




UDACITY MACHINE LEARNING NANODEGREE CAPSTONE PROJECT REPORT

YUSUF OLODO

APRIL 28TH, 2020

PROJECT OVERVIEW

This project focused on building a web app that can classify and identify different breeds of dogs using a convolutional neural network and comparing the model which we built from scratch and the pre-trained models on Torch vision. Deploying this model behind a web application will enable users to classify different dog breeds. A fun element of the



app is that users can input human images and the model predicts the closest resemblance of dog breed to the human image.

PROBLEM STATEMENT

The main objective to solve in this project is to improve dog breed classification. This application can be useful in veterinary medicine for identifying the different breeds of dogs, including pet shelters. This will also help in search engines, as the algorithm can easily detect dog breed pictures accurately. As a fun element to the project, the algorithm will also predict the closest resembling dog breed to a human image. Accuracy of predictions can be easily measured and models can be trained to reduce loss function using gradient descent and backpropagation methods.

DATASETS AND INPUTS

The first step in this stage is to download the dataset of the human and dog images. There are 13233 human images and 8351 dog images used in this project.

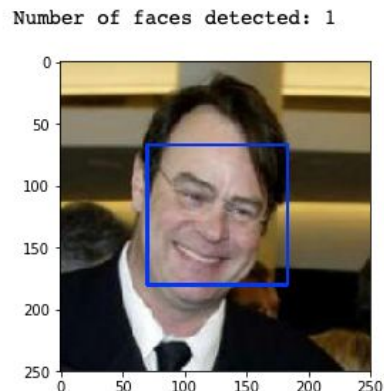
HUMAN FACE DETECTOR

The first step is to use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images. OpenCV is a library of programming functions mainly aimed at real-time computer vision

and it's paramount that we use some pre-trained models as benchmark models.

OpenCV provides many pre-trained face detectors **"haarcascade_frontalface_alt.xml"**.

OUTPUT OF THE DETECTOR




HUMAN-FACE

Using the face detection function and the haarcascade pre-trained model, ideally, we would like 100% of human images with a detected face and 0% of dog images with a detected face. The result is displayed below:

```
Number of faces detected in human files is 98%
Number of faces detected in dog files is 17%
```

Using another frontal face model from the haarcascade library **"haarcascade_frontalface_alt2.xml"**, the results obtained were much better for the number of human faces the model detected but the dog faces detected rose by 4% compared to the previous model.



```
Number of faces detected in human files is 100%  
Number of faces detected in dog files is 21%
```

DOG DETECTOR


The next step undertaken is to define a dog detector function that will output a 100% detection of dog images in the dog images file using a VGG-16 model, along with weights that have been trained on ImageNet. This model has become very popular in the research community due to its simple approach and because the pre-trained weights were made freely available online, facilitating the fine-tuning of this powerful model on new tasks[1].

The image is first converted to an image tensor as a pre-processing step for the pre-trained model, the model is then trained and evaluated on a new set of images.

The result of the VGG16 model is displayed below:

```
Number of dogs detected in human files is 0%  
Number of dogs detected in dog files is 100%
```

The next step is to use a computationally faster model than the VGG16 which is the resnet50 and see if we could reproduce the same results we got earlier with the previous model. The resnet is a highly modularized network architecture for image classification. The result of the pre-trained model on the both image datasets are displayed below:



```
Number of dogs detected in human files is 0%  
Number of dogs detected in dog files is 100%
```

METRICS

The metrics used in this project is accuracy of the models in predicting the test images. The data is split into train, validation and test datasets. The model doesn't train on the validation dataset but a low validation loss output determines the best performing model as this model can generalise well on the test data set.

Accuracy formula is given as $\text{correct predictions} / \text{total number of predictions}$.

The loss function used in the training of the model is CrossEntropy Loss or Log Loss which is mostly used for multiclass label classification. Cross-entropy loss increases as the predicted probability diverges from the actual label of the image datasets.

