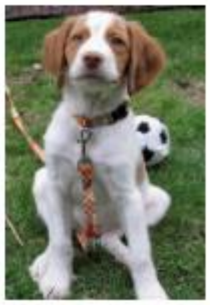
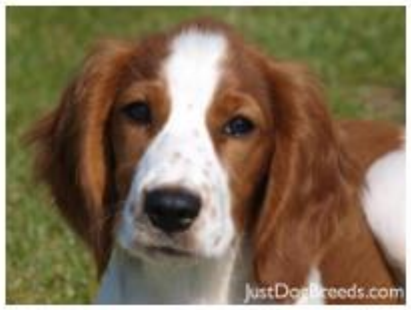
CNN Project: Dog Breed Classification

Project Overview

There are different canine breeds in the world humans wish to identify. With a large range of variety of breeds, it is difficult to point out what the breed of the dog. With machine looking and finding pattern of images of dogs and finding the commonality of the different breeds there are, we will be able to identify the breed having to input an image of a dog. Deep learning is a known technology in machine learning when finding patterns, especially on images. Face detection for instance is a trend on smart phones and security. Convolution Neural Network is a deep learning algorithm which assigns importance to various aspects in the images to differentiate it to another image with enough training ConvNets can learn the characteristics images. ConvNets architecture is patterned to the neurons on human brain. Each neuron corresponds to a stimulus which is a restricted region of the visual field. A collection of visual fields overlaps to cover the entire visual area or the image. Which is why CNN is effective on image classification.

Problem Statement

The goal is for to build an algorithm will identify an estimate of the canine’s breed. Assigning breeds to dogs is challenging that even humans have a hard time distinguishing dogs on the same family. For instance, a Brittany and a Welsh Springer Spaniel.

Brittany	Welsh Springer Spaniel
	

These dogs that have a minimal inter-class variation is difficult to identify but we want to train our model to be able to identify the difference. Moreover, Labradors for example have different colors yellow, black, or brown.

Data Exploration

1. Human Dataset
    a. There are total of 13233 human images
    b. Each human is sorted by human names
    c. Each human name has different numbers of photos
    d. Images are in different backgrounds
    e. All images are of size 250 x 250
    f. Data is not balance

2. Dog Dataset
    a. there are 8351 total dog images
    b. 6680 are used for training the model
    c. 836 are used for testing
    d. 835 are used for validation
    e. Images are in different sizes with different backgrounds
    f. Images are imbalance since the number of images for each breed varies.

Algorithm

The solution to this problem is to build a model using Convolutional Neural Network (CNN). CNN or ConvNet is an artificial neural network that is popular for classifying images. This algorithm has a special feature that can detect pattern and make sense of these patterns. Its hidden layer called convolutional layer receives input and transforms these to the output layer called convolutional operation.

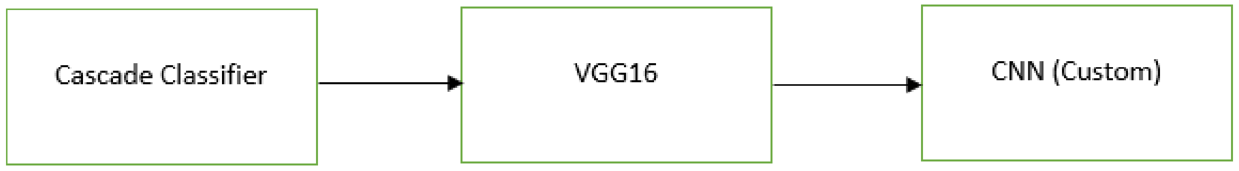


Figure A: Algorithm

Figure A shows the steps the data will go through. First is to detect humans on the picture using the OpenCV's implementation of Haar feature-based cascade classifiers. Then we check if there exists a dog in the image using pre-trained VGG-16 Model. Lastly, the image will be processed by CNN to identify the breed of the dog.

Implementation

Metrics

The metrics to be used to check if the model performed well or not is Cross Entropy Loss which is the uncertainty of prediction based on how much it varies from the actual label. Since there is an imbalance of output classes for this data, so using accuracy score is not a good idea. Although accuracy is shown in the model evaluation which is computed as follow:

$$accuracy = \frac{number\ of\ correctly\ classified}{number\ of\ classified\ items}$$

Instead using F-score to measure the model performance which is the weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. It can be computed as follow:

$$F1\ Score = 2 * \frac{precision * \ recall}{precision + recall}$$

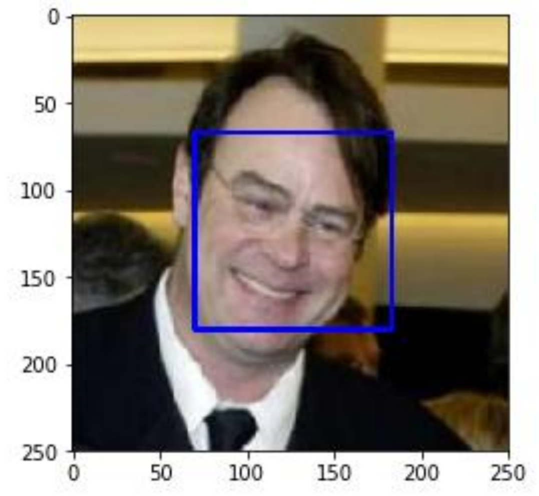
Data Processing

Since VGG16 accepts a size of (244,244) both valid and test data was resized to this size. Train data was also randomly resized to 244 with random rotation to avoid overfitting. All datasets were normalized to be accepted as input too.

