Capstone project

Dog breed classification

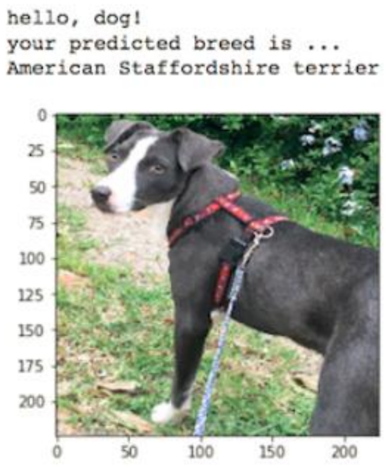
Author: Matija Varga

Date: December 6th 2020

1. Project definition

Introduction

This project will result in an application within a Jupyter Notebook environment that will classify images. If there is a dog in the image, it will identify its breed. If there is a human in the image, the application will say to which breed does it resemble. If there is neither a dog nor a human, the application will report that no dog or human could be detected in the image. Input data is provided by Udacity and it contains images of dog and humans. Here is one example how the program could work:



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

Figure 1: Example output of the final application.

Domain background

The problem of classification of images belongs to the field of deep learning and here we will tackle it using convolutional neural networks (CNNs). There are two distinctive parts of such a network: convolutional and fully connected neural part. The convolutional part decomposes and again recombines parts of an image through several layers. The output of the convolution is an input for the fully connected neural layer. The parameters of the convolutional part (e.g. values of the filter kernels) and parameters of the fully connected neural part (e.g. weights) are obtained in a training procedure (backpropagation and gradient descent using an optimizer). An overview and further literature on convolutional neural networks can be found in a review by Lecun et al.[1]

Problem statement

The problem consists of the following tasks: 1) finding a human face in an image, 2) finding the most similar breed to that of the human. If no human faces are detected, it is necessary to determine if there is a dog in that image. If there is neither a dog nor a human in the image, a message should be displayed. The problem of dog breed classification is challenging due to the wide variety of classes in the classification problem (>100). In addition to that, the within-class variance is large (e.g. different positions of dogs in an image) and between-class variance is small (e.g. different colors of Labradors).

Evaluation metrics

We use accuracy as a performance metric for assessing both the simple model (benchmark) and the final model (dog breed classifier). For assessing the performance of human face detection we test the classifier on both human and dog images and report the accuracy. In that test, the accuracy on dog images can be considered as a true negative rate.

Project design

The project is divided in six phases:

1. Detecting human faces in images.

2. Detecting dogs in images using a pre-trained model such as VGG16.

3. Creating a simple (benchmark) classifier of dog breeds from scratch.

4. Creating the final dog breed classifier using transfer learning on a pre-trained CNN.

5. Writing an application according to the problem statement.

6. Testing the application in Jupyter Notebook with the provided and own images.

[1] Yann Lecun et. al. Nature 2015, Figure 2.: link

2. Analysis

Datasets and inputs

Dog and human datasets are provided by the organizer of this course. They consist of 13233 human images and 8351 dog images. Images have three color channels, different sizes and aspect ratios. Potential problems might arise from different aspect ratios of dog images and the fact that some pre-trained models only work with square images, i.e. the cropping might result in loss of relevant information (e.g. head of a dog, Figure 2).

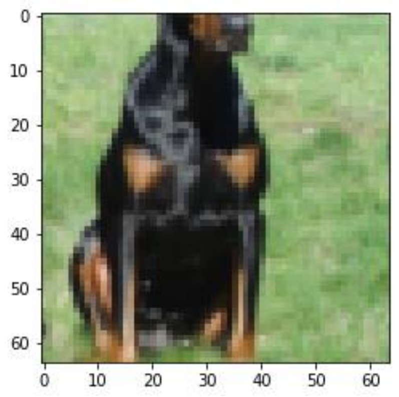


Figure 2: Example how a transformation of cropping (1:1 aspect ratio) might result in a loss of data.

