Udacity Machine Learning Nanodegree 2020

Capstone Proposal

Convolutional Neural Networks

Dog Identification App

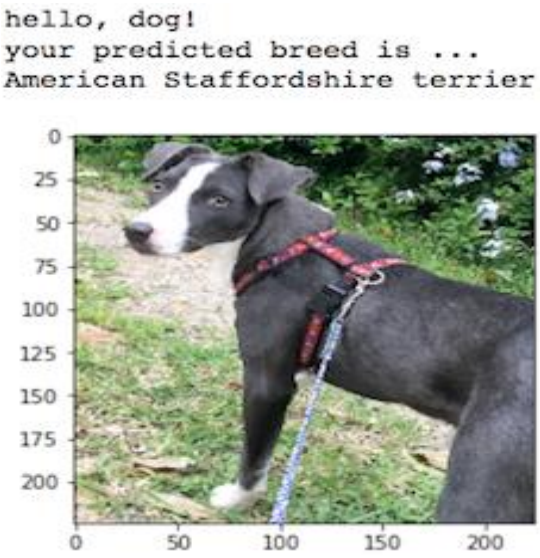
Name: Abhishek Harde

Date: June 2020

Project Definaton

1.Project Overview

Welcome to the Convolutional Neural Networks (CNN) project! In this project,I am going to build a pipeline to process real-world, user-supplied images. Given an image of a dog, My algorithm will identify an estimate of the canine's breed. If supplied an image of a human, the identitify the resembling dog breed.



Example dog breed

2.Project Domain

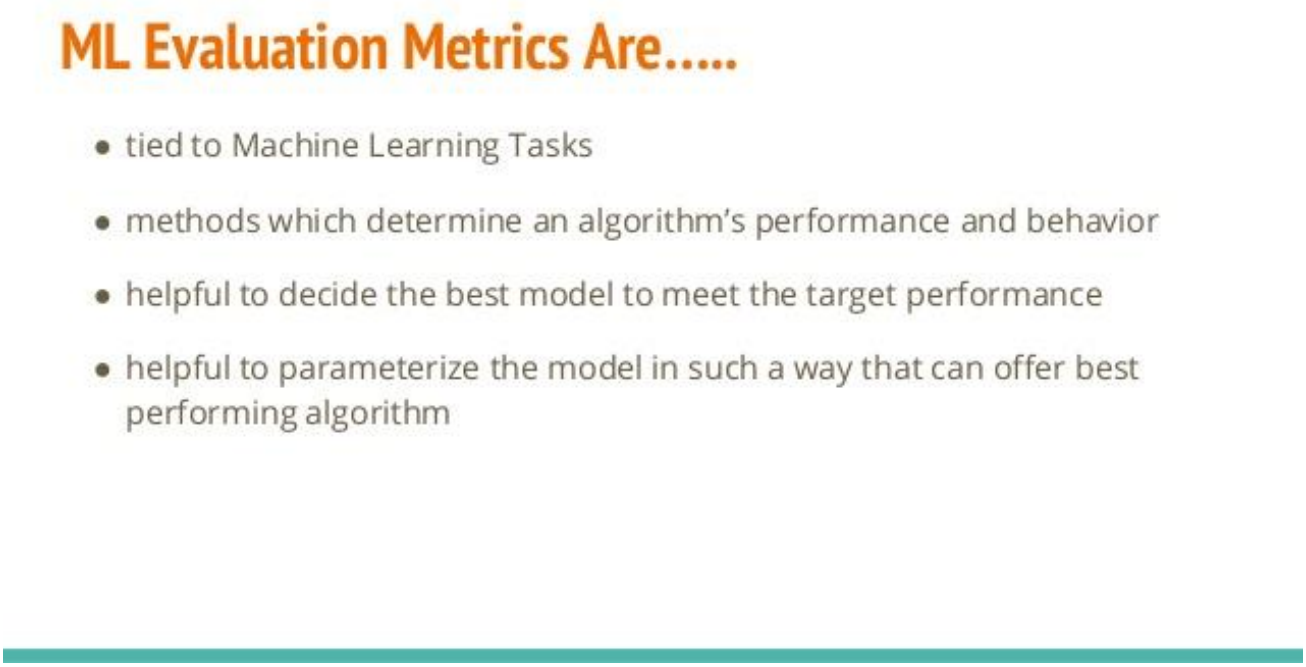
The project involve Image Processing using Deep learning and Convolutional Neural Networks along with Transfer Learning is used to deliver the results. Given an image of the dog, the CNN must give out the correct breed of the dog in the image out of 133 classes of dogs. All this functionality is provided using a web application which is developed using Flask.