Human Detection

OpenCV provides many pre-trained face detectors, stored as XML files on github. I have used the haar cascade classifier. In order to implement such, the image was first converted to grayscale. The grayscale image was sent to the classifier and will return an output of dimensions where the face is detected.

```
Number of faces detected: 1
```



Dog Detection

For implementing the dog detection model, I used a pretrained VGG-16 model. All pre-trained models expect input images normalized in the same way, i.e. mini-batches of 3-channel RGB images of shape (3 x H x W), where H and W are expected to be at least 224. The images must be loaded into a range of [0, 1] and then normalized using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]. So, I transformed the images to that before passing it to the model for detection. The model output several possible results, by getting the result that has the highest percentage of confidence and returning True when checking whether that index falls under the dog category.

CNN from Scratch

For the CNN without transfer learning, there are three sequential layer objects. This method allows the creation of convolutional with applied ReLU function and pooling sequence layer. Inside a sequential layer there is a Conv2d which creates the convolutional filters with channels from 3 to 32 at the first layer. For kernel size of the filter I set it to 5 x 5 and a padding of 2 x 2. To down-sample the data, reducing the effective image size by a factor of 2. To do this, using the formula above, we set the stride to 2 and the padding to zero for the MaxPool2d. To avoid overfitting the model, a dropout layer is added and finally two fully connected layers are created that are Linear model.

Refinement

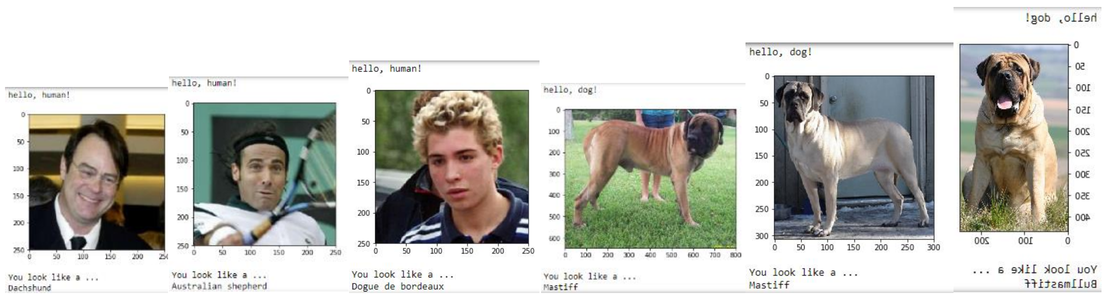
When building the CNN with transfer learning, I first loaded a pretrained model resnet18 and reset the finally fully connected layer having several filters of 512 input features and 133 output features. Since it takes more than 2-3 weeks to train modern ConvNets I used a fine-tuning method for the CNN to fine-tune the weights of the pretrained network by continuing the backpropagation.

Postprocessing

To implement the algorithm the function takes an argument of the image path and first determines whether the image contains a human, dog, or neither. Then, if a dog is detected in the image, return the predicted breed, if a human is detected in the image, return the resembling dog breed, and if neither is detected in the image, provide output that indicates an error. When predicting the dog breed, the CNN using transfer learning was used. When a human face is detected using OpenCV’s Haar feature-based cascade classifier was used, it proceeds with checking if a dog exists in the photo using the pretrained VGG-16 model. If both models produced an output of no detected face or dog the algorithm returns an invalid image note.

Model Evaluation and Results

1. Human Face Detector
    a. Face detected on human files 98%
    b. Face detected on dog files 17%
2. Dog Detector
    a. Face detected on human files 0%
    b. Face detected on dog files 100%
3. CNN from scratch
    a. With 15 epochs the metrics are the following:
        i. F1-Score: 0.137210
        ii. Test Loss: 3.782419
        iii. Test Accuracy: 13% (115/836)
4. CNN using transfer learning
    a. With 15 epochs the metrics are the following:
        i. F1-Score: 0.788069
        ii. Test Loss: 0.710884
        iii. Test Accuracy: 79% (656/836)
5. Results of the Algorithm



```
hello, human!
You look like a ...
Dachshund

hello, human!
You look like a ...
Australian shepherd

hello, human!
You look like a ...
Dogue de bordeaux

hello, dog!
You look like a ...
Mastiff

hello, dog!
You look like a ...
Mastiff

hello, dog!
You look like a ...
Bullmastiff
```

Conclusion and Recommendations

In conclusion the implementation of the algorithm works and was able to classify the breed of the dog with an accuracy of 78% which is not high but is also not bad. As compared to the algorithm that was built on scratch which has an accuracy of 13% with test loss of 3.83 the model using transfer learning is significantly better. The test loss of the algorithm is at 0.7266 which can still be lowered when the number of epochs is increase. In order to further improve the algorithm using exact feature extractor instead of fine-tuning, using a different algorithm for implementing either feature extractor or fine-tuning since in this project I used ResNet101, and augmenting the images for training to improve the model on finding pattern with different augmentations and angles of images .