Machine Learning Engineer Nanodegree

Project “Dog Breed Classifier”

Capstone Proposal

Magda Mirescu

03.04.2021

1. Domain Background

As a dog lover, I must confess, I often find myself wondering about the breeds of certain interesting dogs that I come across while wandering the streets of beautiful Vienna. After finding out what the 3 project proposals are, I instantly knew, the dog breed classifier has my name written all over it.

On a more technical note, I am very much aware of the fact that image classification is one of the most established applications of machine learning. The correct labeling of images uses a particular class of neural networks[1], where convolution is involved in at least one of the many employed layers, hence the name *Convolutional Neural Network (CNN)* emerged.

CNNs are a special class of deep neural learning[2] that is particularly used for both audio and video recognition.

2. Problem Statement

The scope of this project is to develop a dog breed classifying tool that can be used to ascertain whether a given input image is a dog or not and if the former holds true, it can further determine the breed of the dog in the image, provided the dogs in the input images are of so-called "pure-breed" type. This will be done by developing a CNN machine learning model.

More specifically, this tool should go through the following steps:

1) The tool determines whether the input image contains a human face, a dog or neither.
2) If a human face is identified, it should return the dog breed that it resembles the most.
3) If a dog is identified, its dog breed should be rendered.
4) If neither is identified, an error message should pop up.

3. Datasets and Inputs

This project uses the datasets that are kindly provided by Udacity, namely:

- A human face dataset of length 13233;
- A dog images dataset of length 8351.

The human face dataset consists of 5750 folders, most of which containing only one photo, with some displaying a plethora of images of the same person, see e.g. the Winona Ryder folder, which exhibits 24 different photos of the actress. The dataset contains on average just 2 images per person, whereas the minimum lies at 1 and the maximum at a whopping 530 images per person.

The dog images dataset, on the other hand, reveals a different structure – a categorization in "train", "test" and "valid" folders – which represent a strong indicator that machine learning algorithms will be trained, validated and tested using these data. Each data category contains 133 subfolders representing the different pure-breeds to be used for learning. Out of the 8351 total dog images, 6680 (approximately 80%) are destined for training, whereas 835 and 836 (approximately 10%), respectively, are reserved for validation and testing, respectively. The training dataset contains on average 50 images per dog breed, with a dog breed being

characterized by a minimum of 26 and a maximum of 77 images. The validation dataset has an average of 6, a minimum of 4 and a maximum of 9 images per dog breed, whereas the dataset used for testing displays on average 6, a minimum of 3 and a maximum of 10 images per dog breed.

Both dogs and humans datasets are highly imbalanced, as different persons and dog breeds display a varying amount of available files, with the greatest imbalances being present in the dogs training and human faces datasets.

The human images all have roughly the same dimensions (250 x 250 pixels) and are loaded as color (BGR) images, which are then converted to grayscale for face recognition and to RGB for plotting. The dog images are all color images that have varying dimensions, with some images displaying e.g. 1600 x 1200 pixels, while others contain e.g. 600 x 338, 334 x 250, etc. pixels. The varying dimensionality indicates that a transformation or resizing of the input image to a common dimension appears to be necessary when dealing with the dog images.

4. Solution Statement

The solution to this classical image classification problem is to develop a machine learning model that uses convolutional neural networks to identify the breed of a dog based on an input image. Furthermore, the model is expected to discern between human faces and dogs and, on a very funny note, suggest a dog breed that the human face looks closest to. First, a CNN model will be developed from scratch and in a second step, a CNN model will be developed using Transfer Learning.

5. Benchmark Model

Considering the fact that 2 CNN models will be developed in this project, one from scratch and the other one via Transfer Learning, different standards need to be imposed since starting from scratch is more of a challenge than using available knowledge. For the former more rudimentary model we impose an accuracy value of at least 10%, whereas the latter more sophisticated model should ultimately render an accuracy of at least 60% as a standard requirement. These are the minimum prerequisites requested in the standard Jupyter Notebook sheet provided by Udacity.

While researching benchmark models[3] for the Transfer Learning model, I came across the Kaggle competition for Dog Breed Identification, where I found different pre-trained models used by peers in their work. Among those I picked the InceptionV3 model as my benchmark, which for the user Dhawal Soni[4] rendered an accuracy of 90.62% and a log loss of 0.41936. With my Transfer Learning model I aim at surpassing the minimum standard requested by Udacity (60% accuracy) and hope to come closer to the accuracy achieved via the InceptionV3 model (90.62% accuracy) than the minimum requirement.

6. Evaluation Metrics

The goal of this project is not just to simply develop a CNN-model, but to develop a CNN-model that has a good performance. The way performance will be measured is by means of cross-entropy loss (or log loss) as an evaluation metric, since the most widely used metric, accuracy, would in this case render misleading results as the dataset classes are highly imbalanced (see Section 3. Datasets

and Inputs). Thus, the multi class log loss will be minimized, while the classical accuracy of prediction will also be displayed as supplementary information.

7. Project Design

The modus operandi (MO) for this project can be roughly structured in 7 different steps or stages:

Stage 1: Import Datasets

In this primordial step, the two different datasets described in Section 3 “Datasets and Inputs” are imported in Python / Jupyter Notebook.

Stage 2: Detect Humans

In step 2 a function (that uses OpenCV and Haar cascades) will be developed to detect human faces. The performance of this function will be tested based on the first 100 images of both datasets.

Stage 3: Detect Dogs

In step 3 we will make use of the pre-trained VGG-16 model (which was already trained on the ImageNet datasets) to detect dogs. At this point, we are already capable of differentiating between humans and dogs.

Stage 4: Create a CNN to Classify Dog Breeds (from Scratch)

In this stage, a CNN model will be created in PyTorch from scratch. The purpose of this CNN-model is to classify dog breeds.

Stage 5: Create a CNN to Classify Dog Breeds (using Transfer Learning)

In this step an alternative CNN dog classifying model to the model in Stage 4 is created using Transfer Learning. This model will also be implemented in PyTorch, however, as opposed to the model from Stage 4, it will employ pre-existing knowledge (in this case the ResNet-50 model[5]) as a starting point, aka before fine-tuning. This model should exhibit superior performance relative to the former model, as it employs already existing knowledge, which gives it the upper hand.

Stage 6: Write your Algorithm

Simply put, everything done in Steps 1 to 5 converges towards Step 6 in the sense that everything developed so far will be used here. Based on an input image, we will determine whether a human or a dog can be identified in that image. Next, a dog breed will be rendered based on the model developed and calibrated in Step 5.

Stage 7: Test your Algorithm

In this last and final step the algorithm will be tested with images that I will get to pick.

8. References

[1] https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-hdstat-rnn-deep-learning.pdf
[2] Valueva, M.V.; Nagomov, N.N; Lyakhov, P.A.; Valuev, G.V.; Chevyakov, N.I. (2020) "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". Mathematics and Computers in Simulation. Elsevier BV. 177: 232-243
[3] https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce
[4] https://www.kaggle.com/dhawalsoni/keras
[5] https://iq.opengenus.org/resnet50-architecture/#:~:text=ResNet50%20is%20a%20variant%20of,explored%20ResNet50%20architecture%20in%20depth.