

# Convolutional Neural Networks

## Project: Write an Algorithm for a Dog Identification App

What dog breed does Arnold Schwarzenegger look like? It's not easy to answer that question objectively! Or is it?

### problem statement

To answer that question scientifically without any bias an Artificial Intelligence (AI) is trained on various pictures of dogs. In 2021 with the use of the python programming language and various packages this is done in a matter of seconds. The only important question is what network (the brain behind the AI) to use.

The goal within this project is to classify images of dogs according to their breed. If the Human face is given as input it will suggest a dog breed.

### project design

The workflow is already defined in the Jupyter Notebook given by Udacity:

1. Step 0: Import Datasets  
Load the image datasets for dogs and humans.
2. Step 1: Detect Humans  
OpenCV is used to detect human faces.
3. Step 2: Detect Dogs  
A Pre-trained VGG-16 Model is used to detect dogs in images.
4. Step 3: Create a CNN to Classify Dog Breeds (from Scratch)  
Create a CNN that classifies dog breeds from scratch.
5. Step 4: Create a CNN to Classify Dog Breeds (using Transfer Learning)  
Use transfer learning to create a CNN that can identify dog breeds.
6. Step 5: Write your Algorithm  
Write an algorithm to solve the problem statement.
7. Step 6: Test Your Algorithm  
Test of the algorithm on self-provided images.

### datasets and inputs

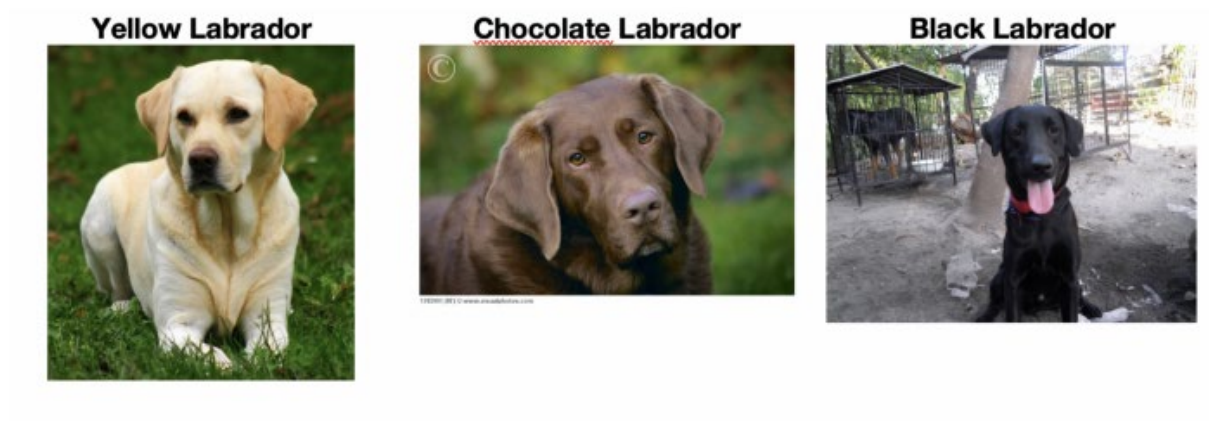
Analysing the given Dataset:

<https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip>

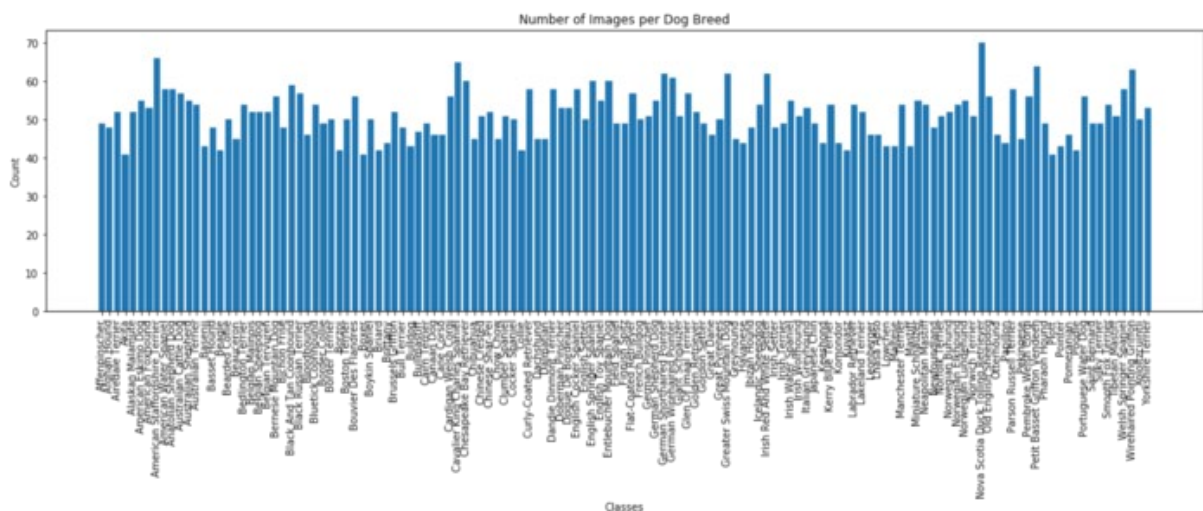
```
There are 133 total dog categories.
There are 8351 total dog images.
```

There are 6680 training dog images.  
There are 835 validation dog images.  
There are 836 test dog images.

