Machine Learning Engineer Nanodegree

Capstone Proposal

Chidimma Nweke

November 2, 2020

Domain Background

A convolutional neural network (CNN or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery (Valueva, et. al). Deep learning (DL) in artificial neural network (ANN) is a branch of machine learning that attempt to model high level abstractions in data (A, 2015). Machine learning is arguably a field of artificial intelligence where computers are employed to learn without being explicitly programmed.

Convolutional Neural Networks (CNNs) have had many qualified successes in different backgrounds. In the field of image recognition, CNNs are used to effectively classify or detect objects in images by extracting specific features from the image. A CNN named AlexNet won the ImageNet Visual Recognition Challenge, which involved detecting and classifying objects in an image. In the field of drug discovery, they are being used to predict the interaction between molecules and biological proteins which can be used to identify potential treatments. Other interesting applications of CNNs include: Identifying human intention to control assistive devices (Electromyography (EMG) recognition), Video Analysis, Anomaly Detection, Time series forecasting, game developments, e.t.c.

Motivation

Health is paramount. Cancer is reportedly one of the leading causes of death in the United states. A total of 1,806,590 new cancer cases and 606,520 deaths are expected in the US this year, which translates to about 4,950 new cases and more than 1,600 deaths per day (source: American

Cancer Society). Building robust systems that can detect cancer cells in their most early stages has motivated me to delve into CNNs and other related machine learning fields of medicine.

Problem Statement

In this project, I will use deep CNNs to build app that detects dogs and human faces in images and returns the closest dog breed for the images.

Objective – Dog or human face detection in images and classification.

Performance – Percentage of correctly classified and detected images, and accuracy on the dog images dataset.

Datasets and Inputs

The dataset for this project is the dog dataset provided by Udacity, and Labelled Faces in the Wild (LFW) dataset which was created and managed by researchers at the University of Massachusetts. Both datasets can be downloaded here and here.

By analysis:

The dog dataset is made up of 8,351 images, 133 labels which is the different dog breeds. The dataset is already split into train, test and validation files. The train file contains 6650 images and all the 133 labels with about 50 images for each label while the test and validation files contain 836 and 835 dog images respectively.

In The lfw (labeled faces in the wild) dataset consists 13,233 faces of 5,749 people which were collected from the web, each face labelled with the name of the person in the image. 1680 people captured in the dataset has two or more distinct photos.

Solution Statement

To tackle this problem, I will employ transfer learning which involves using pretrained models. To detect human faces, I will use a Haarcascade face detector implemented using the OpenCV library. To detect dogs, I will use either a pretrained VGG-16 or a ResNet-50 CNN model. Then I will build and train a CNN model to classify the dog breeds. After, I will collect all these models and bundle them with a pre-processor for the images. Together, they make up the dog app.

Benchmark Model

VGG-16 model: The model was one of the leading models in the imagenet competition. Here, I will extract the classifier features, fine tune it and use it to classify the dog breeds.

The benchmark model is measured by its cross entropy loss (also negative log likelihood loss (NLLLoss)) and accuracy on the dog images dataset.

Evaluation Metrics

❖ Model accuracy

Model accuracy here is the percentage ratio of the correctly classified images to the total number of images in the dataset.

Mathematically:

$$accuracy = \frac{correct\ predictions}{size\ of\ dataset}\ x\ 100\%$$

This metric takes only the true negatives and positives into consideration

❖ Cross Entropy Loss

This metric actually involves two steps:

1. The first applies a Softmax function to any output it sees.
2. Then applies Negative Log LikelihoodLoss (NLLLoss).

It then returns the average loss over a batch of data.

Mathematically:

$$\ell_{CE}(p, t) = -[t \log p + (1 - t) \log(p - 1)]$$

Where: t = target, and p = prediction

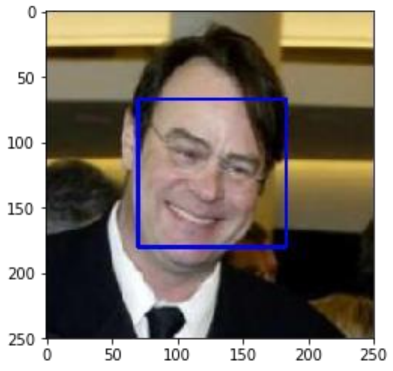
Project Design

I will go over the following steps to implement the problem's solution.

Step I: Importing and Exploring datasets.

As mentioned earlier above, I will be using the dog images and the lfw datasets. The most important data in this project is the dog data. The lfw data is only used to test out the implementation of the haarcascade face detector using OpenCV.

When the face detector model detects as face in an image, it draws a rectangular box over the space of the detected face as shown below.



Next, I will use either of the VGG-16 or the ResNet-50 pretrained models from pytorch to detect dogs present in an image.

Step II: Image Preprocessing

Before feeding these images into the models, they need to be preprocessed. The CNN models accept images as normalized tensors while the haarcascade detector accept grayscale images. To do this, I will define a transform job for the images going into the CNN and define a function that converts images to grayscale before they go into the haarcascade model.

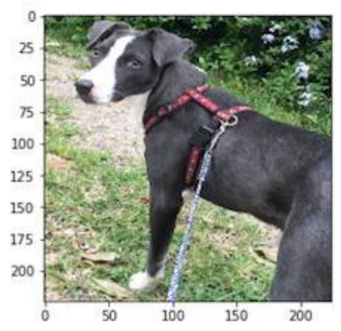
Step III: Building a CNN model

At this step, I will design my model architecture which I will use to classify the dog breeds, train and evaluate it, and measure its accuracy to that of the benchmark model which I will extract and evaluate too.

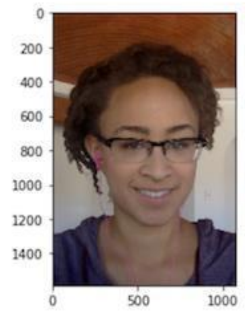
Step IV: Bundling models

Here, all my models are ready, I will bundle of them along with the image pre-processors. The app will return either of the following outputs to image inputs.

```
hello, dog!
your predicted breed is ...
American Staffordshire terrier
```

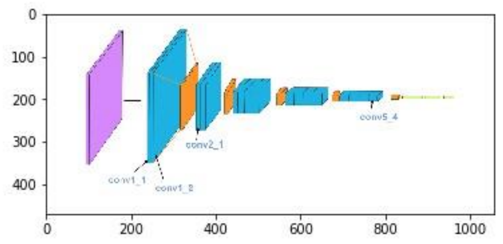


```
hello, human!
```



```
You look like a ...
Chinese_shar-pei
```

```
Oh dear.....
This is neither a dog nor a human!
```



Conclusion

Classifying dog breeds has become a tedious task as there are over 400 different breeds with many having close resemblance with each other. This app, when trained robustly and deployed to the market will make dog breed classification more efficient.

References

A, B. Y. (2015). *"Deep Learning"*. Cambridge: MIT Press.

Valueva, M., Nagornov, N., Lyakhov, P., Valuev, G., & Chervyakov, N. (2020). Application of the residue hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*. doi:10. 1016/j.matcom.2020.04.031