The dataset is not heavily imbalanced with most dog breeds having 30-50 samples. The following image shows the distribution of classes in dog breed dataset:

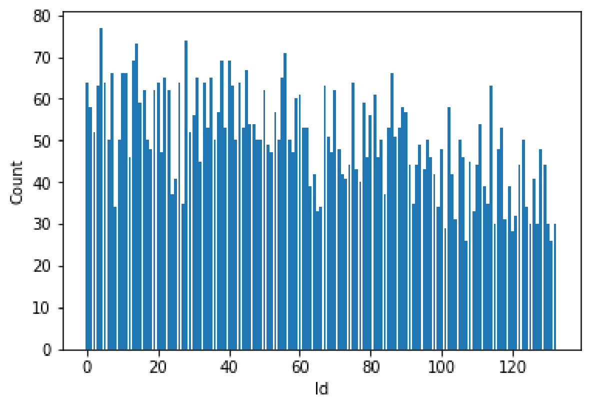


Figure 3: Distribution of dog breed (classes) within the dataset. Courtesy of [2]

[2] Jing Xian Lin: link

After exploring the dataset, one can notice images with both owners and dogs that are saved under “dog images”. Figure 4 depicts one such example. Section Results elaborates further on this challenge.

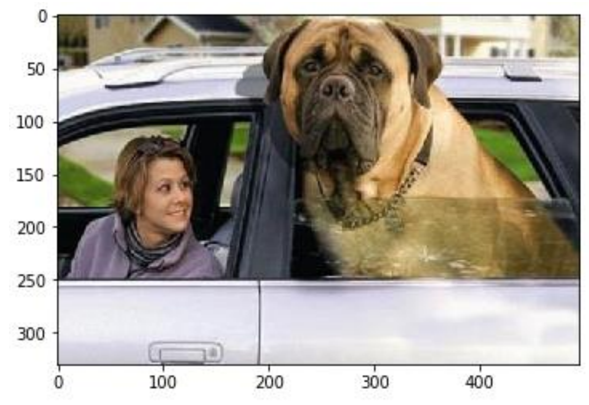


Figure 4: Example of “dog image” with a human in the image.

Images are split in train, validation and test sets and stored in different folders. The validation dataset is used after each epoch to determine whether there is an improvement in the model’s performance and whether it should be saved. Finally, the performance of the dataset is reported on a test set.

Algorithms and techniques

For the problem of dog breed classification we use large networks with many convolutional layers (>10). To speed up the training procedure (with resources provided within the MLE course), we will use an existing human face detector (Haar feature-based cascade[3]) and pre-trained networks for image classification. The solution is presented in three steps. The first step is to quantify the accuracy of a human face detector on a dataset of dog and human images. Then, another classifier is built that should determine a dog breed. Quantitative assessment (accuracy) of the dog classifier is obtained on a test set with dogs. The last step is to build an app which fulfils the requirements in the *Problem statement*. The reasoning behind using the above workflow is partially determined by the outline of the project provided by Udacity.

Finally, the application should follow this algorithm:

[3] P. Viola, M. Jones, Conf. on Computer Vision and Pattern Recognition, 2001, link

If human is detected then:

- greet human,
- display image,
- find to which dog breed does the human resemble.

else, if dog is detected:

- greet the dog,
- display image,
- determine its breed.

else:

- display that this is neither a dog nor a human,
- display image.

Benchmark model

As a benchmark model (*model_scratch*) for the most challenging part – the dog breed classification, we choose a simple convolutional network with 3 convolutional layers and 2 fully connected layers. The goal is to set a benchmark on the lowest value of accuracy and also to understand how complex should the final classifier be. It is expected that a simple benchmark classifier will have an accuracy greater than 10%. The final classifier should achieve accuracy higher than 60%.