3.Problem Statement

The aim of this project is to create a classifier that is able to identify a breed of a dog if given a photo or image as input. If the photo or image contains a human face, then the application will return the breed of dog that most resembles the person in the image. I decided to opt for this project as I found the topic of Deep Neural Networks to be very fascinating and wanted to dive deeper into this with some practical work.

4.Metrics

Evalution metrics are use to measure performance ,accuracy and efficiency of machine learning model.



Metric	Formula
True positive rate, recall	$\frac{TP}{TP+FN}$
False positive rate	$\frac{FP}{FP+TN}$
Precision	$\frac{TP}{TP+FP}$
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
F-measure	$\frac{2 \cdot precision \cdot recall}{precision + recall}$

In our model we will be using the accuracy as metrics of mesurement to evalute the performance of model.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Analysis

1.Data Exploration

In the data_set we have we have two data set one is dog data set and other is human data set. In dog data set we have images of 133 classes of dogs in 133 folders.

The dog data set have total 8351 images and human data set have 13233 images.

I find percentage of human images in first 100 images of dog datset is 17% and human data set is 98%.

I find percentage of dog images in first 100 image of dog data set is 100 and human data set is 1.

Then I split train and test data is 90%-10%, i.e. 90% for training and 10% for testing purposes. In the training data, we have reserved another 10% for validation. Total of 6680 images will be used to train our machine, to further fine-tune the parameters we use another 835

images for validating it. And, lastly, we will be using 836 images to test our model's performance for the evaluation of metric i.e. Accuracy.

2.Algorithms and Techniques

The classifier is a Convolutional Neural Network, which is the state-of-the-art algorithm for most image processing tasks, including classification. It needs a large amount of training data compared to other approaches.

In machine learning, Convolutional Neural Networks (CNN or ConvNet) are complex feed forward neural networks. CNNs are used for image classification and recognition because of its high accuracy.

❖ Neural network architecture

● Number of layers

● Layer types (convolutional, fully-connected, or pooling)

● Layer parameters

Methodology

1.Data preprocessing

The image we given as input to the model process the image before passing it to the network. In this step, the image is normalized, resize, crop the image to speed up the training. Then the image is converted into tensors. In this step, there are some parameters we can choose, like the parameter for resizing, center crop, and normalize methods.

2.Implementation

1.I have defined CNN architecture with batchnorm and dropouts to prevent overfitting

2. I created an architecture which has 5 Convolution layers. This is beacuse in theory the first convolution layer will detect edges for different dog images and the last four should be able to detect more complex details which should allow the model to distinguish between different and similar dog breeds, and because of the small size of the data set, I chose to make the model deeper for better detection of details for distinguishing the breeds of dogs.

3.The convolution layers each, are followed by MaxPooling layers for dimesionality reduction and more complex pattern detection.

4.The final Convolution layer is followed by a flattening action in the forward function and three fully connected layers with dropout to avoid overfitting. ReLU activation is used in order to identify classes greater than 0.

The final Fully Connected layer has 133 as its final parameter matching the 133 classes of dogs in the dataset.

```
class Net(nn.Module):
    ### TODO: choose an architecture, and complete the class
    def __init__(self):
        super(Net, self).__init__()
        ## Define layers of a CNN
        self.conv1 = nn.Conv2d(3, 16, 3, padding=1)
        # convolutional layer (sees 16x16x16 tensor)
        self.conv2 = nn.Conv2d(16, 32, 3, padding=1)

        # convolutional layer (sees 8x8x32 tensor)
        self.conv3 = nn.Conv2d(32, 64, 3, padding=1)

        # max pooling layer
        self.pool = nn.MaxPool2d(2, 2)
        # linear layer (64 * 28 * 28 -> 500)
        self.fc1 = nn.Linear(64 * 28 * 28, 256)
        # linear layer (500 -> 133)
        self.fc2 = nn.Linear(256, 133)
        # dropout layer (p=0.25)
        self.dropout = nn.Dropout(0.25)
        self.batch_norm = nn.BatchNorm1d(num_features=256)

    def forward(self, x):
        ## Define forward behavior
        x = self.pool(F.relu(self.conv1(x)))

        # add dropout layer
        x = self.dropout(x)

        x = self.pool(F.relu(self.conv2(x)))

        # add dropout layer
        x = self.dropout(x)

        x = self.pool(F.relu(self.conv3(x)))

        # add dropout layer
        x = self.dropout(x)

        # flatten image input
        # 64 * 28 * 28
        # x = x.view(-1, 64 * 28 * 28)
        x = x.view(x.size(0), -1)
```

