Capstone Project

Machine Learning Engineer Nanodegree

Mohamed Reda

March 20th, 2021

Dog Breed Classifier

Definition

This report contains the capstone project of the Machine Learning Engineer nanodegree about creating a dog breed classifier using a Convolutional Neural Network (CNN).

Project Overview

Today there are millions of dog owners throughout the world. The European Union has an estimated dog population of 85 million[1]. With over 300 breeds of dogs , many dog owners might not be aware of the different breeds in existence or the breed of their own dog. The importance of breed awareness could arise when considering a buying a dog, toys, food or medical visits to the local veterinary physician. Therefore, each owner should at least know the breed(s) of their dog(s). There can be a web app that returns estimated breed of a dog given a user-supplied image of a dog or human (if people want to know which breed they look like).

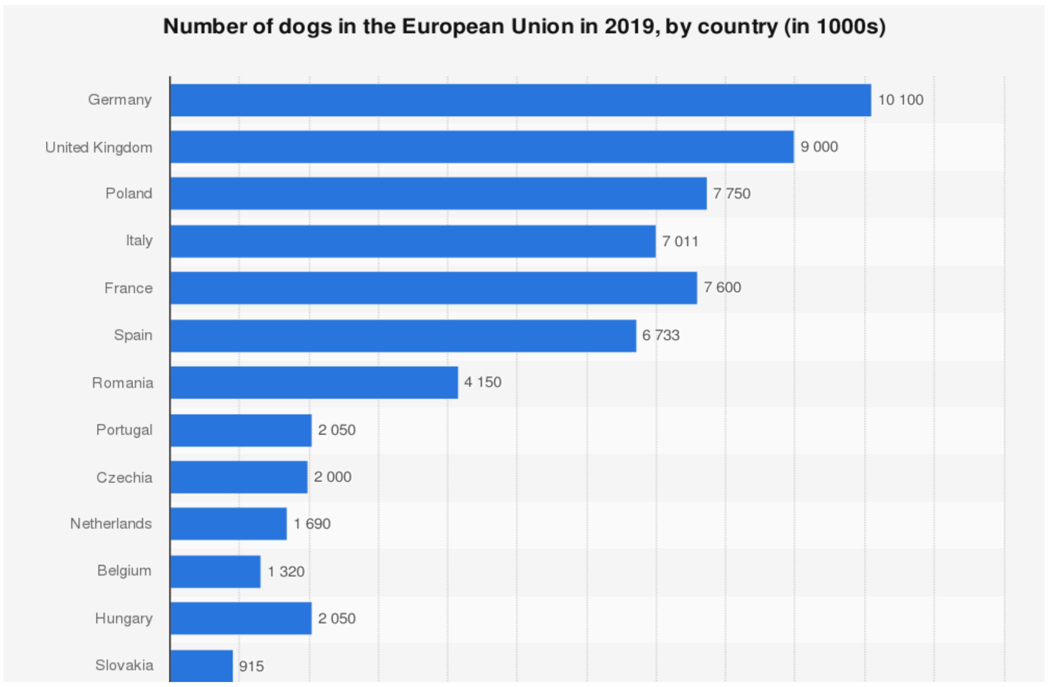


Figure 1:statistic_number-of-dogs-in-the-european-union-2019-by-country

This project is forked from Udacity and the source can be found here.
For our project, the classification is absolutely based on images of the dogs and the goal defined for this is to classify 133 dog breeds using state of art deep learning algorithms to teach and train the computer how to give an estimation of a particular dog breed from a dog image provided.

Problem Statement

The primary goal of this project is to do the following tasks using CNN:

- Given an image of the dog, the model should predict its breed
- If human is detected in image, the model should predict resembling dog breed

To implement the solution, we will create 3 different model. The first model will classify human face present in the image using the HaarCascade algorithm.
The second model will be a CNN custom model to classify dog in the image. This model is created from scratch defining the number of convolution layer, pooling, and batch normalization within it. The third model will be based on transfer learning using Resnet50 algorithm to improve over model and compare it with the earlier created models. Resnet50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.
The main aim of this project is to create a robust classifier that can successfully determine breed of the dog image. We will be creating a pipeline where, given an image of a dog, the algorithm will identify the breed of the dog. If there exists a human in the image, the code will return the resembling dog breed.

