Magali Semeria
Udacity ML engineer nano degree
2021-02-13

Capstone Project Proposal

Dog breed classification web application

This project showcases an application of computer vision techniques, in particular image classification techniques, to identify dog breeds in images.

Domain Background

Computer vision is a thriving scientific field in machine learning. Its aim is to automate a task easily achieved by the human brain: recognizing objects from visual data. Computer vision has many applications in various domains. For instance, Facebook, Google's photo app and Adobe's Lightroom use computer vision to identify people in pictures. Self driving cars rely on the correct interpretation of their environment. Doctors use automated image analysis tools to help diagnose tumours.

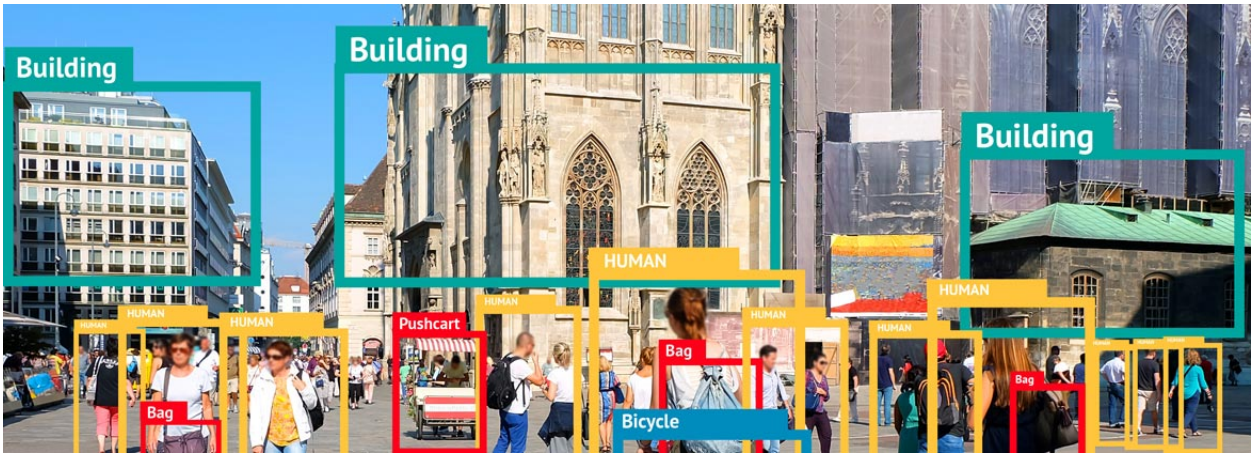


Image from https://microsap.co.uk/services/computer-vision-and-ocr/

The foundations of computer vision were laid in the late 1960s [1]. In 1968 Hubel and Wiesel characterized different types of neurones involved in the visual cortex and proposed a model for

[1] https://en.wikipedia.org/wiki/Computer_vision#Recognition

recognition tasks based on their discoveries[2]. Throughout the 1980s and 1990s, researchers gradually developed Convolutional Neural Networks (CNNs). CNNS are deep learning models that are predominant today in image classification[3].

CNNs take as input a 3D matrix representing an image. The 3 dimensions are always:

- The length of the image in pixels
- The width of the image in pixels
- A tuple containing the 3 colour channels (RGB) of the image

CNNs' output is a class label. In the example pictured below, the model's input is a colour picture and the output is a binary class value (is there a cat in the picture, yes, or no).

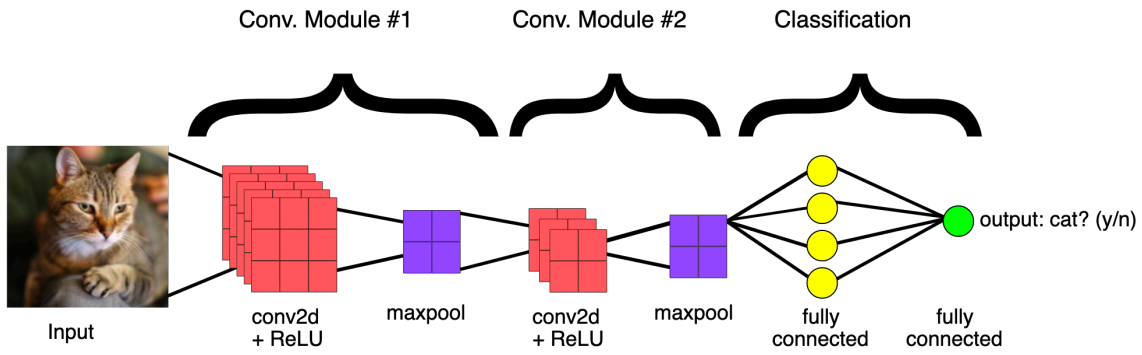


Image from https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks. « The CNN shown here contains two convolution modules (convolution + ReLU + pooling) for feature extraction, and two fully connected layers for classification. Other CNNs may contain larger or smaller numbers of convolutional modules, and greater or fewer fully connected layers. Engineers often experiment to figure out the configuration that produces the best results for their model. »

CNNs are based on 3 different types of mathematical operations:

- Convolutions
- Rectified Linear Unit (ReLU) transformations
- Pooling

These operations are described in details with helpful animations in Google's developer course on CNNs[4].

