Machine Learning Engineer Nanodegree

Capstone Proposal

Ikechukwu Nigel Ogbuchi
January 19, 2020.

Proposal: Dog Breed Classifier

Domain Background

Convolutional neural networks (CNN) have been used to achieve amazing results in various applications like image classification, face recognition, scene labelling and document analysis. It comprises a deep learning architecture which is inspired by the structure of visual system. Convolutional networks are trainable multistage architectures with each stage consisting of multiple layers. The input and output of each stage are sets of arrays called as feature maps. In the case of a colored image, each feature map would be a 2D array containing a color channel of the input image, a 3D array for a video and a 1D array for an audio input. The output stage represents features extracted from all locations on the input. Each stage generally consists of a convolution layer, non-linearity and a pooling layer. A single or multiple fully-connected layers are present after several convolution and pooling layers.

Problem Statement

The way image classification works is this: A trained model is given an input and it is able to predict a class (for example “This is a dog”, “This is not a dog”) as an output. It can also give you the probability that the input belongs to a particular class (for example: “This is a dog. 90% confidence!”). The goal of the project was to create a pipeline that takes an image and detects whether a human or dog is present, predicting the breed for the dog or deciding what dog breed the human looks similar to. Sometimes we have cases of missing dog pets which could have been easily found if only we had a trained recognition system that can identify and alert humans if such a dog is seen on camera. This is the problem I am trying to solve with CNN.

Datasets And Inputs

The datasets I plan to use are those which can be obtained from the links below:

- The **dog dataset:** https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
  Folder is unzipped and placed in the Dog Classifier repo, at location "path/to/dog project/dogImages."
  The dogImages/ folder should contain 133 folders, each corresponding to a different dog breed. (Contains train, valid(validation) and test images)

- The **human dataset:** http://vis-www.cs.umass.edu/lfw/lfw.tgz
  Folder is unzipped and placed in the Dog Classifier repo, at location "path/to/dog-project/lfw".

- The **Python Packages** installed include: "load_files" from sklearn.datasets, "np.utils" from keras.utils package, numpy and glob.

After downloading the data, I will load the images into numpy arrays using the codes:

human_files = np.array(glob("/data/lfw//*"))*
dog_files = np.array(glob("/data/dog_images//*/*"))*

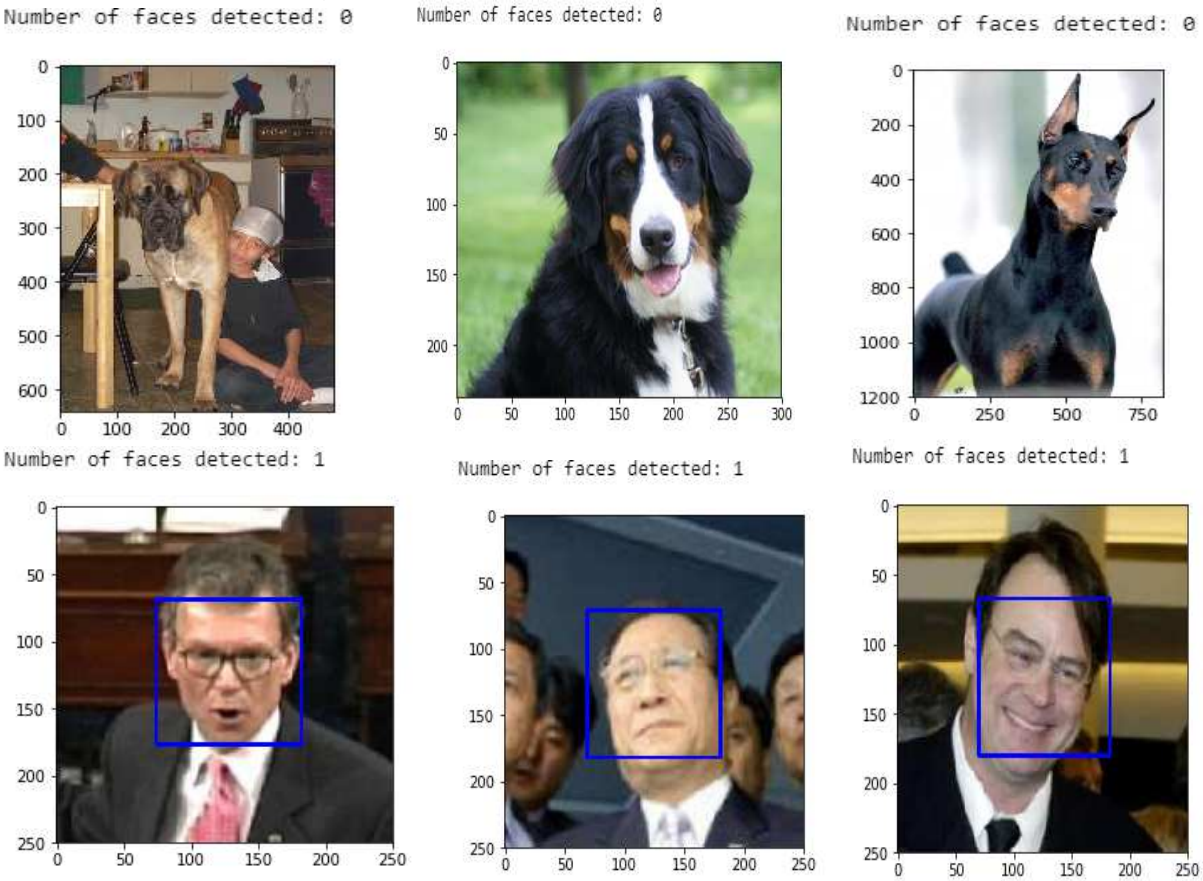
A brief analysis of the data given shows that there are 8351 total dog images with 133 dog categories.