In the dataset there is an average of about 63 pictures per breed including all type of coloured Labradors.



We now have a detailed look at the distribution of the images over the various dog breeds. Within the training set we have an average of 51 images per breed with a minimum of 41 images, a maximum of 70 images and a std deviation of about 6. Therefor we can consider the dataset classes to be closely balanced.



## solution statement

We will use Convolutional Neural Networks (CNN) to make a model. CNN is a part of deep neural networks and can be used for classifying images.

A CNN basically is a large set of tables that stores all the informations on how to link a feature set to some labels. The tricky part is to select the number of tables, the sizes and how to mathematically link them together. This structure can be thought of as filters that are applied to our images. Each of this filters point out a certain feature of the image i.e. a horizontal line and lead to a classification of the image.

Two different Convolutional Neural Networks (CNN) are trained within this investigation:

- A simple CNN based on the guide by cs231n
- A pretrained ResNet50

The simple CNN is based on the one provided from Udacity (source: Udacity):

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 223, 223, 16)	208
max_pooling2d_1 (MaxPooling2D)	(None, 111, 111, 16)	0
conv2d_2 (Conv2D)	(None, 110, 110, 32)	2080
max_pooling2d_2 (MaxPooling2D)	(None, 55, 55, 32)	0
conv2d_3 (Conv2D)	(None, 54, 54, 64)	8256
max_pooling2d_3 (MaxPooling2D)	(None, 27, 27, 64)	0
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 64)	0
dense_1 (Dense)	(None, 133)	8645
Total params: 19,189.0		
Trainable params: 19,189.0		
Non-trainable params: 0.0		

INPUT

CONV

POOL

CONV

POOL

CONV

POOL

GAP

DENSE

The most common form of a ConvNet according to the guide by cs231n is:

- INPUT  $\rightarrow$   $[[\text{CONV} \rightarrow \text{RELU}] * N \rightarrow \text{POOL?}] * M \rightarrow [\text{FC} \rightarrow \text{RELU}] * K \rightarrow \text{FC}$

We can fit the sample into that common structure and get the following parameters:

- $N=1$  ( $N \geq 0$  and  $N \leq 3$ )
- $M=3$  ( $M \geq 0$ )
- $K=1$  ( $K \geq 0$  and  $K < 3$ )

Using a pretrained model:

VGGnet and Resnet proved to be very good for image classification in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC): <https://image-net.org/challenges/LSVRC/>

They are available pretrained on various classifiers and can easily be modified to predict own labels. Using a pretrained model speeds up the process a lot and as shown also increases the accuracy of the prediction.

Using a pretrained model is possible because basic filters (i.e. detecting horizontal lines) are actually the same. The difference is just the final classifiers.

The ResNet50 gives about 75% while the simple CNN only about 10%.

### **benchmark model**

10% does not seem much at first. But as there are 133 different dog breeds random guessing would only result in less 1% correct guesses. On the other hand, a state-of-the-art network achieves 75% with only little effort.

Some ideas on how the model performance could be improved:

- Try other pre-trained models (there are 3 more in the notebook)
- Using Data Augmentation to generate more data (63 pictures per breed is not that much)
- Tune the Hyperparameters (well yes, that's always an option ...)

### evaluation metrics



As we have shown the dataset classes are closely balanced therefor as metrics accuracy is used.

You can find out more about different metrics

here: <https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce>

### results

As stated, the model achieves an accuracy of about 80%, so not every image is classified correctly and I managed to find especially bad samples.

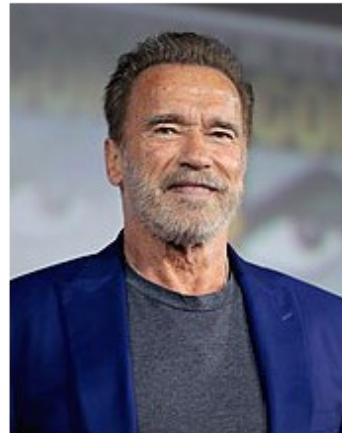
Dogs	Humans
This is a Irish Water Spaniel 	This photo looks like a Entlebucher Mountain Dog 



This is a Australian Cattle Dog



This photo looks like a Dogue De Bordeaux



This is a Cardigan Welsh Corgi



This photo looks like a Basset Hound



So well this time it seems Schwarzenegger looks like a Dogue De Bordeaux.