Machine Learning Engineer Nanodegree - Capstone Proposal

Xin Tao

Domain Background

Image classification refers to the task of identifying or predicting the category of an image based on its visual content. Like other classification problems, it can be a binary, multi-class or multi-label classification. For example, given an image, an example of binary classification is to identify whether the image contains a dog or not. Among all the known breeds of dogs, classifying the breed of a dog is a multi-class classification problem. If a picture has multiple objects, detecting what objects it has is a multi-label task.

Convolutional Neural Network (CNN) is a type of deep learning neural network, which has brought breakthroughs in image classification and computer vision. CNN has been widely proven to have high performance in computer vision related tasks. Images have high dimensionalities as each pixel is considered a feature. As the number of layers grows, the total number of parameters in the neural network explodes rapidly, which makes it impossible to train a model with regular computational power. CNN can effectively reduce the number of parameters without compromising the quality of the model. The dimensions are reduced in CNN by a sliding window (called filter or kernel) with a size less than input image[1]. As the window sliding across the whole image matrix, it extracts the features such as edges and colors from the input image but with a reduction in its dimensionality. As the network becomes deep, it extracts features from low level to high level. For the above reason, CNN is very suitable for image classification problems.

Problem Statement

In this project, I will be working on the "CNN Project: Dog Breed Classifier " [2] provided by Udacity. This is a classic multi-class image classification problem. The goal of this project is to develop an app that will identify an estimate of the canine's breed, given an image of a dog. If supplied an image of a human, the code will identify the resembling dog breed. It might sound easy for humans to identify a dog breed. But what are the breeds of the images below? Give it a try [3]!

Belgian Malinois or German Shepherd Dog?

Boston Terrier or. French Bulldog?

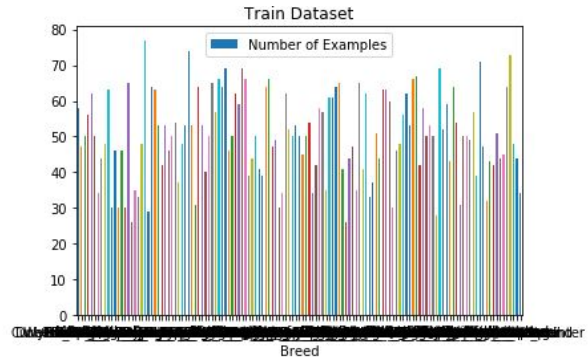
See, even for humans, we can't guarantee we have 100% accuracy of identifying the dog's breed. Some dogs of different breeds look very similar just by their appearance. DNA test is the most accurate way to identify the breed. A professional can have an accuracy above 90%. For regular people, like me, the accuracy can range from 10%-80% depending on knowledge about dogs.

Given the above background information, to be successful in this project, I must build a machine learning model that can beat average human accuracy of identifying dog's breed. Specifically Convolutional Neural Network (CNN) has been proven to achieve very high performance on image classification tasks. I will use variant of CNN models to solve two problems:

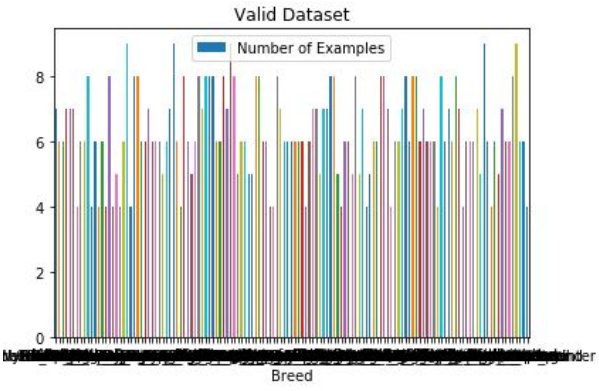
1) Tell if an image is a human or a dog. If it is a human, tells the resembling dog breed
2) If the image is a dog image, tells an estimate of the dog's breed

Datasets and Inputs

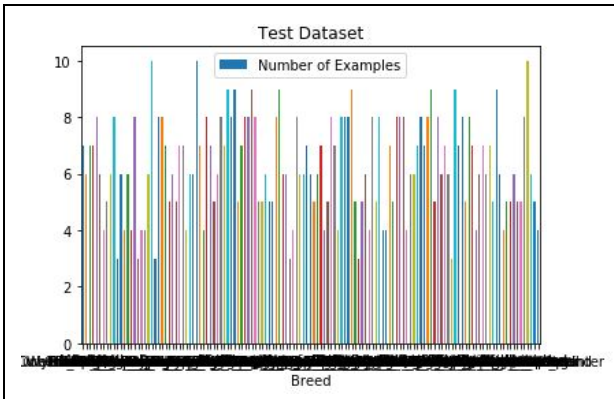
The datasets are provided by Udacity, including two different folders containing human dataset and dog dataset. The dog datasets are further split into train, valid and test datasets for machine learning tasks. There are 13233 total human images and 8351 total dog images. Among the dog images, the train/valid/test ratio is 6680/836/836. There are 133 breeds in each dataset stored in subfolders. It needs to point out that the data is unbalanced because some breeds have more images than others. A distribution of the dog's breeds in train/valid/test dataset is shown below.