3. Methodology

Data Preprocessing

We employ slightly different preprocessing methods for different models. For models based on a pre-trained network (VGG16[4]) we preprocess the data so that it matches the expected format (224x224) and scaling of pixel values (mean = [0.485, 0.456, 0.406], standard deviation=[0.229, 0.224, 0.225]). For the simple model (benchmark) we set the format to 64x64 to have a more lightweight first convolution. We set the mean and standard deviation of all image channels to

[4] K. Simonyan and A. Zisserman, arXiv:1409.1556

0.5. The resolution of 64x64 pixels is the smallest resolution at which we could visually identify a dog breed. To augment the training set we perform a randomized resize-crop as well as a horizontal flip. In the validation and test sets, we perform a resize operation and a centered crop.

Implementation

The project is split in stages (see Project definition) and each stage implements one part of the final application. There are three parts of the final application:

1. human detector,
2. dog detector,
3. dog breed classifier.

We use OpenCV library *cv2* and a pre-trained detector (*haarcascade_frontalface_alt.xml*) for detecting human faces. A pre-trained network (VGG16) is employed in a *dog_detector* method which returns *True* if a dog is identified in an image and *False* otherwise. The training of the simple CNN (benchmark) for dog breed classification is conducted using Pytorch library *torchvision*. As a starting model, we use an example of CNN from Udacity's github profile[5]. We adapt layers to the fit dog-breed classification problem: increase the dimensions of input data and increase the number of neurons in both fully connected layers. We have left the dropout layers unchanged because we find them appropriate for such a problem and it will helps us not to overfit to the training data. The activation function and the maxpool operation seem sensible, so we also keep them unchanged.

The transfer learning is performed also with Pytorch model VGG16 and *torchvision*. We have decided to use this pretrained network because it has demonstrated high accuracy when detecting dogs. The fully connected layers need to be retrained for our target application (dog breed classification), whereas the "feature-extracting" part (convolutional layer) is powerful enough and thus we decide to keep the weights of the convolutional kernels frozen during the training. The weights of the convolutional part are fixed using the following command:

```
for param in model_transfer.features.parameters():

    param.require_grad = False
```

[5] https://github.com/udacity/deep-learning-v2-pytorch/blob/master/convolutional-neural-networks/cifar-cnn/cifar10_cnn_solution.ipynb

The last fully connected layer is modified to match the number of classes in our problem (133). In both models, benchmark and transfer learning, we have used cross entropy loss as a metric that was forwarded to the stochastic gradient descent optimizer.

During the development there have been little refinements since almost all of the components have satisfied the requirements after the first build. The only component that required further optimization was the benchmark model for dog-breed classification. It was necessary to increase the size of its first fully connected layer so that it can capture the variety which is present in the training dataset.

4. Results

Face detector

In 17% of dog images the algorithm has identified human face(s). The algorithm detects human faces in 98% of all human images. We have observed that in some dog images there are human faces, so the higher rate of human faces in dog images is not only due to the errors in the face detection algorithm. Figure 5 shows two examples of “dog images” where a human was detected (false positive) – one with a human and one without.

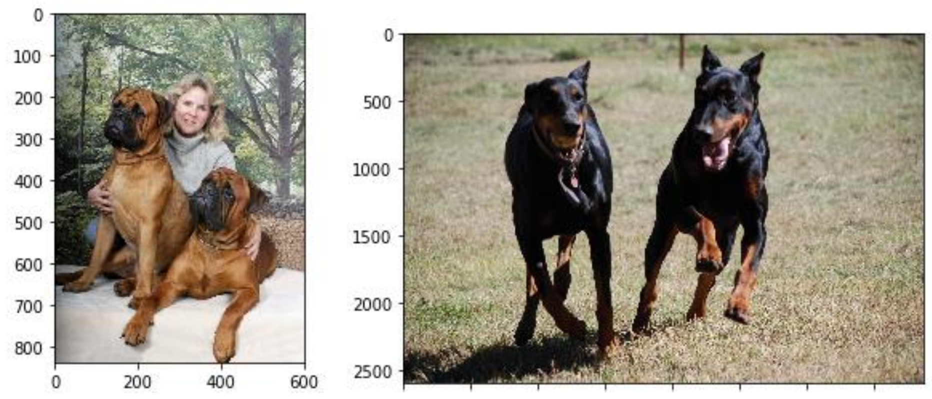


Figure 5: Examples of dog images where a human face was detected.