Metrics

For the evaluation the labelled data is used. For example, the accuracy of the face detection algorithm can be measured by feeding it with (previously labelled) images of humans:

$$\text{Accuracy Score} = \frac{Human\ detected}{images\ provided}$$

A slightly modified algorithm is used to evaluate the dog detecting model.

The dog detector provides a breed identification on top, so in the first step the upper term can be
used to judge the accuracy of the dog detection. Additionally the quality of the breed identification is measured as follows:

$$\text{Accuracy Score} = \sum_{dog\ breeds} \frac{correct\ breed\ detected}{dog\ images\ provided}$$

Analysis

Data Exploration

The data consists of two different sets, provided by Udacity are described as follows:
Dog dataset is already divided into three different datasets (folders) for

- test

- training

- validation

Each folder contains sub directories of the specific formating XXX.breed name, where XXX represents a 3 digit number, followed by the breed name after a dot. Each subfolder contains a hand full of images of the specific breed. 1Overall there are **133** different dog breeds, and **835** images provided for validation, 6680 images for training and **836** images for testing. The images within the training dataset are unbalanced, the amount of images per breed varies from 26 for the **Norwegian Buhund** and the **Xoloitzcuintli** to 77 for the Alaskan Malamute.
So, some breeds are represented roughly 3 times more often. The dog images contain a singular dog each, mostly of the whole dog, sometimes only of the snout, but they are not equally sized, they vary from 5 kB up to 5 MB. Their aspect ratios cover all ranges between portrait, landscape and quadratic.

Human dataset consists of **13233** images of 5749 **persons,** which are stored in a separate directory named after each celebrity. The images of the people are already cropped and centered around their face and all of the same size of 250x250, but the backgrounds vary. Sometimes there are additional people in the background.

Exploratory Visualization

Also. Here you can see this sample of dog dataset to be in sight how is the data looks like:

And the human dataset looks like this sample:

Algorithms and Techniques

This problem of the dog breed identification will be solved using a convolutional neural network. These networks are especially good on solving the tasks of image recognition, within a 2-dimensional space. Whereas classical neural networks (NN) depend on a flat input vector and usually fully connect the different layers, in a convolutional network many different subsets of images are analyzed with the help of a kernel. The kernel can be thought of as a kind of filter, which extracts the information that pixels have in relation to one another, like for example a horizontal line. This extracted information than is passed on to another layer and can be abstracted even more. As In figure below is a short brief how kernels use methods to extract features.