---

[2] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1557912/

[3] https://en.wikipedia.org/wiki/Convolutional_neural_network#Receptive_fields_in_the_visual_cortex

[4] https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

Building a CNN from scratch involves choosing a sequence of layers using convolutions, ReLu or pooling.

Problem Statement

In this project, we'll build a web application that classifies dog breeds. For people like me who are wistfully ignorant in the dog domain, such an application will be a precious tool to befriend dog owners. Dog lovers will also enjoy submitting pictures of their favorite companion and seeing if the application guesses correctly.

Dog breed classification is particularly interesting as a « playground » project in image classification because there exist well-curated datasets of dog images (the most famous is the Stanford dogs dataset[5]). It is also one of Udacity's proposed capstone project[6], which means we'll benefit from instructions included in a Jupyter Notebook[7] to develop the models.

We'll build a simple web application that takes as input a picture of a dog and returns the name of the corresponding dog breed. As suggested in the project notebook, we'll first develop a dog detector that will return True if a dog is present in the uploaded picture. We will then develop a dog breed classifier that will return the dog breed of the dog in the image. For additional fun, we'll develop a human face detector that will detect if a human face is present in the uploaded image. If so, we'll use the dog breed classifier to return the predicted dog breed for the human. If neither human or dog is detected in the picture, the application will not return a prediction.

Datasets

Dog dataset

We'll use Udacity's dog dataset available at https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip to train and test the dog breed classification models. The dataset contains 8351 images of dog belonging to 133 different dog breeds. Images are already split into train (6680 images), test (836 images) and validation (835 images) sets.

Let's look at the repartition of images in train, test and validation datasets:

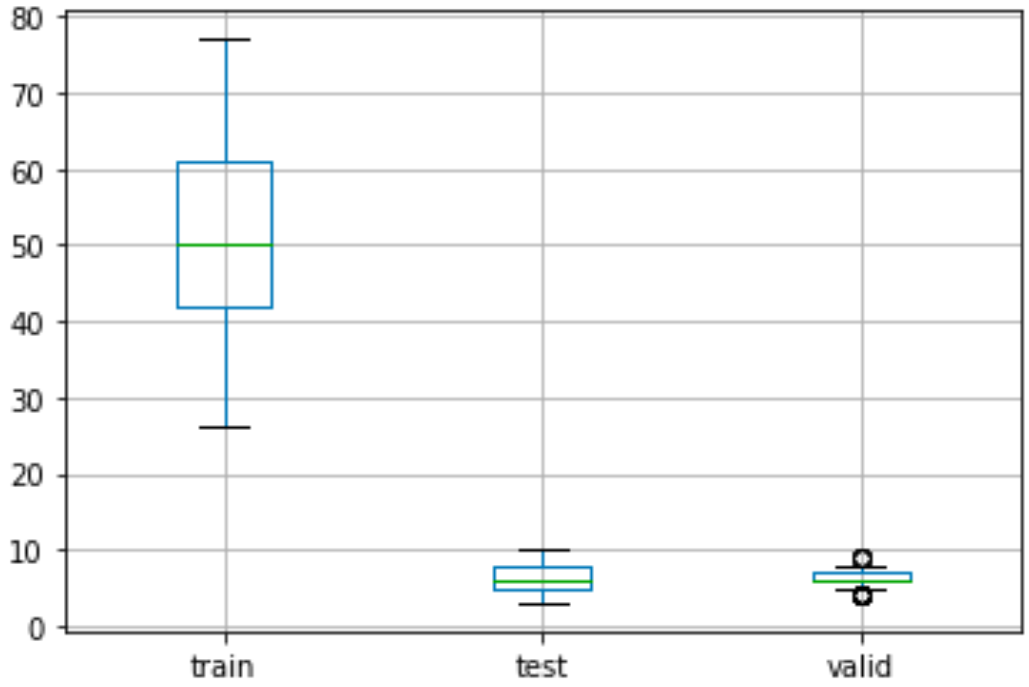
---

[5] http://vision.stanford.edu/aditya86/ImageNetDogs/

[6] https://github.com/udacity/deep-learning-v2-pytorch.git

[7] https://jupyter.org/

	Train dataset	Test dataset	Validation Dataset
Mean number of images	50,22	6,29	6,28
Std	11,86	1,71	1,35
Min number of images	26	3	4
Max number of images	77	10	9



Number of images in the train, test and validation sets

We can see that classes are slightly imbalanced. However we have at least 26 images for each breed in the train dataset, which should be enough to start with.