Model Architecture

2.Model Evalution and Validation

During development, a validation set was used to evaluate the model.

First I tried the VGG16 architecture and got accuracy of 63% wich is not that good.

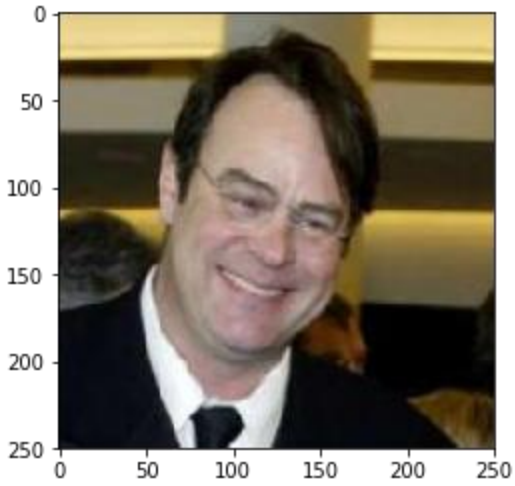
Then trained the model with resnet101 wich gave me an accuracy of

86% wich is much better tha previous build models.

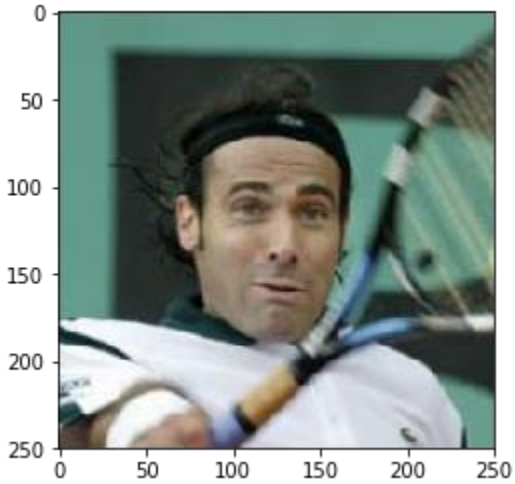
The validation set is used for hyperparameter tuning and the test set is used to test the performance of the model. After completing the training, and testing of the model I got an accuracy of 81% At last, I used 6 images to test the model and found that the model correctly classified the images .

At last, I used 6 images to test the model and found that the model correctly classified the images.