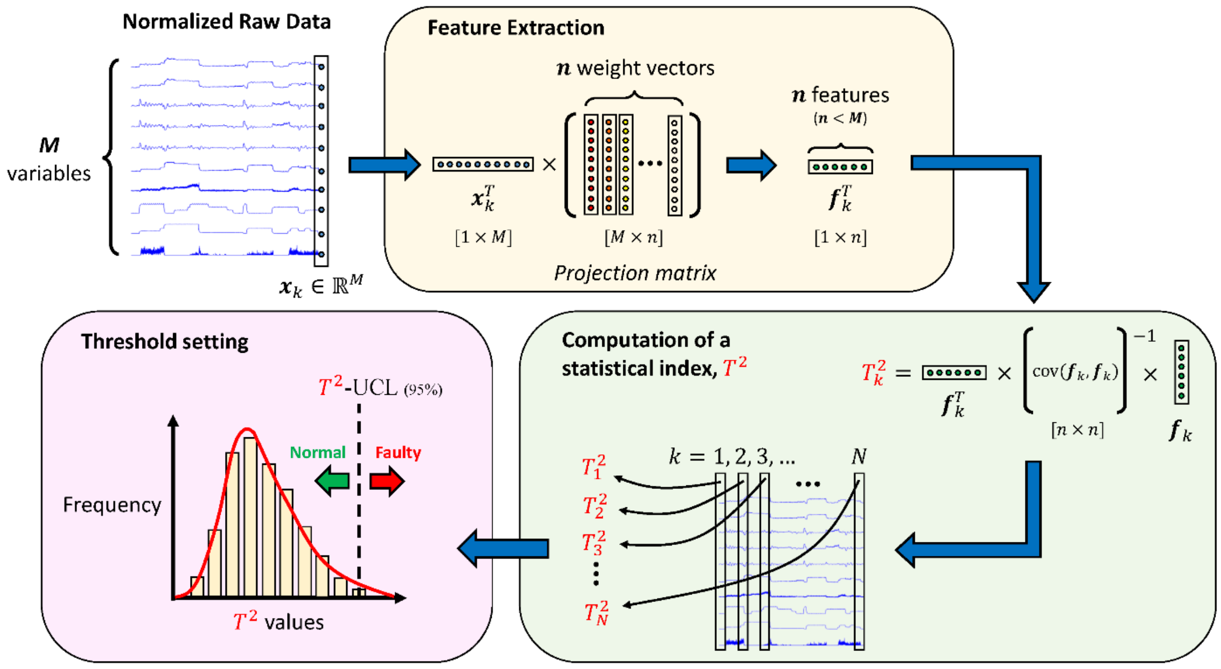


Figure 2: kernel methods of feature extraction

As for this problem in a brief There are 7 main steps drive me to the solution of the project as follows:

- Step 1: Explore and check the datasets in order to understand how to use them and choose the proper algorithms for it.
- Step 2: Then Implementing a Haar feature-based cascade classifier by using OpenCV in order to detect faces in the human dataset provided.
- Step 3: Then Using a pre-trained VGG16 model so that it can detect dogs in the dogs-dataset provided by Udacity.
- Step 4: Use transfer learning to create a CNN to classify dog breed in order to classify the 133 dogs breeds and have an accuracy more than 10% as required.

- Step 5: Use the transfer learning technique in order to get a pre-trained VGG-16 architecture and then continue the training with the dogs dataset provided. The minimum required accuracy is 60% in the test set given.
- Step 6: Write a custom algorithm that accepts a file path for the image and first of all it will detect whether the image provided contains a human, dog, or neither.
  Step 7: Testing the Algorithm with some random images user-supplied.

Benchmark

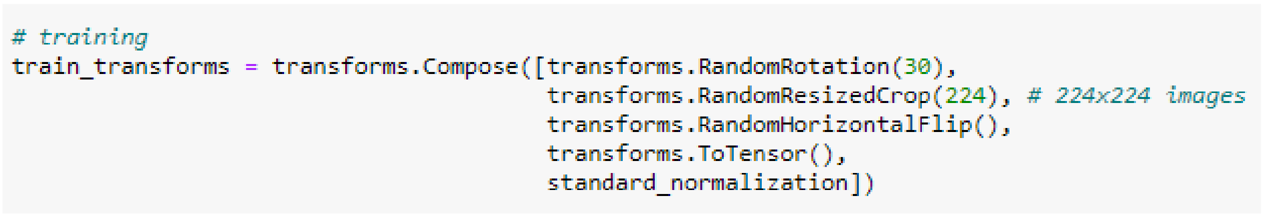
Pre-trained VGG-16 model can be a benchmark model. VGG-16 model, with weights that have been trained on ImageNet (a well-known very large and popular dataset used for image classification and other computer vision problems). Although The CNN model created from scratch must have accuracy of at least 10%. The CNN model created using transfer learning must have accuracy of 60% and more.