Dog detector

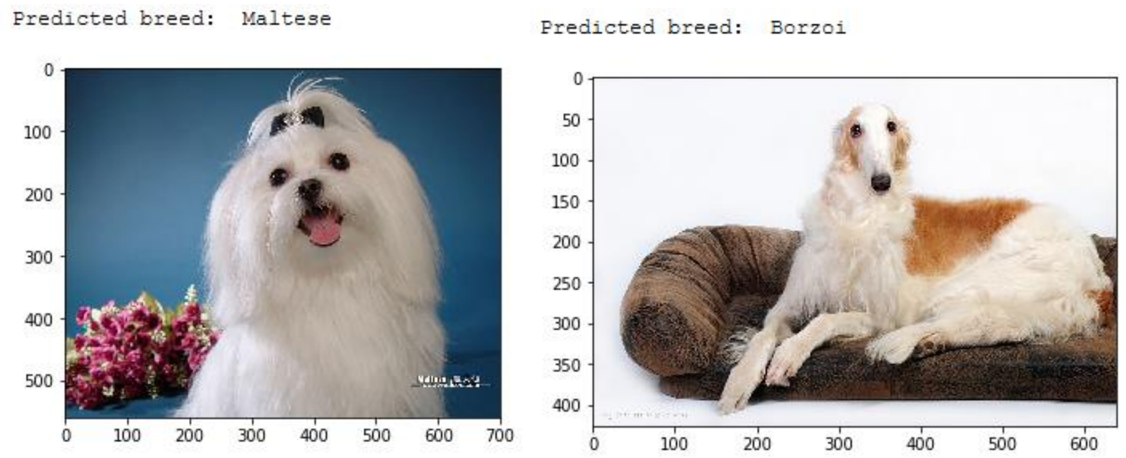
All images in the first 100 dog images have been detected as dogs and zero as humans. In contrast to the face detector, we have a higher accuracy because this detector is based on a large pre-trained model VGG16.

Dog breed classifier – benchmark

This simple model achieves 14% accuracy on a test set. It indicates that a much larger neural network (with more convolutional layers and more neurons) is necessary to classify 133 dog breeds.

Dog breed classifier – transfer learning

The transfer learning model achieves 82% accuracy on the test set. This is a large improvement compared to the benchmark model.



```
Predicted breed:  Maltese
```

```
Predicted breed:  Borzoi
```

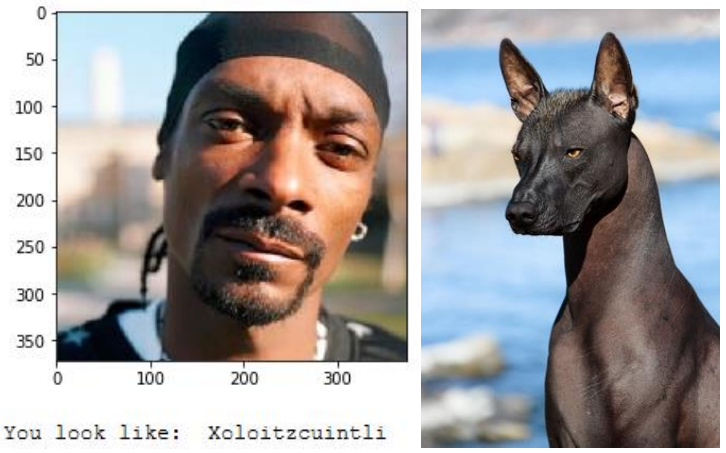
Figure 6: Output of the dog breed classifier based on transfer learning.

