Capstone proposal: CNN Dog Breed Classifier

Udacity’s Machine Learning Engineer Nanodegree

By: Luis Martín

Domain background

Image classification is a traditional problem in Artificial Intelligence, belonging to the field of Computer Vision. A problem traditionally considered at the "human level", which has been a test for different Machine Learning algorithms. More concretely, some of the benchmark problems include **character recognition or OCR** (MNIST[1]), and more recently **person and object identification**, tasks that have been tackled thanks to deep learning algorithms, such as **Convolutional Neural Networks**, with applications in many different fields, from Security to Autonomous Driving[2].

Convolutional neural networks (LeCun, 1989[3]) changed the world. Although showing impressive results back in the XX century, mainly in the recognition of handwritten characters, it wasn't until 2011 were they became vastly popularized, during the era of Deep Learning. CNNs achieved superhuman performance in several tasks of image analysis, winning several image recognition competitions, like the popular IMAGENET challenge[4].

A **convolutional network** makes use of backpropagation to find the parameters adjustment, but with the peculiarity of applying a convolution operation to the inputs. A convolution is a matrix operation that acts as a filter. The filter is applied to each input example, that is represented as a matrix, and gives us as output a transformation of the original example. In a convolutional network, the parameters that are learned through backpropagation are the numbers contained in the filters.

Problem statement

This problem is proposed as a capstone project candidate by Udacity. We want to make use of **convolutional networks** to classify dog images and figure out their breed. We are presented with a dataset containing images of dogs, and want to learn how to determine, given a new image, the breed of a dog in the image.

We assume that the images to be classified contain one and one only dog. Images without a dog are also not possible. Images are supposed to be colorful, and with a resolution like the ones in the dataset.

As mentioned, we will make use of convolutional neural networks. We will compare a **transfer learning** solution with other one developed **from scratch**.

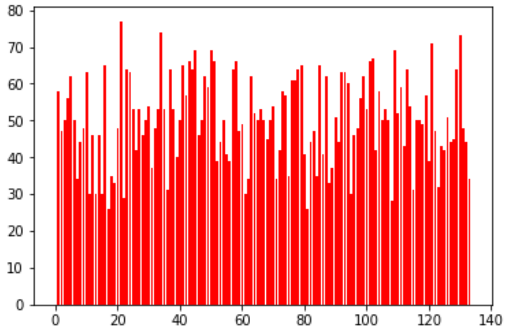
[1] Lecun et al., «Gradient-based learning applied to document recognition».
[2] Al-Qizwini et al., «Deep learning algorithm for autonomous driving using GoogLeNet».
[3] LeCun et al., «Handwritten Digit Recognition with a Back-Propagation Network».
[4] Krizhevsky, Sutskever, y Hinton, «ImageNet Classification with Deep Convolutional Neural Networks».

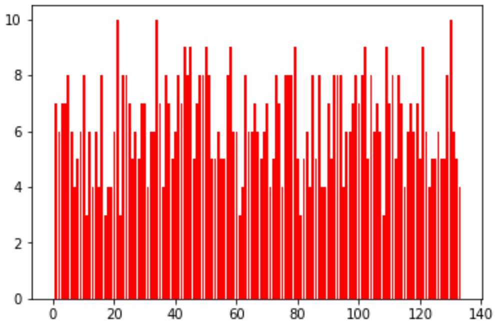
Datasets and Inputs

We are provided with a fully categorized set of dog images. This dataset contains a total of 8351 dog images, with 133 different breeds. The images are organized in three different folders: train, valid, and test. Inside each of these folders, images are distributed the following way:

- Train set: 6680 images
- Validation set: 835 images
- Test set: 836 images

The distribution among classes is not homogeneous. For example, the following histograms shows the distribution of class representation in both the training and test set:

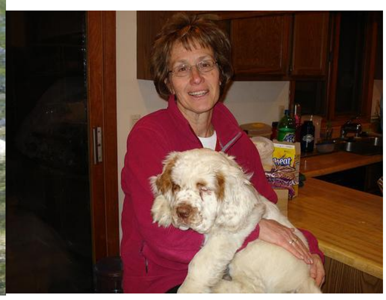




Although it is difficult to tell specific data heterogenities, these graphs show clearly that 1) the distributions are different, and 2) there is a significant variance between most and least represented classes in both datasets. For example, the least represented class in the training set has 26 images (the Norwegian buhund), while the most represented has 77 (the Alaskan malamute).

The images have a variety of different resolutions, including widths going from a minimum of 121 pixels to 3456, and heights from 153 to 3072 pixels. The images are also colorful. However, they are taken from different angles and cover different parts of the dogs' bodies, so we will need to do some pre-processing/augmentation in order to learn the right attributes. Some images also present humans. Here are some examples of images in the dataset:





Solution statement

We will design two different **convolutional network**-based architectures. The first one will be made from scratch using the PyTorch[5] library. We will make some experimentation with image pre-processing, using techniques like augmentation, cropping, rotation, etc. We will also try different configurations for the network, including network architecture, optimizer types and hyperparameters like learning rate.

The second solution will be based in **transfer learning**. We will make use of an already trained large model, like RestNet50, and add a classification network at the end that will be trained for our problem. We will compare these two approaches in terms of accuracy and performance.

Benchmark model

We will compare the performance of our solution with state-of-the-art networks solving similar problems, like for example VGG16, that is already proposed in the notebook.

Of course, the **transfer learning** solution acts as a benchmark itself, as we will be using an already trained network.

Evaluation metrics

We will be measuring the classification accuracy against the already proposed **test** dataset. This metric is calculated by calculating the percentage of correctly classified examples in the dataset. Also, we plan to try some other relevant metrics for classification problems, such as Area Under the Curve, since the variety of classes can lead to this necessity.

In addition, we will be providing some performance metrics interesting for the engineering process, such as training time, inference time and weight of the models.

Project design

We identify several components for the completion of this project:

1. End to end basic notebook completion with simple solutions
2. CNN from scratch
    a. Experiment the relevance of different image transformation methods

[5] «PyTorch».

    b. Experiment the relevance of different neural network architectures
    c. Compare at least two optimizers
    d. Experiment with learning rate values
3. Build and train the transfer learning solution
4. Build classification algorithm
5. Compare solutions and document the process