Human faces dataset

We'll use the University of Massachusetts « Labeled Faces in the Wild »[8] dataset available at http://vis-www.cs.umass.edu/lfw/lfw.tgz to assess the performance of the human detector model. The dataset contains 13233 images of human faces collected from the web using the

[8] http://vis-www.cs.umass.edu/lfw/

Viola-Jones detector[9]. 5749 people were identified. 1680 identified people have at least 2 images in the dataset. Of course, for this project, we are not interested in identifying the person in the picture.

Solution Statement

As suggested in the notebook, we'll use different CNNs to classify dog breeds and detect dogs and humans:

Dog detector

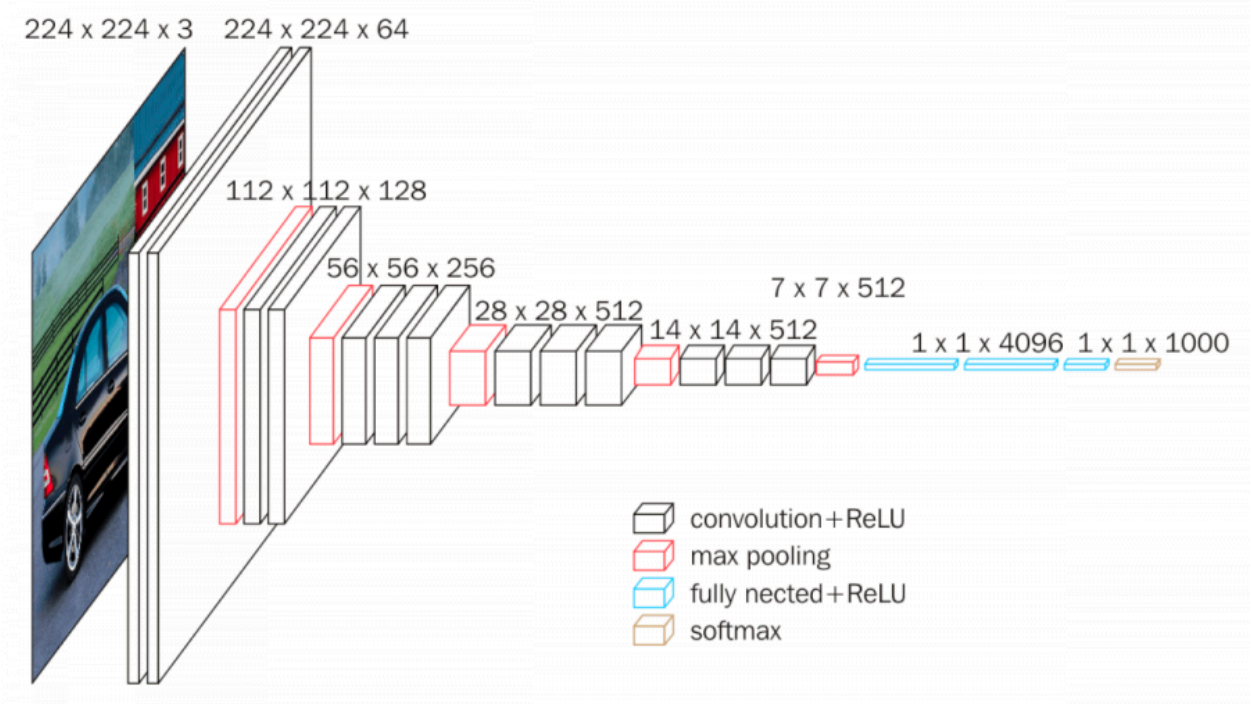


Image from https://neurohive.io/en/popular-networks/vgg16/

We'll use a pre-trained VGG16[10] model to detect dogs. VGG models were first published in 2015 by Karen Simonyan and Andrew Zisserman. VGG16 was trained on the ImageNet dataset[11], one of the largest dataset of images. ImageNet contains ~14 million hand-annotated images. Annotations fall into ~20,000 categories such as «strawberry », « bow-tie » or in the case that

[9] https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

[10] https://arxiv.org/abs/1409.1556

[11] http://www.image-net.org/

interests us « dog ». A category typically includes several hundreds of images[12]. VGG16 achieved an accuracy of 92.7% when predicting the top-5 classes present in ImagesNet's images.

VGG16 is available in pytorch's torchvision models[13].

Human detector

We'll use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images[14]. OpenCV provides many pre-trained face detectors, stored as XML files on github. We'll use the pre-trained face detector already available in the Udacity's dog classification project files.