Methodology

Data Preprocessing

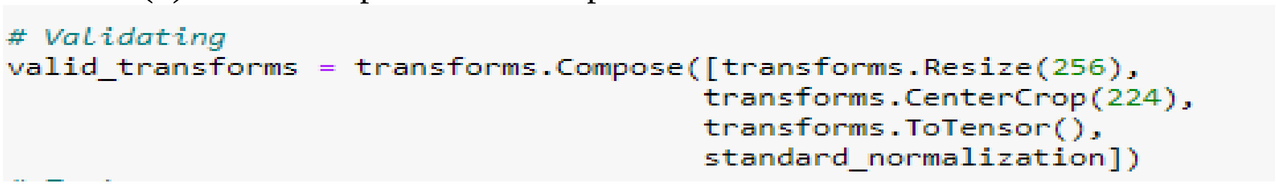
For the Haar feature-based cascade classifier no data preprocessing is necessary. For the pre-trained VGG16 model, is used only an image resize transformation to 224 x 224 pixels because is the input size of the network. The images are preprocessed and augmented in three different kinds, for the testing, the training and the validation dataset.
First, for all the three datasets i.e. train, validation, test Datasets.I put the data a mean to 0 and a standard deviation to1
Then the following steps:

1. Training dataset used the following transformation:
   (a). A Random rotation of a maximum of 30 degrees is done
   (b). A Random resize crop of 224 x 224pixels is done
   (c). A Random horizontal flip is done.



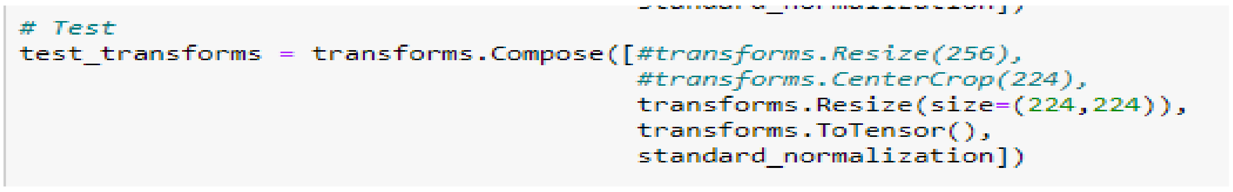
```
# training
train_transforms = transforms.Compose([transforms.RandomRotation(30),
                                       transforms.RandomResizedCrop(224), # 224x224 images
                                       transforms.RandomHorizontalFlip(),
                                       transforms.ToTensor(),
                                       standard_normalization])
```

2. Validation dataset used:
   (a). Image resize to 256 x 256 pixels
   (b). Centre crop of 224 x 224 pixels



```
# Validating
valid_transforms = transforms.Compose([transforms.Resize(256),
                                       transforms.CenterCrop(224),
                                       transforms.ToTensor(),
                                       standard_normalization])
```

3. test dataset is used the following transformation:
   (a). Image resize to 224 x 224



```
# Test
test_transforms = transforms.Compose([#transforms.Resize(256),
                                      #transforms.CenterCrop(224),
                                      transforms.Resize(size=(224,224)),
                                      transforms.ToTensor(),
                                      standard_normalization])
```

Finally, the data is put into three different data loaders and a batch size of 64 is used. And all of them combined in *loaders_scratch*.

Implementation

For the implementation I depended on Udacity Lesson "Transfer Learning" within the extra-curricular material from here . Essentially the VGG16 network was trained on the image-net data base with 14 million images and can extract 1000 classes within the image-net database. These 1000 labels are defined by the last fully connected layer of the CNN.As the following diagram:

