**INTRODUCTION**

The World Canine Organization (FCI) is currently listing more than 300 officially recognized dog breeds. Over thousands of years, mankind has managed to create an impressive diversity of canine phenotypes and an almost uncanny range of physical and behavioral characteristics of their faithful four-legged friends. However, apart from sinology scholars, dog breeders and some proven dog lovers most people shrug their shoulders in a clueless gesture, when asked to name the breed of a randomly presented dog, at least when it is not exactly a representative of one of the most popular and well known breeds like Dachshund, German Shepard or pug. If you are one of the few people who finds it slightly embarrassing not being able to identify dogs like a sinologist, you are probably pleased to learn that there might be a technical solution. Because thankfully, the aspiring and astonishing field of Deep Learning and artificial neural networks provides powerful concepts and methods for addressing this sort of classification tasks.

In this project we will develop ideas for a dog identification app using deep learning concepts. The software is intended to 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.

**CREATE A CNN TO CLASSIFY DOG**

Create a CNN to Classify Dog Breeds (from Scratch)

Now we will come to the really interesting part and tackle the implementation of the app’s principal task to tell the correct dog breed label from an image of a dog. We could make things easy and just use the pre-trained model from step two and predict the dog breed labels defined in the categories of the ImageNet dataset. But of course it’s much more exciting, interesting and educational to build our own solution, so here we go. Before we start building our own classifier, a few words about convolutional neural networks.

Convolutional neural networks (CNNs) are a class of deep neural networks primarily used in the analysis of images. To a certain extent, the design of convolution networks was inspired by the way in which a mammal’s brain processes visual impressions. Translation invariance and shared weights are mostly cited to explain the advantages of CNNs over using other types of neural networks in image analysis. The architecture of a convolution network involves the use of multiple hidden layers that perform mathematical convolution operations on their input.

**DETECT DOGS**

There is no comparable “dog detector” available for OpenCV’s Cascade Classifiers. Therefore, I choose another approach by employing an image classification model which has been pre-trained on the vast image database of ImageNet. More specifically, we will use the high-level deep learning API Keras to load the ResNet-50 convolutional neural network and run images through this model. For a specific image the network predicts probabilities for each of 1000 image categories in total. I have attribute a positive dog detection to an image, if the model assigns the maximum probability to one of the 118 dog related categories.

**Use a CNN to Classify Dog Breeds (using Transfer Learning**)

The general idea behind ​​transfer learning is the fact that it is much easier to teach specialized skills to a subject that already has basic knowledge in the specific domain. There are a lot of neural network models out there that already specialize in image recognition and have been trained on a huge amount of data. Our strategy now is to take advantage of such pre-trained networks and our plan can be outlined as follows:

find a network model pre-trained for a general image classification task

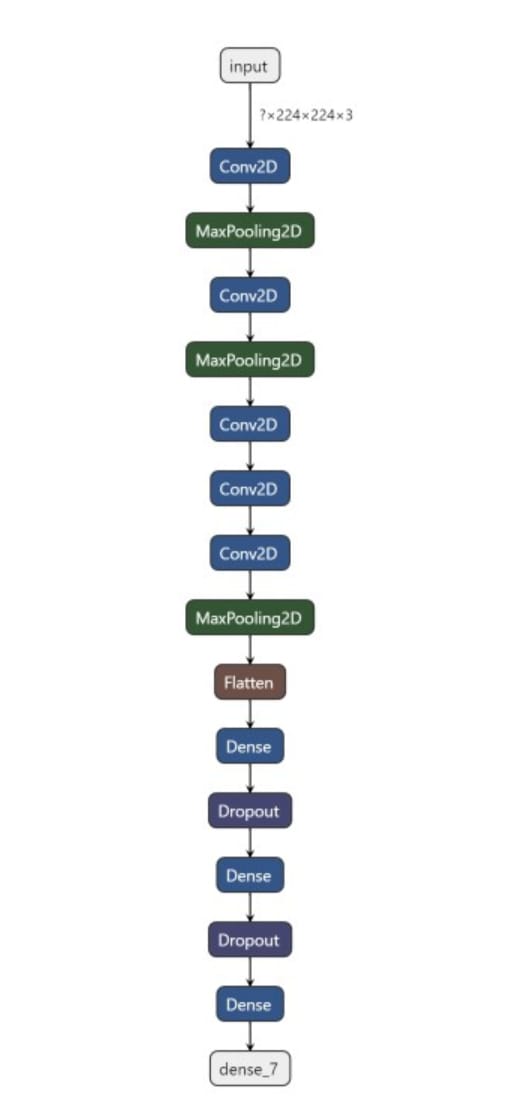
load the model with the pre-trained weights

drop the “top of the model”, i. e. the section with the fully connected layers, because the specific task of a model is generally defined by this part of the network

run the new data through the convolutional part of the pre-trained model. (this is also called feature extraction and the output of this step is also called bottleneck features.)

create a new network to define the specific task at hand and train it with the output (the bottleneck features) of the previous step.

As we will see in a moment, the structure of the model into which we stuff the bottleneck features can usually be quite simple because a large part of the training work has already been done by the pre-trained model. In step 4 of this project Udacity is providing some kind of blueprint for this strategy by having already fed our images dataset into a pre-trained VGG16 model (another classic in the field of CNN models for image classification) and making available the output as bottleneck features, which we can now feed into a very simple training network that essentially consists of just one global average pooling layer and a final dense output layer.



