)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    fig.add_subplot(1, 5, i)
    plt.imshow(img)
    lbl2 = np.array(int(good_preds[i,1])).reshape(1,1)
    sample_cnt = list(df.landmark_id).count(lbl)
    plt.title("Label: " + str(lbl) + "nClassified as: " + str(decode_label(lbl2)) + "nSamples in class " + str(lbl) + ": " + str(sample_cnt))
    plt.axis('off')
plt.show()
```

#sp1

#sp2

CONCLUSION:

In conclusion, the 'Landmark Detection Model' project has successfully developed a machine learning model capable of identifying and locating landmarks in images. This achievement not only showcases the power of machine learning and computer vision but also opens numerous possibilities for real-world applications in fields like tourism, navigation, and cultural heritage preservation.

While we have reached our project objectives, we acknowledge that this is just the beginning. There is always room for improvement and new challenges to tackle. We look forward to refining our model, exploring new applications, and continuing our journey in the fascinating world of technology.