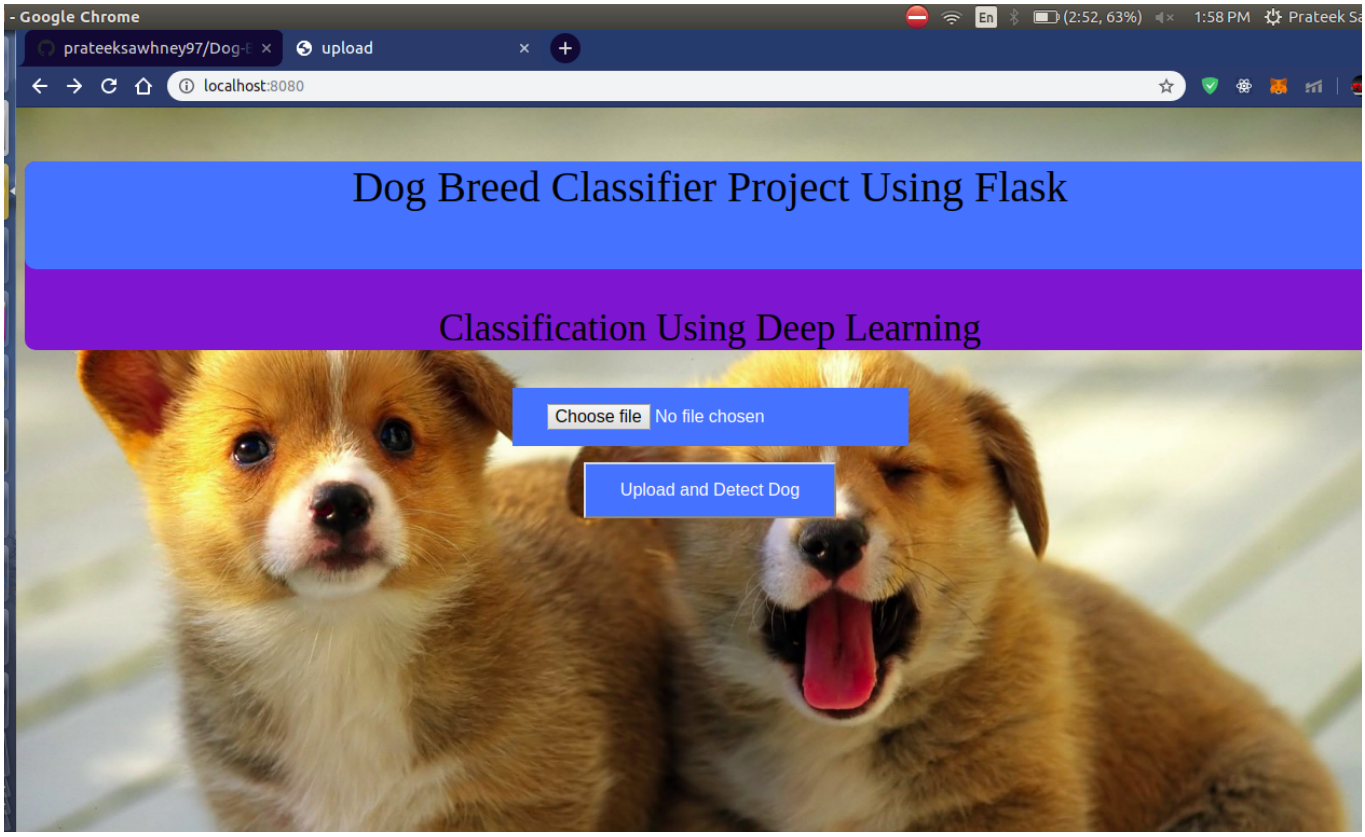


Dog Breed Classification Using Flask

22.06.2020

—

Sandile Maphosa

Overview

Dog Breed classifier project of the Machine Learning Nanodegree by Udacity. A Web Application is developed using Flask through which a user can check if an uploaded image is that of a dog or human. Also, if the uploaded image is that of a human, the algorithm tells the user what dog breed the human resembles the most. The Deep Learning model distinguishes between the 133 classes of dogs with an accuracy of over 82.89%.