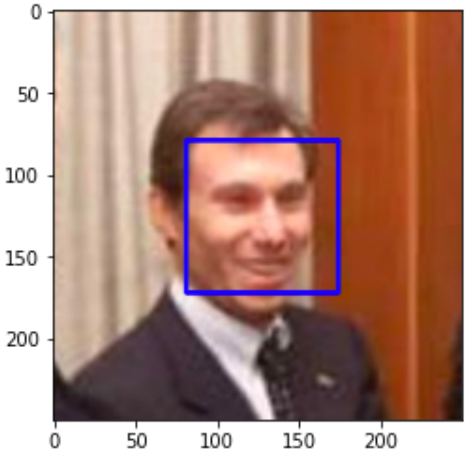
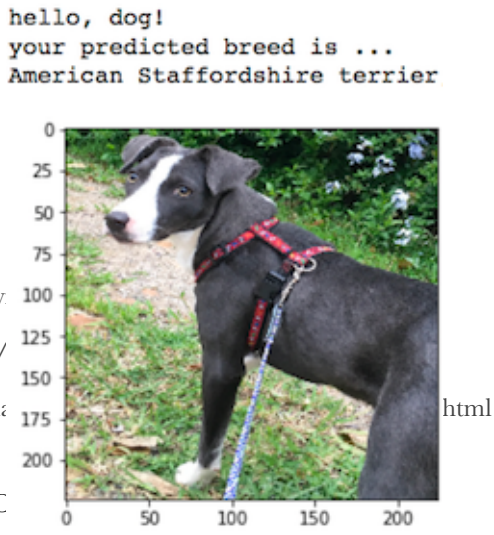


Image from https://github.com/udacity/deep-learning-v2-pytorch/blob/master/project-dog-classification/dog_app.ipynb

Dog breed classifiers



[12] https://en.wikipedia.org/w

[13] https://pytorch.org/vision/

[14] https://docs.opencv.org/ma html

Image from https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification

As suggested in the dog classification project notebook, 2 different models will be developed and tested to classify dog breeds:

- A CNN built from scratch, using a simple architecture derived from VGG16. This model will be developed inly for educational purposes. Such model is not expected to achieve a high accuracy for the task at hand.
- A transfer learning model using pytorch's pre-trained VGG16 model and a custom classifier. Transfer learning[15] is a machine learning technique that allows to take advantage of models trained on generic tasks (in our case object recognition) and adapt them to a more specific task (in our case dog breed classification). It is therefore especially adapted in our case. As a first transfer learning model, we'll use a VGG16 architecture in which the last fully connected layer has been replaced with a fully connected layer where the number of output neurones = the number of dog breed classes present in the dataset (133).

Benchmark Model

CNN built from scratch

We don't expect to reach a high accuracy for the CNN built from scratch. Classifying dog breeds is a challenging task because there can be considerable variability within a dog breed, and because some breeds are very hard to distinguish, even for humans. According to the project notebook « a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%. ». We'll follow the notebook, suggestion and be satisfied with an accuracy > 10%.

Transfer learning model

We found quite a few examples of transfer learning models for dog breed classification. Accuracy is often in the 40%-80% range[16][17][18]. We'll set ourselves an objective of accuracy>60%, as suggested in the notebook.

---

[15] https://en.wikipedia.org/wiki/Transfer_learning

[16] https://levelup.gitconnected.com/dog-breed-classification-using-cnn-and-transfer-learning-cc93a4497e90

[17] https://medium.com/@wangshuocugb2005/dog-breeds-classification-with-cnn-transfer-learning-92217cba3129

[18] https://towardsdatascience.com/dog-breed-classification-using-cnns-and-transfer-learning-e36259b29925

Evaluation Metrics

We'll use accuracy on the test datasets as our evaluation metric. The accuracy is computed according to the following formula:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

Project Design

Model development

First, we'll complete the dog classification notebook to develop the dog detector, the human detector and the 2 dog breed classifiers (CNN from scratch and transfer learning model).

Each model will be improved just enough to achieve an accuracy higher than the set threshold. Improvements may include modifications of the initial model architecture, hyper-parameters values or training algorithm. We'll also tinker with image transformations to enrich our initial dataset.

Web application development

Once we have a satisfying transfer learning model, we'll deploy it in a simple web application using Flask[19]. The workflow of the application will be as follows:

[19] https://flask.palletsprojects.com/en/1.1.x/

User uploads a picture

Image is pre-processed to fit models' input format

Dog detector predicts if the image contains a dog

No

Yes

Human face detector predicts if the image contains a human

No

Yes

The application does not display a prediction

Result

Hello dog! I think you are a ... Bulldog

Dog breed classifier predicts a dog breed