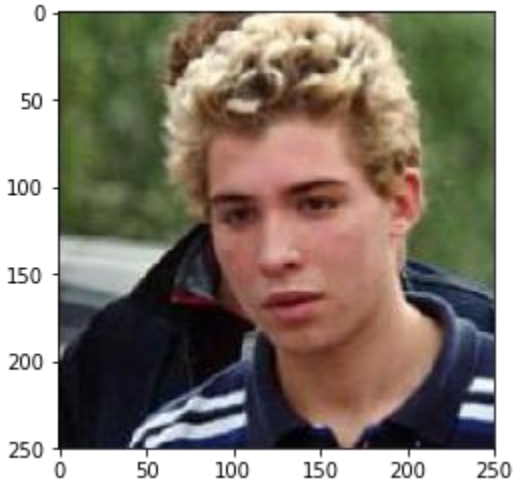
```
face is detected
```



This person if look like "Dachshund"face is detected

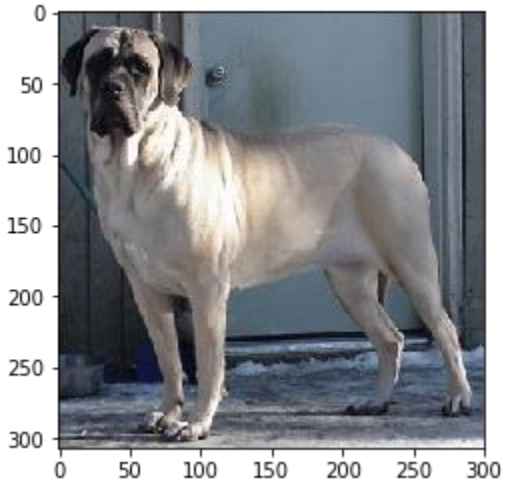


This person if look like "Dogue de bordeaux"face is detected

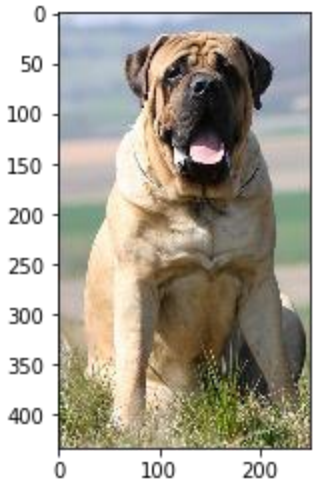


This person if look like "Cane corso"dog is detected

This dog is "Bullmastiff" breeddog is detected



This dog is "Bullmastiff" breeddog is detected




```
This dog is "Bullmastiff" breed
```

3.Result

In the implementation phase of the pre-trained model VGG 16, we test the prediction of the model and we also observe that the results show the right prediction value of the dog. On implementing the custom Convolutional Neural Network model, we construct out the relative stack of layers required for our purpose and tune the necessary hyperparameters of the model.

On testing the performance of the custom model through the metric evaluation i.e. accuracy in our case, we observe that the results give us a result of 15% in terms of accuracy. The project had the condition to achieve the results of more than 10% accuracy.

After the implementation and training of the transfer learning model for a total of 20 epochs, we need to evaluate the performance of the given test data of dog images so that the model should predict the breed of the dog with utmost accuracy. Therefore, getting the model ready for evaluating the performance on the test data, we used the test function that was previously defined for the custom model and evaluated the performance of the transfer learning model. We see that the model achieved an accuracy of 81%, and we see that there is a considerable rise in the performance of the model after combining the predefined model with our custom model. If we compare the results with our benchmarking models(VGG19: 83.9%, InceptionV3; 79.9%, Xception: 84.33%), we can see that our model scored the comparable but slightly lesser, but keeping in mind that we only used 20 epochs. It doesn't mean that our model is bad, with proper adjustments the accuracy of our model can be increased.

Conclusion

1.Reflection

The process used for this project can be summarized using the following steps:

● An initial problem and relevant, public datasets were found

● The data was downloaded and preprocessed

● The classifier was trained using the data (multiple times, until a good set of parameters, were found)

Out of these steps, construction of CNN was little hard but done succesfully after lots of try.Traning the model also take about 2-3 hours. I did this project on udacity workspace so it reduced my task to download dataset.

2.Improvement

Model I created performed well and give good prediction but it can be improved more by following steps

1.Increasing the number of convolutional layers

2.Adding more dropout layers

3.Hypertuning the model(To improve CNN model performance, we can tune parameters like epochs, learning rate etc.. Number of epochs definitely affect the performance. For large number of epochs , there is improvement in performance. But need to do certain experimentation for deciding epochs, learning rate. We can see after certain epochs there is not any reduction is training loss and improvement in training accuracy. Accordingly we can decide number of epochs. Also we can use dropout layer in the CNN model. As per the application, need to decide proper optimizer during compilation of model. We can use various optimizer e.g SGD,rmsprop etc. There is need to tune model with various optimizers . All these things affect the performance of CNN.)

4. Image Data Augmentation(Image augmentation parameters that are generally used to increase the data sample count are zoom, shear, rotation, preprocessing function and so on. Usage of these parameters results in generation of images having these attributes during training of Deep Learning model. Image samples generated using image augmentation, in general existing data samples increased by the rate of nearly 3x to 4x times.)

References:

https://medium.com/@dipti.rohan.pawar/improving-performance-of-convolutional-neural-network-2ecfe0207de7