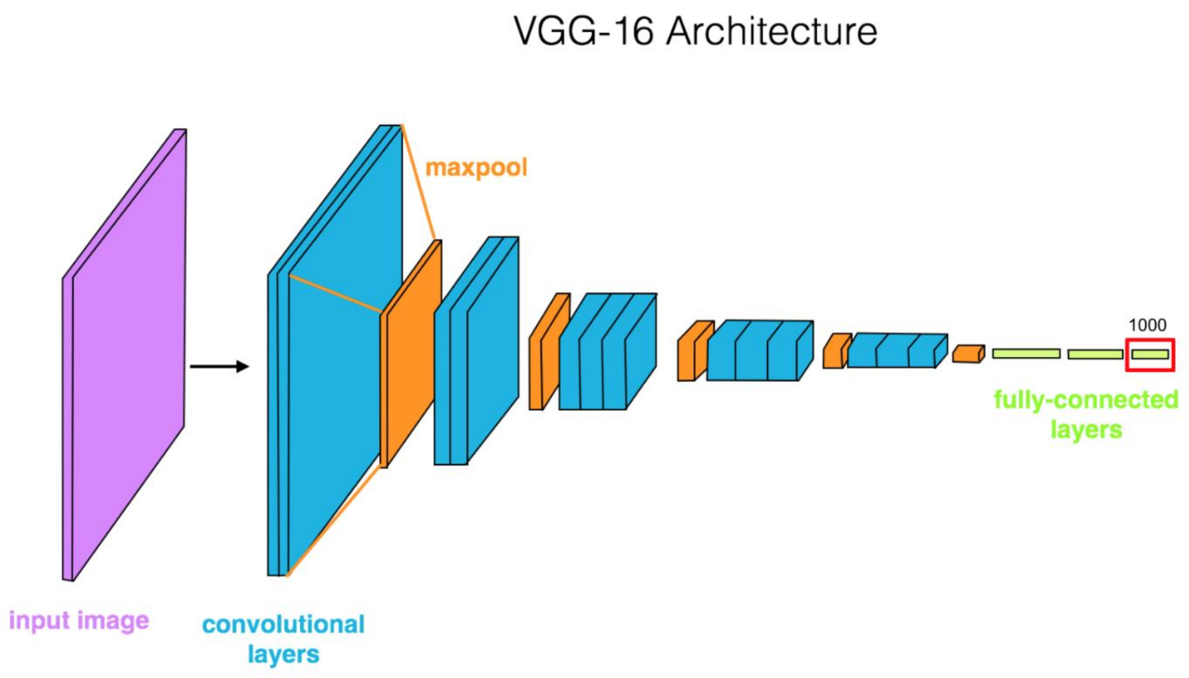
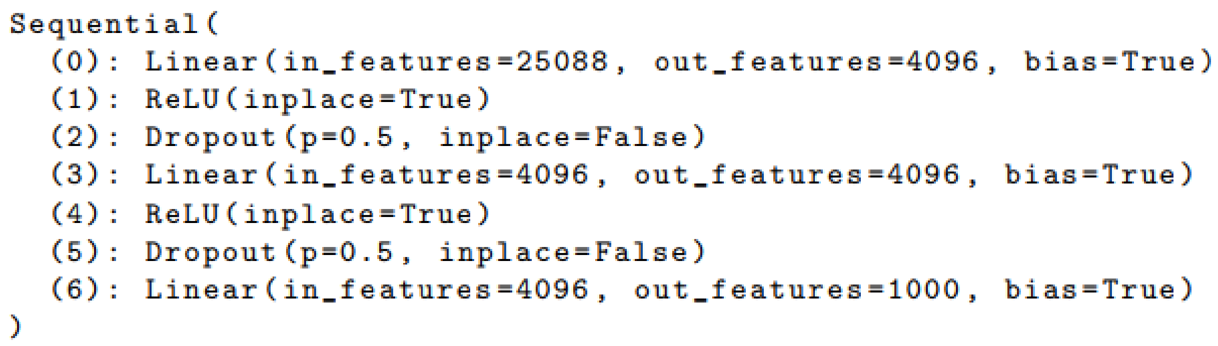


Figure 3:VGG-16 Architecture, from Udacity

The benefit of such a highly trained network is, that all the image recognition filters, which are there to recognize a vast amount of different geometrical artifacts like: horizontal lines, vertical lines, circles etc. are already in place. Moreover, the subsequent pooling and convolutional layers are also trained to identify more complex shapes like wheels, aeroplanes, etc.
Therefore the neural net can be adapted and retrained, by only changing the last layer, which defines the 1000 labels of the VGG16. The following figure shows