DATA EXPLORATION

The dataset has pictures of both dogs and humans which are 8351 and 13233 images respectively. There are 6680 train images, 836 test images and 835 validation images in the dog images dataset. Each of the train, test and validation dataset have 133 folders corresponding to dog breeds.

The images are of different sizes and different backgrounds which suits a CNN architecture for prediction rather than a Multi-layer Perceptron architecture.

The data is not balanced as the number of images provided for different breeds of dog varies in the dataset. Human images dataset: The human images dataset contains 13233 total human images which are sorted into 5750 folders. All images are of size 250x250 pixels.



ALGORITHM AND TECHNIQUES

Convolutional Neural Network architecture is the main algorithm used for this project, the first step was to use pre-trained models such as in the haarcascade library to classify human and dog images. But a CNN is much preferred for the classification task of dog breeds as the model has to learn complex image patterns using high-frequency filters or kernels that aids in convolving an image onto a convolutional layer. The main goal was to build a CNN model from scratch after pre-processing of the inputs into image tensors and then passing these inputs through convolutional layers, max pooling layers to reduce the dimensions of the images before passing it through linear fully connected layers and then output layer that predicts the probability of an image being of a certain class of dog breed.

BENCHMARK

The model built from scratch has a test accuracy of 11%, a test loss of 4.070189 and 96 correctly predicted out of 836 test cases.

DATA PREPROCESSING

All the images are resized to 224*224 pixels, re-centered, horizontally and rotated before being converted to image tensor values. The data augmentation is done to reduce overfitting and aid the generalisation and robustness of the trained model. The RGB values of the pixel images are also being normalised between ranges of 0 and 1. The closer the value of a pixel is to one, the lighter the pixel and 0 being a black pixel.

The image datasets are loaded in from the train, validation and test data directory using data loaders before the training dataset is being passed into the model for training.

IMPLEMENTATION

I opted for 32 filters in the first convolutional layer, 64 in the second layer and 128 in the third layer. This changes the input shape of the image which is (224,224,3) with RGB depth of 3 which is the first parameter passed into the constructor, stride value is 2 for the first and second convolutional layer and padding is 1 since I am using a 3x3 kernel filter for all 3 layers.

I have decided to go with the max pooling layer of stride =2 and filter =2 which reduces the dimensions of my images by a factor of 2 for every layer whilst keeping the depth value.

Getting to the first fully connected layer, the input is now given as (128 = number of filters in my last convolutional layer, 7 x 7 is the new dimensions of my (x,y) input after going through three max pooling layers). The max pooling layer is activated using the ReLu function as well as the fully connected linear layers. The dropout value is 0.25 which is applied to the linear layers. My final linear layer has 500 hidden layer nodes and 133 output classes.

REFINEMENT

The CNN created from scratch has an accuracy of 11%, although it meets the benchmarking score for the accuracy metric, the model can be improved by using transfer learning.

To build a transfer learning convolutional neural network, I have selected the Resnet50 architecture as it is computationally faster than the VGG16. We only need to add a fully connected layer to produce 133-dimensional output (one for each dog breed) as the trained resnet model on the imageNet data has 1000 output features.

The model performed extremely well when compared to built CNN from scratch. The test accuracy of the model is 81%, test loss of 0.604335 and 685/836 accurately predicted labels in the image test set.

MODEL EVALUATION AND VALIDATION

The model evaluation for both CNN networks using test accuracy are 11% and 81% respectively. The validation loss is used to select the best trained model, as we do not want an overfitting model with low training loss and high validation loss. The validation loss for the built from scratch CNN and resnet50 model is 4.081096 and 0.505251 respectively.

JUSTIFICATION

The transfer learning approach clearly solved the dog breed classification problem when compared to the built from scratch model with a improved test accuracy score of 70%.

Improvements on the project can be made by tuning the hyperparameters better such as learning rates, batch sizes and optimization algorithms.

A balanced dataset would also improve the accuracy of the model as all classes of dog breeds can have equal and a good amount of training examples.

REFERENCES

1. TY - JOUR,AU - Nash, Will,- A review of deep learning in the study of materials, 2018/12/01,npj Materials Degradation