**OUTCOMES**

In this project I developed several approaches for the development of an app for the identification of dog breeds, and we achieved our best results with the application of a transfer learning model. I obtained an accuracy of 83% in our tests. I also learned how to build convolution networks from scratch, which was a very educational undertaking, even though we soon realized that there are significantly more promising methods, particularly with the application of transfer learning.

However, we still see several options to further improve our algorithm in the future:

I could gather more training data.

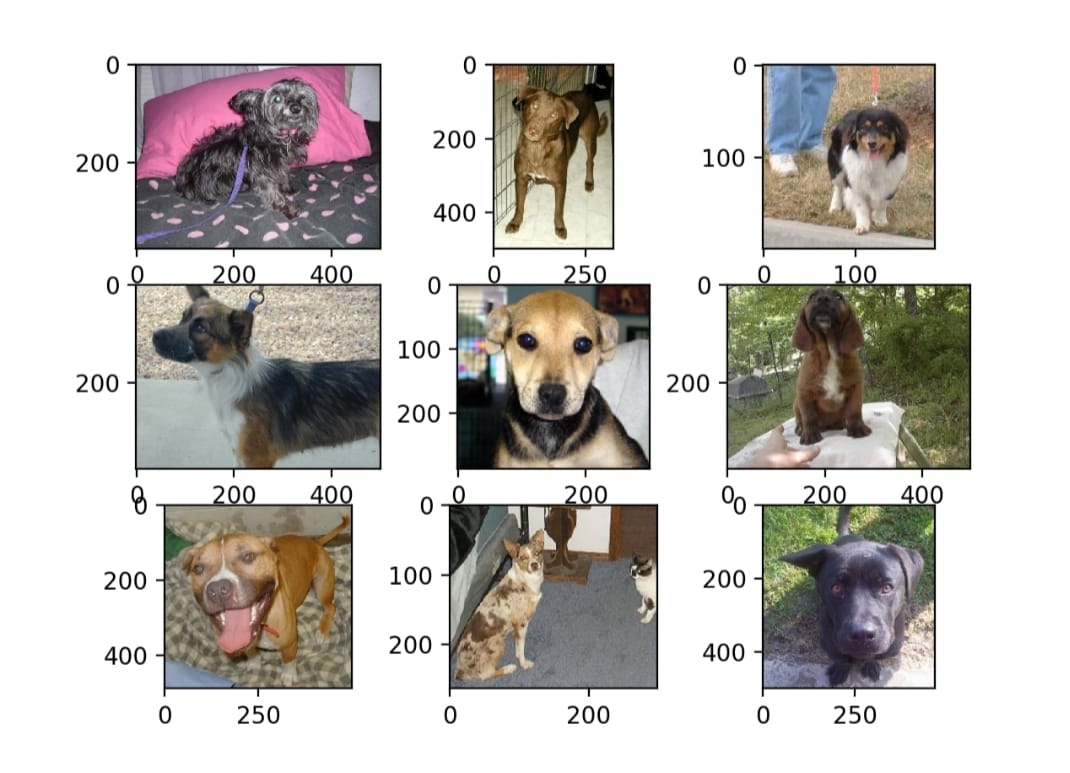
I could employ data augmentation to prevent overfitting.

I could add more layers to make our model more complex and hopefully more powerful.

I could extend our training time and add more epochs to the training.

But all in all, the accuracy levels from our tests, along with the tests with specific sample images, suggest that I already have a serious model we could work with in a real app.

**EXPECTED OUTPUT EXAMPLE**

****

**TRAINING DATASET**

Having a good training dataset is a huge step towards the robust model. There is Stanford Dogs Dataset with ~20K images of dogs of 120 breeds. Every image in the dataset is annotated with the breed of a dog displayed on it. As you might have noticed, having only 100 images of 120 different breeds is not enough to train a deep neural network. Convolutional neural network (CNN) is by all accounts the best machine learning model for image classification, but in this case, there are not enough training examples to train it. It would not be able to learn generic enough patterns off this dataset to classify different dog breeds. Most likely, it will just overfit to this small amount of training examples so that accuracy on the test set will be low. There are two possible approaches to mitigate the lack of training examples:

Merge dogs dataset with another bigger dataset with images and train a CNN on these merged examples.

Take an already pre-trained deep neural network on a larger dataset, cut into it, and attach an additional “classification head” i.e. several additional fully connected layers with the SoftMax layer on top of them.

The first approach has two big downsides: a much bigger amount of data has to be analyzed and the training on this big dataset will take much more time and resources. The second approach seems to be promising: the training has to be executed on the original dataset and training the “classification head” which has just several fully connected layers will not require a lot of time and resources.

**Conclusion:**

With a very good accuracy the dog breed classifier performed well. The algorithm correctly classified the breeds from step 4 which is a very challenging task for human beings as well. The accuracy can be further enhanced by data augmentation. Data augmentation enables the network to differentiate the features irrespective of the orientation and scale. Clearly, building a convolutional neural network using transfer learning yielded a far better accuracy than building it from the scratch. However, the model architecture used in Part 4 overcome the cons of the method used in ref.(1) in which the units are so fragile during training which poses the risk of the gradient flowing through the unit being zero forever from a point. It also overcomes the challenges posed in ref(2) where the number of parameters are doubled for every single neuron.