	Number of Examples
count	133.000000
mean	50.225564
std	11.863885
min	26.000000
25%	42.000000
50%	50.000000
75%	61.000000
max	77.000000



	Number of Examples
count	133.000000
mean	6.278195
std	1.350384
min	4.000000
25%	6.000000
50%	6.000000
75%	7.000000
max	9.000000



	Number of Examples
count	133.000000
mean	6.285714
std	1.712571
min	3.000000
25%	5.000000
50%	6.000000
75%	8.000000
max	10.000000

The input data types are all images, which will be later converted to tensors for training to feed to the model. The images have different size, color, resolutions, and angles, which need data preprocessing. The use of the input datasets is very appropriate for CNN image classification because they are raw images with 3 channels, perfect to feed into VGG and ResNet.

Solution Statement

The solution will have two steps. First I will detect human faces in images using OpenCV's implementation of **Haar feature-based cascade classifiers** [4]. Second, I will use a pre-trained **VGG-16** model to detect dogs images. Third, I will develop the CNN model to predict the dog's breed both from scratch and from transfer learning.

Particularly, The CNN model from scratch will be built using a modified ResNet CNN Architecture, with parameters tuned to be appropriate to the input datasets. Deep Residual Network (ResNet) has been one of the most groundbreaking algorithms in Deep Learning community in the past few years, especially in the field of object detection and face recognition. ResNet makes it possible to train hundreds or even thousands of layers with high compelling performance. [5]

For the transfer learning model, I will use the most recent pretrained **ResNet152** Architecture. I will modify the FC layer parameters to match the dimension of our case. ResNet152 is suitable for our problem because it has the deepest layers in all ResNET architecture so far. When the layer got deeper, it can extract more relevant features (edges, shapes, colors, etc) and thus can make a better generalization/prediction. Though it is deep, each layer is relatively simple. ResNET152 is pre-trained on the ImageNet with a depth of 152 layers -- 8x deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set [6]. ResNet152 makes the solution qualifable (parameter to tune, loss function to use), measureable (evaluation method to use) and replicable (reliable result).

Benchmark Model

The CNN model created from scratch is a good benchmark model. It will be trained on the datasets provided with limited available images (8351 total dog images). A random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%. For the model created

from scratch, I will use 10% as a success criteria. Later the pre-trained transfer model, built up on a lot more training data, should perform much better than the benchmark model.

Evaluation Metrics

I will primarily use **test accuracy** to evaluate the model. A good model will achieve a test accuracy above 60%. Also the Transfer model which trained based on larger amounts of data should outperform the benchmark model (created from scratch). The accuracy is calculated by the number of correctly labeled images divided by the total number of images in the test set.

Apart from test accuracy, another metric to look at is the **Cross-entropy loss**. The Cross-entropy loss combines LogSoftmax and NLLLoss in one single class [7], which is very reliable for multi-class problems, especially when you have an unbalanced training set. The loss increases as the predicted probability diverges from the actual label. The loss will not directly tell how many images are predicted correctly, but it will be very useful in the training stage. If we see the loss value is small and continue decreasing for both train and valid dataset, that will be a good sign that the model will succeed.

Project Design

The project Design is provided by Udacity [8] with the steps listed below:
Step 0: Import Datasets: download the required human and dog datasets
Step 1: Detect Humans: use OpenCV's implementation to detect human faces in images
Step 2: Detect Dogs: use a pre-trained VGG-16 model to detect dogs in images
Step 3: Create a CNN to Classify Dog Breeds (from Scratch)
Step 4: Create a CNN to Classify Dog Breeds (using Transfer Learning)
Step 5: Write the Algorithm: if dog is detected, return the predicted breed; if a human is detected, return the resembling dog breed, if neither is detected, output an error
Step 6: Test Your Algorithm on sample images.

References

[1] Why are Convolutional Neural Networks good for image classification?
https://medium.com/datadriveninvestor/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8
[2] CNN Project: Dog Breed Classifier
https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification
[3] Can You Tell These Dog Breed Look-Alikes Apart?
https://www.akc.org/expert-advice/lifestyle/dog-breed-look-alikes/
[4] Face Detection using Haar Cascades
http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html
[5] An Overview of ResNet and its Variants.
https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035
[6] Deep Residual Learning for Image Recognition. https://arxiv.org/abs/1512.03385
[7] Cross-entropy loss https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html
[8] Git Repo https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification