Machine Learning Capstone Proposal

Udacity Machine Learning Engineer Nanodegree

Author: André Vargas

Date: Oct 2021

Project: Dog Breed Classifier using Convolutional Neural Networks

Domain Background

What is Machine Learning?

On modern days, the algorithms are all around us. Language assistants that instantly translates from one language to another, self-driving cars, personalized movie suggestions in your favourite streaming platform, personalized shopping suggestions and automatic price setting, diagnosis of a disease based on collected data. All these tasks that a few years ago would be science fiction are possible today thanks to Machine Learning algorithms. A technology that allow computers and other devices to learn and solve different tasks by themselves without explicitly programming them how to do so.

The Collins English Dictionary defines Intelligence as "the ability to think, reason, and understand instead of doing things automatically or by instinct" [1]. For example, before lifting a vase we are able to predict its weight based on our intelligence, and know whether or not we are able to do it. This reasoning to predict certain situations is possible because of our memory from previous experiences, and it's exactly the same principles applied to Machine Learning.

Computers can learn trough associations of different types of data, which can be numbers, images, patterns, classes, and so on. And beyond a massive amount of problems to solve, image classification is one of the biggest areas of study in the domain of Machine Learning.

What is Image Classification?

There are hundreds of applications for image classification with Machine Learning. For example, let's say we want a machine to analyse pictures of recent harvested strawberries to see which of them are good or bad for humans to eat. Or lets imagine a factory that has a machine that produces five types of steel tools, and they want to use images to count how many bars of each kind are being produced everyday.

Regardless of being a simple classification problem with two categories, good strawberries and bad strawberries, or a more complex one like identifying steel bars, they are easy problems for humans simply because we just know in our brain how a good strawberry looks and how a bad strawberry looks, but it's impossible to tell a computer how to do it. And one of the solutions is to use Machine Learning algorithms to teach the machines how to identify specific patterns in these images that allow them to be classified.

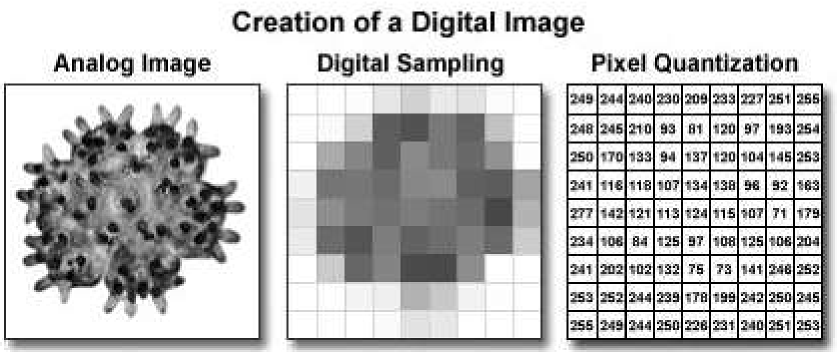
Picture 1 – A very bad and a very good strawberry [2]

One way to teach a machine how to understand these images, is by using Convolutional Neural Networks.

What is a Convolutional Neural Network?

A CNN is a specific type of neural network and its largely used for image classification. To understand a little bit better how a CNN works, it helps if we look at how the computers "see" an image.

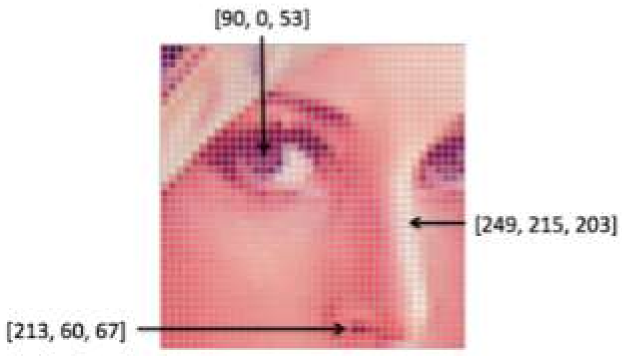
A black and white image is seen by the computer as a 2D matrix, and each position of this matrix represents a pixel of this image. The values for each element vary between 0 (black) and 255 (white). This can be seen in the example bellow:



249	244	240	230	209	233	227	251	255
248	245	210	93	81	120	97	193	254
250	170	133	94	137	120	104	145	253
241	116	118	107	134	138	96	92	163
277	142	121	113	124	115	107	71	179
234	106	84	125	97	108	125	106	204
241	202	102	132	75	73	141	246	252
253	252	244	239	178	199	242	250	245
255	249	244	250	226	231	240	251	253

Picture 2 – Creation of a Digital Image [3]

A coloured image, on the other hand, is represented by a 3D matrix in which we can store a combination of the colours red, green and blue (RGB) from 0 to 255. The image bellow in an example of that:



Picture 3 – Digitalization of a coloured picture [4]

So, considering that a machine can only see groups of numbers, one way to analyse and determine whether that image is a “cat” or a “dog” is by looking for specific patterns and characteristics for each image class. These operation of “reading” the matrices showed above, assign some weights and biases to some aspects of the image, and based on that differentiate the images is the basic premiss of a CNN. [5]

CNNs are also used in a variety of other fields like face recognition, document analysis, understanding climate changes, speech recognition, and so on. [6]

Problem Statement:

The goal of this project is to identify dog breeds from image inputs. This is a very complex problem because dogs are very similar among them in therms of body features and overall structure.

Dogs are one of the most diverse species on the planet, there are over 340 dog breeds known around the world. The American Kennel Club recognize over 170 official dog breeds, not counting mixed breeds and mutts.[7] So differentiating between all these categories is not an easy task for a machine to do. A very large dataset is required to accomplish this task.

The basic premiss is that given an image of a dog, the algorithm must identify an estimate of the canine’s breed. If supplied an image of a human, the code will identify the resembling dog breed.

Datasets and Inputs:

The entire dataset used in this project are images supplied by Udacity. There are a total of 8351 dog images separated in train (6680 images), test (836 images) and valid (835 images), and each group separated in 133 different dog breeds.

Also, since this project should be able to identify and give a breed to a human image, there are 13233 human images as well.

Some examples of images from the dog datasets:

Classes 072.German_shorthaired_pointer and 099.Lhasa_apso.

Examples from the human dataset:

Images from Harrison Ford and Roger Federer.

Solution Statement:

The solution requires 3 important tasks:

1) Receiving an image as in input and check if it can detect a dog: in order to complete this task, the pre-trained model VGG-16 will be used, along with weights that have been trained on ImageNet, a very large, very popular dataset used for image classification and other vision tasks. ImageNet contains over 10 million URLs, each linking to an image containing an object from one of 1000 categories, from which categories 151 to 268 are dog breeds.

2) If it can't find a dog, see it finds a person in that image: for this task we will use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images. OpenCV provides many pre-trained face detectors, stored as XML files on GitHub. We have downloaded one of these detectors and are going to use on this project.

3) If it sees a human or a dog from previous tasks, it will try to give a dog breed as output: this is where the Convolutional Neural Network (CNN) shows up. First, we will create a CNN to classify dogs from scratch, and then we will create a CNN using Transfer Learning, expecting a higher accuracy.

Benchmark Model:

The Benchmark Model in this project will be the CNN created from Scratch.

Evaluation Metrics:

The evaluation metrics for this project will be the test accuracy. My goal is to ensure a 30% accuracy on the CNN model created from scratch, and an 85% accuracy on the CNN model using transfer learning. I consider those decent results for both methods.

Project Design:

This project will be separated in 7 major steps:

1) Create a human face detector using OpenCV:

Using OpenCV's implementation of Haar feature-based cascade classifiers, we will define a function that analyses an image and returns True if it can spot a human face and False if it cannot spot a human face. This information will be used later in the final algorithm to determine what will be shown to the user, for example: "This is a human, but it looks like a Chihuahua".

2) Create a dog detector using VGG16 pre-trained model:

In the event of failing to identify a human in the input image, we will then use a VGG16 pre-trained model to see if its able to classify among the different dog breeds it has within its categories. The function will return True if it finds a dog, and false if it doesn't find a dog. This information will be used later, following Step 1 example, in two different ways: "This is a dog and it looks like a Dalmatian", or, if it can't see a dog, it will say "Sorry, I can't see a human or a dog in this image, please try another one".

3) Create a CNN to classify dog breeds from scratch:

Create the entire CNN architecture from scratch, which includes:

3.1) Defining the Constructor:

All Convolutional Layers, the Pooling Layer, the Dropout Layer and the Fully Connected Layer.

3.2) Defining the forward function:

How the model is going to be run, from input to output [8]

3.3) Define the data loaders:

Create three separate data loaders for training, valid and test datasets of the dogImages after preprocessing the data using torchvision's Transforms. We intend to use Transforms in the following way:

Train dataset: All images will be Random Resized Crop to 224x224 pixels, the size the model accepts the tensors, and the dataset will be augmented with a Random

Horizontal Flip and a Random Rotation, then the images will be converted to tensor so it can be interpreted by the model, and lastly normalized.

Valid and Test datasets: All images will be Resized to 256x256 pixels, and then will be Center Cropped to 224x224 pixels, the size the model accepts the tensors, converted to tensor and then normalized. No data augmentation on these two datasets.

3.4) Define a training function:

Train, validate and save the model for future uses.

3.5) Test the model:

Test the model on the test dataset of dogImages to see how well it perfoms. An accuracy of at least 30% is expected with a model trained in 100 epochs.

4) Create a CNN to classify dog breeds using transfer learning:

Define the best architecture to classify dog breeds using transfer learning, which includes:

4.1) Define the architecture:

Select among a few architectures available the best suited for this task.

4.2) Define the data loaders:

I will use the same data loaders used in the other CNN for comparison.

4.3) Define a training function:

Train, validate and save the model for future uses.

4.4) Test the model:

Test the model on the test dataset of dogImages to see how well it perfoms. An accuracy of at least 85% is expected with a model trained in 100 epochs.

5) Write the full algorithm:

Using the face detector function, the dog detector function and the trained model together, code an algorithm that will work as follows:

- Reads an image.

- Check if its a dog or a human.

- If its a dog, it will output the predicted breed.

- If its a human, it will output the breed it resembles the most.

- If its not a dog or a human, it will give an error as ask for another image.

6) Test the algorithm with images that are not in the dataset:

Using images that are not included in the dataset, I will test how well the model is performing.

7) Deploy the algorithm in two different ways:

7.1) In a Flask Web Application:

I will write a small flask web application that asks the user for an input image and will output the results of the algorithm defined in Step 5. I will try to host the application on Python Anywhere, but if it's not possible, I'll include the flask app files in the project submission explaining what it has to be done in order to put it to work.

7.2) In a Twitter Bot:

This is a more ambitious approach and it's something that I've never done before, but it looks like a fun challenge. The idea is to use Tweepy, the python library for Twitter API, and create a bot that will be called in a posted image of a dog or a person and reply to its mention with the dog breed contained in that image. I'm not sure if I'll be able to pull this off, but it's definitely worth the try.

References:

[1] – Definition of Intelligence by Collins English Dictionary – **https://www.collinsdictionary.com/dictionary/english/intelligence**

[2] – Image taken from "This Strawberry Hack Will Instantly Make Your Mushy Fruit as Good as New Again" – https://www.delish.com/food-news/a32475445/strawberry-hack-ice-water/

[3] – Image taken from "Concepts in Digital Imaging Technology" - https://hamamatsu.magnet.fsu.edu/articles/digitalimagebasics.html

[4] – Image taken from "Stanford AI Lab Tutorial 1: Image Filtering" - https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

[5]- Training Convolutional Neural Networks to Categorize Clothing with Pytorch - https://towardsdatascience.com/training-convolutional-neural-networks-to-categorize-clothing-with-pytorch-30b6d399f05f

[6] – "7 Applications of Convolutional Neural Networks" - https://www.flatworldsolutions.com/data-science/articles/7-applications-of-convolutional-neural-networks.php

[7] – "American Kennel Club: Dog Breeds" - https://www.akc.org/dog-breeds/

[8] – "Pytorch Tutorial: NN PACKAGE" - https://pytorch.org/tutorials/beginner/former_torchies/nnft_tutorial.html