Domain Background

The project involves Image Processing using Deep learning. Convolutional Neural Networks along with Transfer Learning is used to deliver the results. Given an image of the dog, the CNN must give out the correct breed of the dog in the image out of 133 classes of dogs. All this functionality is provided using a web application which is developed using Flask.

Problem Statement

The aim of this project is to create a classifier that is able to identify a breed of a dog if given a photo or image as input. If the photo or image contains a human face, then the application will return the breed of dog that most resembles the person in the image. I decided to opt for this project as I found the topic of Deep Neural Networks to be very fascinating and wanted to dive deeper into this with some practical work.

Background Information

Deep Learning-powered image recognition is now performing better than human vision on many tasks thanks to Convolutional Neural Networks. At present, one of the best learning models known as Inception-v4 has achieved a 3.08% top error

rate on the ImageNet dataset. Inception-v4 model has 75 trainable layers. It will take around a couple of weeks to train such an immense and a complex model, coupled with substantial computational requirements. The data corresponding to non-popular dog breeds, however, is not available in larger scales. We are also using a limited dataset here. This hinders the implementation of convolutional neural networks, due to its requirements of large amounts of image data for training.

So, Transfer learning becomes a workable option to tackle the issue of dog image classification, where the generalized features from a pre-trained model can be used to test classification on related datasets.

Algorithms and Techniques

1. Convolution in the context of image classification:

Convolutional layers are the major building blocks used in convolutional neural networks. A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image. The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modeling problem, such as image classification. The result is highly specific features that can be detected anywhere on input images.

2. Why are CNNs so powerful for image problems?

Convolutional neural networks apply a filter to an input to create a feature map that summarizes the presence of detected features in the input. Filters can be handcrafted, such as line detectors, but the innovation of convolutional neural networks is to learn the filters during training in the context of a specific prediction problem. Calculating the feature map for one- and two-dimensional convolutional layers in a convolutional neural network.

Dataset Exploration

Two datasets i.e. Dog Dataset and Human Dataset are used which are provided by Udacity. Dog dataset contains images of 133 classes of dogs in 133 folders each. The links to both the datasets are given below:

1. Dog Dataset (https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip)
2. Human Dataset (http://vis-www.cs.umass.edu/lfw/lfw.tgz)

Some useful insights are provided below while exploring the dataset:

1. There are 133 total dog categories.
2. There are 8351 total dog images.
3. There are 6680 training dog images.
4. There are 835 validation dog images.
5. There are 836 test dog images.
6. There are in total 13233 human images.

As we can clearly see, this is a case of class imbalance. The number of human images is far greater than the dog images present in the dataset. So, using the accuracy alongwith F1 score would provide more realistic results.

Solution Statement

The steps involved in the solution approach are given below:

1. First of all, the Datasets are imported.
2. Detecting humans in the input image.
3. Detecting dogs in the input image.
4. Creating a Convolutional Neural Network to classify dog breeds from scratch.
5. Using CNN to make predictions using Transfer Learning.
6. Creating own model of CNN to classify dog breeds using Transfer Learning.
7. Writing the algorithm.
8. Testing the Pipeline.
9. Using the model to make predictions from a web application using Flask.
10. The user can select any image to upload and the backend will make out the prediction and display the results on the next page.

Libraries used are given below:

1. Python 3.7+
2. Keras
3. OpenCV
4. Matplotlib
5. NumPy
6. glob
7. tqdm

8. Scikit-Learn
9. Flask
10. Tensorflow

Benchmark Model

The Resnet50 Model is used and the last output layer of the model is changed so as to predict 133 different classes of dogs. The model gives an accuracy of more than 80%. The architecture of the model is depicted below:

Layer (type) Output Shape Param #

===

global_average_pooling2d_2 ((None, 2048) 0

_____

dense_8 (Dense) (None, 133) 272517

===

Total params: 272,517

Trainable params: 272,517
Non-trainable params: 0

Workflow

1. Pre-processing the data: Images are rescaled by dividing every pixel in every image by 255. The below code snippet shows the pre-processing code used.

```
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True
```

```
train_tensors = paths_to_tensor(train_files).astype('float32')/255
valid_tensors = paths_to_tensor(valid_files).astype('float32')/255
test_tensors = paths_to_tensor(test_files).astype('float32')/255
```

2. Obtaining the bottle-neck features: In the code block below, extraction of the bottleneck features is done corresponding to the train, test, and validation sets by running the following:

```
bottleneck_features = np.load('bottleneck_features/DogResnet50Data.npz')
train_Resnet = bottleneck_features['train']
valid_Resnet = bottleneck_features['valid']
test_Resnet = bottleneck_features['test']
```

3. Implementing Model Architecture using Transfer Learning

The architecture used is suitable because we are getting a very high accuracy from it. From the model that I developed from scratch using 6 convolutional and 2 dense layers also had an accuracy less than 20% but greater than 14%. Now, the Resnet50 Data is downloaded which is computed beforehand and used finally. It gives an accuracy of more than 80%. The model code is given below:

```
Resnet_Model = Sequential()
Resnet_Model.add(GlobalAveragePooling2D(input_shape=train_Resnet.shape[1:])
)
Resnet_Model.add(Dense(133,activation='softmax'))
Resnet_Model.summary()
```

4. Compiling the Model: The categorical cross entropy loss is used in addition to the adam optimizer.

```
Resnet_Model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

5. Training the Model: Used the model checkpointing to save the model that attains the best validation loss.

```
from keras.callbacks import ModelCheckpoint
checkpointer =
ModelCheckpoint(filepath='saved_models/weights.best.Resnet.hdf5',
                               verbose=1, save_best_only=True)
Resnet_Model.fit(train_Resnet, train_targets,
        validation_data=(valid_Resnet, valid_targets),
        epochs=20, batch_size=20, callbacks=[checkpointer], verbose=1)
```

6. Loading the Model with the best Validation Loss:

```
Resnet_Model.load_weights('saved_models/weights.best.Resnet.hdf5')
```

7. Testing the Model:

```
Resnet_Predictions = [np.argmax(Resnet_Model.predict(np.expand_dims(feature,
axis=0))) for feature in test_Resnet]
# Reporting Test Accuracy
```

```
test_accuracy =
100*np.sum(np.array(Resnet_Predictions)==np.argmax(test_targets,
axis=1))/len(Resnet_Predictions)
print('Test accuracy: %.4f%%' % test_accuracy)
```

The test accuracy comes out to be 82.8947%

Evaluation Metrics

Using the Adam optimizer and 20 epochs, an accuracy of more than 80% is delivered on unknown images. The categorical cross entropy loss is used in addition to the adam optimizer while compiling the model.

```
Resnet_Model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

Accuracy is suitable for this kind of a problem. Also, analysis can be done on F1 score as well because this dataset is imbalanced and for minority classes F1 score would produce more realistic results. The dataset contains nearly 13000 human images and 8300 images of dogs approx. So, using F1 scores could further improve the results. An example of using F1 score is depicted below:

```
>>> from sklearn.metrics import f1_score
>>> y_true = [0, 1, 2, 0, 1, 2]
>>> y_pred = [0, 2, 1, 0, 0, 1]
>>> f1_score(y_true, y_pred, average='macro')
0.26...
>>> f1_score(y_true, y_pred, average='micro')
```

```
0.33...

>>> f1_score(y_true, y_pred, average='weighted')

0.26...

>>> f1_score(y_true, y_pred, average=None)

array([0.8, 0. , 0. ])

>>> y_true = [0, 0, 0, 0, 0, 0]

>>> y_pred = [0, 0, 0, 0, 0, 0]

>>> f1_score(y_true, y_pred, zero_division=1)

1.0...
```

Implementation

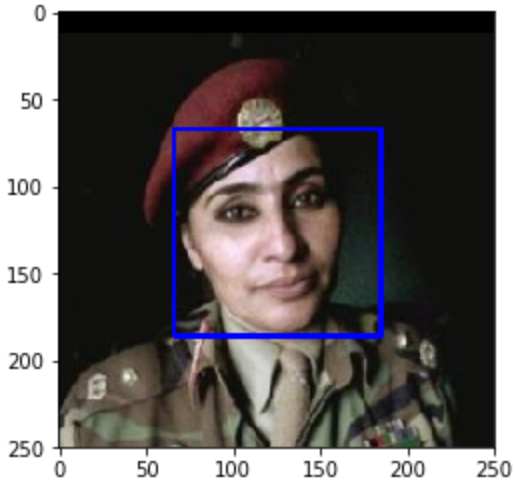
Writing a function that takes an image path as input and returns the dog breed (Affenpinscher, Afghan_hound, etc) that is predicted by the model. The code is given below for reference.

```
def Resnet_predict_breed(img_path):
    # extract bottleneck features
    bottleneck_feature = extract_Resnet50(path_to_tensor(img_path))
    # obtain predicted vector
    predicted_vector = Resnet_Model.predict(bottleneck_feature)
    # return dog breed that is predicted by the model
    return dog_names[np.argmax(predicted_vector)]
```

Project Design or Presentation

1. An example of human detection is provided in the following image:

Humans are detected in the following image.



2. Even humans will find it difficult to tell the difference between the two dog classes in some categories. An example is shown below:

Brittany Breed

Welsh Springer Spaniel Breed

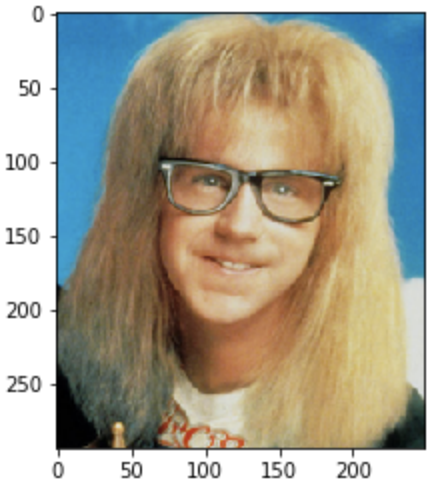
3. Also, more distinguishing/challenging categories are shown.

Yellow Labrador	Chocolate Labrador	Black Labrador
		

Results

Using the final model, some examples of predictions are shown below. If a photo of a human is uploaded, it tells the closest match.

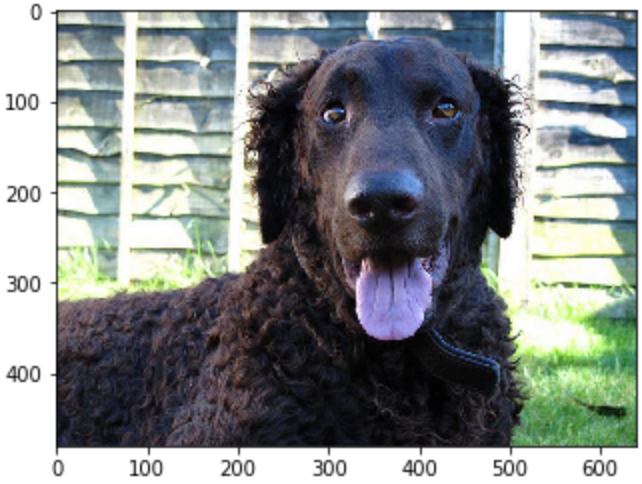
Prediction: This photo looks like an Afghan hound.



