Capstone Project **Maanav Shah**
Nanodegree **5th May 2020**

Convolutional Neural Networks (CNN) Project
Dog Breed Identification

Project Overview

In the last decade, ML becomes more popular thanks to very powerful computers that can handle to process lots of data in a reasonable amount of time. The machine learning concept was introduced by Arthur Samuel in 1959 so it is not new, but today we can use lots of its potential. One more reason for this is that today there is a lot of digitized data that we need to successfully implement good ML models.

The Dog breed classifier is a well-known problem in Machine Learning. The problem is to identify a breed of dog. The input can be a dog image or an image of a human, and we should be able to predict the breed of the dog or if that human was a dog, which breeds it would belong. The idea is to build a pipeline that can process real-world user-supplied images and identify an estimate of the canine's breed. This is a multi-class classification problem where we can use supervised machine learning to solve this problem. After completing this model, I am planning to build a web app using Amazon SageMaker where users can input an image and obtain prediction from this model. This project gives me an opportunity to build and deploy Machine Learning models, so I have chosen this as my capstone project.

One another reason to build this project is to create an android application that will able to tell which breed does this dog belongs to, using the real-time camera.

Dog breed identification problem is well known in the ML community. We can find it on Kaggle:
https://www.kaggle.com/c/dog-breed-identification/overview/description

Problem Statement

The goal of the project is to build a machine learning model that can be used within a web app to process real-world, user-supplied images. The algorithm has to perform two tasks:

1. Dogface detector

   Given an image of a dog, the algorithm will identify an estimate of the canine's breed.

2. Human face detector

    Given an image of a human, the code will identify the resembling dog breed.

The bigger picture is that we take a picture with a phone and the app tells us what dog breed is on the picture. Additionally, we want that app to tell us to what dog breed a human is most look alike. We want to have an answer to a few questions here. The main question is What dog breed is on the picture? The second question is: How would you look if you were a dog? To answer these two questions we first have to answer the question: Is on the picture human or a dog?

This is a supervised learning problem and because we have our dog images divided into breed classes we will use classification predictive modeling more precisely multi-class predictive model.

Evaluation Metrics

The data is split into training, testing, and validation dataset. The model is trained using the training dataset. We use the testing data to predict the performance of the model on unseen data. We will use accuracy as a metric to evaluate our model on test data.

Accuracy=Number of items correctly classified/ All classified items

Also, during model training, we compare the test data prediction with the validation dataset and calculate Multi-class log loss to find the best performing model. Log loss takes into account of the uncertainty of prediction based on how much it varies from the actual label and this will help in evaluating the model.

The problem we try to solve is a classification problem. Because our data is an unbalanced, simple accuracy score is not very good here. On Kaggle they use multi-class log loss metrics to evaluate models. I will also use this metric so I can compare it to results on Kaggle. I will also use F1 score testing because it considers precision and recall and it is easier for me to understand results.

We calculate F1 with the formula:

F1 = 2 * (precision * recall) / (precision + recall)

Our data is already divided into training, validation, and test partitions so we can now use our train data to make a benchmark model using Convolutional Neural Networks. After creating a model we will test it with test data. When we get accuracy over 10% we will proceed on building a new model using transfer learning. With transfer learning, we can build our model with fewer

data to give us a better result. We will use the same training data as before. We will then test our model with the same test data as before but know we expect our accuracy to be over 60%. Then we can try to experiment with different model parameters to get better results. We will use f1 score and log loss to evaluate our models.

Data Exploration

For this project, the input format must be of image type, because we want to input an image and identify the breed of the dog. The dataset has pictures of dogs and humans.

Dog images dataset: The dog image dataset has 8351 total images that are sorted into the train (6,680 Images), test (836 Images), and valid (835 Images) directories. Each of these directories (train, test, valid) has 133 folders corresponding to dog breeds. The images are of different sizes and different backgrounds, some images are not full-sized. The data is not balanced because the number of images provided for each breed varies. Few have 4 images while some have 8 images.

Human images dataset: The human dataset contains 13233 total human images which are sorted by names of humans (5750 folders). All images are of size 250x250. Images have different backgrounds and different angles. The data is not balanced because we have 1 image for some people and many images for some.

Algorithms and techniques:

For performing this multiclass classification, we can use a Convolutional Neural Network to solve the problem. A Convolutional Neural Network (CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The solution involves three steps. First, to detect human images, we can use existing algorithms like OpenCV's implementation of Haar feature-based cascade classifiers. Second, to detect dog-images we will use a pre-trained VGG16 model. Finally, after the image is identified as dog/human, we can pass this image to a CNN model which will process the image and predict the breed that matches the best out of 133 breeds.

Benchmark

The CNN model created from scratch must have an accuracy of at least 10%. This can confirm that the model is working because a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%.

Data Preprocessing

All the images are resized to 224*224, then normalization is applied to all images (train, valid, and test datasets). For the training data, Image augmentation is done to reduce overfitting. The train data images are randomly rotated and a random horizontal flip is applied. Finally, all the images are converted into tensor before passing into the model.

Implementation

I have built a CNN model from scratch to solve the problem. The model has 3 convolutional layers. All convolutional layers have a kernel size of 3 and stride 1. The first Conv layer (conv1) takes the 224*224 input image and the final Conv layer (conv3) produces an output size of 128. The ReLU activation function is used here. The pooling layer of (2,2) is used which will reduce the input size by 2. We have two fully connected layers that finally produce 133-dimensional output. A dropout of 0.25 is added to avoid overfitting.

Refinement

The CNN created from scratch has an accuracy of 16%, Though it meets the benchmarking, the model can be significantly improved by using transfer learning. To create CNN with transfer learning, I have selected the Resnet101 architecture which is pre-trained on the ImageNet dataset, the architecture is 101 layers deep. The last convolutional output of Resnet101 is fed as input to our model. We only need to add a fully connected layer to produce 133-dimensional output (one for each dog category). The model performed extremely well when compared to CNN from scratch. With just 5 epochs, the model got 81% accuracy.

Model Evaluation and Validation

Human Face detector: The human face detector function was created using OpenCV's implementation of Haar feature-based cascade classifiers. 98% of human faces were detected in the first 100 images of the human face dataset and 17% of human faces detected in the first 100 images of the dog dataset.

Dog Face detector: The dog detector function was created using a pre-trained VGG16 model. 100% of dog faces were detected in the first 100 images of the dog dataset and 1% of dog faces detected in the first 100 images of the human dataset.

CNN using transfer learning: The CNN model created using transfer learning with ResNet101 architecture was trained for 5 epochs, and the final model produced an accuracy of 81% on test data. The model correctly predicted breeds for 680 images out of 836 total images.

Accuracy on test data: 81% (680/836)

Justification

I think the model performance is better than expected. The model created using transfer learning has an accuracy of 81% compared to the CNN model created from scratch which had only 13% accuracy.

Improvement

The model can be improved by adding more training and test data, currently, the model is created using only 133 breeds of dog. Also, by performing more image augmentation, we can avoid overfitting and improve accuracy. I have tried only with ResNet 101 architecture for feature extraction, also the model can be improved using different architecture.

References

1. GitHub Repository: http://github.com/udacity/deep-learning-v2-pytorch/blob/master/project-dog-classification
2. Restnet101 https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html#resnet101
3. ImageNet Training in PyTorch https://github.com/pytorch/examples/blob/97304e232807082c2e7b54c597615dc0ad8f
4. PyTorch Documentation https://pytorch.org/docs/master/
5. Convolution Neural Network (CNN) https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53