Final application

Here we display some of our own test images that are forwarded to the final application (Figure 7-Figure 10). We test the final application on the following images:

- Golden retriever: a distinctive dog breed that should be classified easily.
- Snoop Dogg: a famous actor whose name suggest that he is a “dogg”, but is he really?
- Dachsund with a human face: an image for testing the face detector.
- Lion: image of a fur animal that is neither a human nor a dog.
- Author of this work: a regular human.
- Human and dog: testing the performance of the face detector or the dog breed classifier.

```
Hi human!
```



```
You look like:  Xoloitzcuintli
```

Figure 7: Output of the application for an image of Snoop Dogg (left). Image of the predicted dog breed is given as a reference (right).

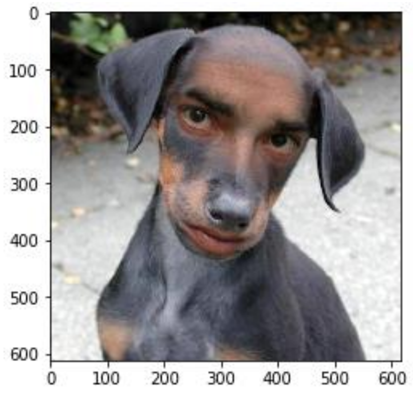
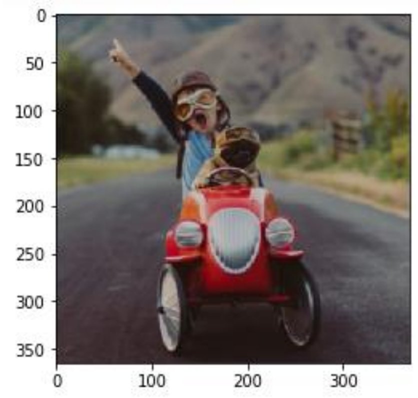


Figure 8: Output of the application for an image of a dog with a human face.



```
You are neither a dog nor a human!
```

Figure 9: Output of the application for an image with both human and a dog.

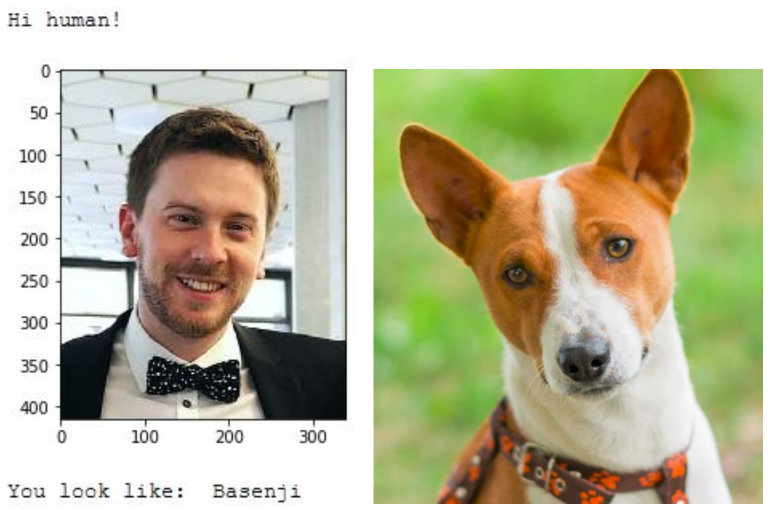

```
Hi human!
You look like:  Basenji
```

Figure 10: Output of the application for an image of author of this project (left). Image of the actual dog breed is given as a reference (right).

5. Conclusion

Compared to the performance of the benchmark ("quick-and-dirty") classifier, the transfer learning classifier performs better than expected. Possible improvement points of the project are:

1) The algorithm for detection of humans does not look for both a dog and a human (face). It could happen that a dog and its owner are in the same picture (few cases observed during dataset exploration). The algorithm would detect this image as a human and then determine the similarity using the dog-breed classifier. The classifier would most probably output the breed of the dog, rather than say something about the human’s resemblance with a certain dog breed.

2) The algorithm could use the boundary of a human face (from Haar's detector) to make sure that it is working with a human face when providing the resembling dog breed.

3) The training data set could be augmented with more image transformations such as stretching or rotations.