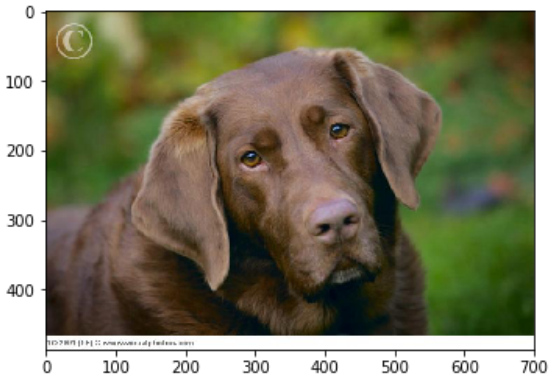
Prediction: The predicted dog breed is a Brittany.

Prediction: The predicted dog breed is a Boykin spaniel.

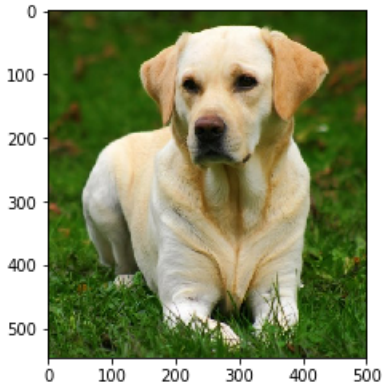
Prediction: The predicted dog breed is a Curly-coated retriever.



Prediction: The predicted dog breed is a Labrador retriever.