For data in the the dog image, we will create these variables for our model:

- *train_files, valid_files, test_files* — numpy arrays containing file paths to images
- *train_targets, valid_targets, test_targets* — numpy arrays containing one hot-encoded classification labels
- *dog_names* — list of string-valued dog breed names for translating labels

Analysis also shows there are 13233 total human images and will be used as face detector.
The data for both cases are **mostly evenly distributed**.

Here are sample dog and human images from the distribution:



Solution Statement

The goal of this project is to create a dog breed identification application using convolutional neural networks. The resulting algorithm could be used as part of a mobile or web app. This design can be applied to image classification problem.

The code will accept any user-supplied image as input. If a dog is detected in the image, it will provide an estimate of the dog's breed. If a human is detected, it will provide an estimate of the dog breed that is most resembling.

Benchmark Model

I will use AlexNet and ResNet50 transfer learning models as benchmarks.

Evaluation Metrics

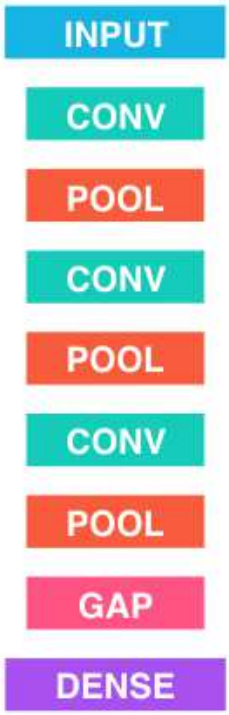
The evaluation metrics of this problem will be the Accuracy Score for the designed algorithm and the model that give us the lowest validation loss.

Project Design

First we will import the dataset using "load_files" function from scikit-learn library. And get a good summary of the data with feature names.
Next, we use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images, after that we use a pre-trained ResNet-50 model to detect dogs in images. This would involve preprocessing of the data to 4D arrays. I will then proceed to create a CNN from scratch just like the one shown in the image below.
The basic idea for this model's architecture is to use the convolutional layers from a pre-trained model to extract the features from the input image. These features are distilled down to a single vector using a Global Average Pooling layer. This vector is passed to a fully-connected output layer that calculates the probability for each dog breed. These models use convolutional neural networks to extract the image features and they can be used in this new model by transferring what the pre-trained models have already learned. The following figure shows a basic summary of the Dog breed classification model.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 223, 223, 16)	208
max_pooling2d_1 (MaxPooling2	(None, 111, 111, 16)	0
conv2d_2 (Conv2D)	(None, 110, 110, 32)	2080
max_pooling2d_2 (MaxPooling2	(None, 55, 55, 32)	0
conv2d_3 (Conv2D)	(None, 54, 54, 64)	8256
max_pooling2d_3 (MaxPooling2	(None, 27, 27, 64)	0
global_average_pooling2d_1 (	(None, 64)	0
dense_1 (Dense)	(None, 133)	8645

Total params: 19,189.0
Trainable params: 19,189.0
Non-trainable params: 0.0



After that, we compile the model and use it to classify dog breeds, then use a pre-trained VGG-16 model to extract key features while adding a global average pooling layer and a fully connected layer. After compiling and training the model, we will load the model with the best validation loss. We can now test the model and use it to make predictions about the class of a given input.