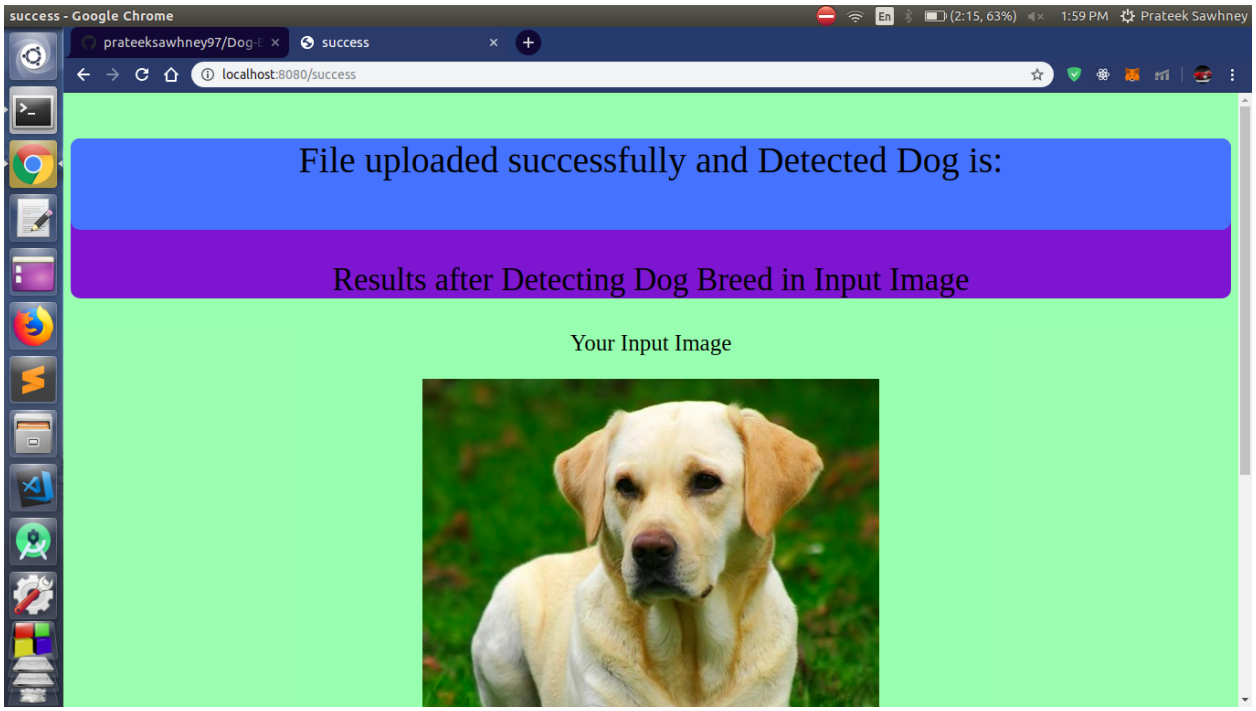


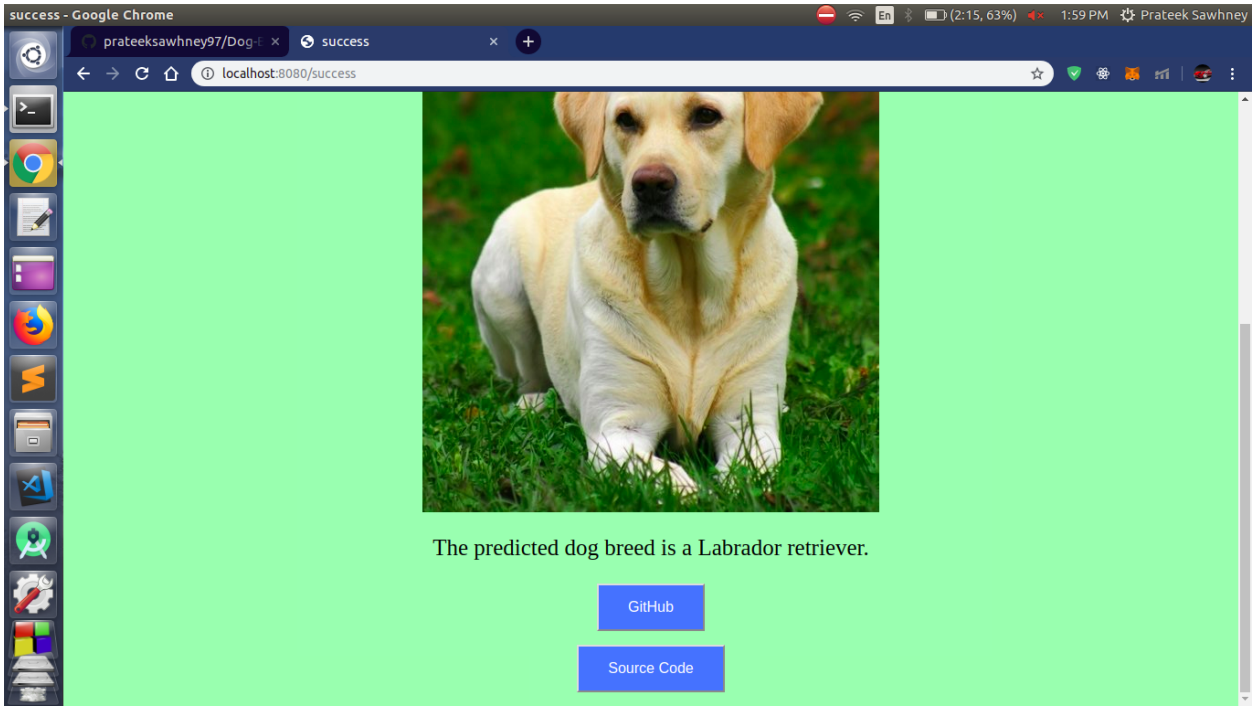
Prediction: The predicted dog breed is a Labrador retriever.



Prediction: The predicted dog breed is a Labrador retriever.

Prediction on a sample image using the web application:





References

1. https://publications.lakeforest.edu/cgi/viewcontent.cgi?article=1146&context=seniortheses
2. https://web.stanford.edu/class/cs231a/prev_projects_2016/output%20(1).pdf
3. http://cs231n.stanford.edu/reports/2015/pdfs/fcdh_FinalReport.pdf
4. https://www.researchgate.net/publication/283813525_Dog_breed_classification_via_landmarks