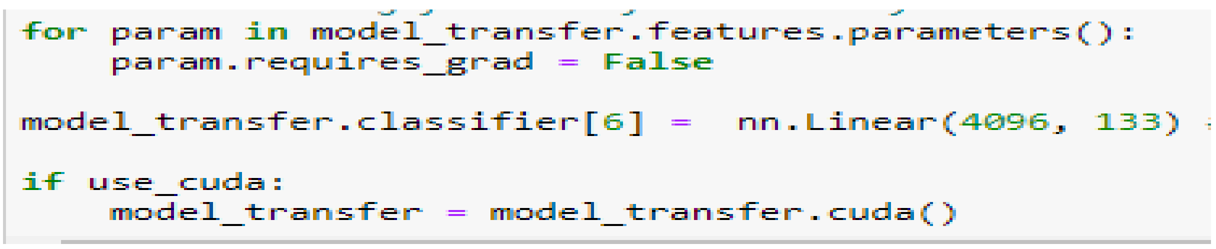
the classifier of the VGG, which correspondent to the three green boxes in figure3.



```
Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace=True)
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace=True)
  (5): Dropout(p=0.5, inplace=False)
  (6): Linear(in_features=4096, out_features=1000, bias=True)
)
```

Figure 4:original VGG16 Classifier

There are three fully connected linear layers, the last one providing **1000** nodes for 1000 image classes. The dog breed data set contains ***133 breeds***, so the following cell code reconfigure the original classifier :

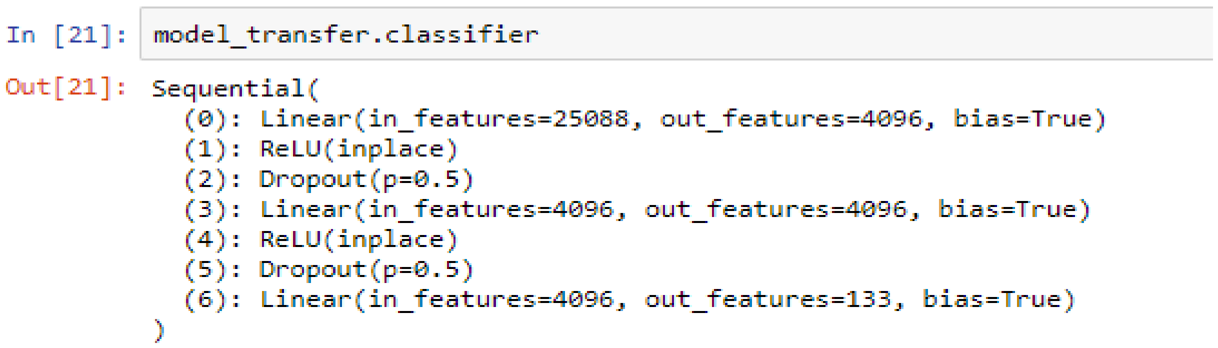


```
for param in model_transfer.features.parameters():
    param.requires_grad = False

model_transfer.classifier[6] =  nn.Linear(4096, 133)

if use_cuda:
    model_transfer = model_transfer.cuda()
```

And here is the output :



```
In [21]: model_transfer.classifier

Out[21]: Sequential(
           (0): Linear(in_features=25088, out_features=4096, bias=True)
           (1): ReLU(inplace)
           (2): Dropout(p=0.5)
           (3): Linear(in_features=4096, out_features=4096, bias=True)
           (4): ReLU(inplace)
           (5): Dropout(p=0.5)
           (6): Linear(in_features=4096, out_features=133, bias=True)
         )
```

For classification, some fully connected layers are used. For the feature, extraction is used some convolution layers. I also implemented maxpooling in order to reduce the dimension of the layers and the dropout in order to prevent overfitting. The loss function used for Cross Entropy. The optimizer used for: Stochastic gradient descent (SGD). The learning rate is set to 0.05

I just take a copy of the last step and that was an option ,so I used it then I made a little bit modification:

- The loss function used is: Cross Entropy.
- The optimizer used is: Stochastic gradient descent (SGD)

- The learning rate is set to 0.001

Refinement

For this section may be some improvement would be added as following:

- The dog images dataset could be refined using Face detection (opencv) for example or other algorithm to clean the data from human faces.
- Prediction accuracy increased due to initializing the CNN weights by speeding up the training process
- These are some parameters tuned during the process: Training epochs, dropout value, learning rate, etc

Results

Model Evaluation and Validation

The challenge is to Create a CNN to Classify Dog Breeds (from Scratch) and achieve a Test accuracy higher than **10%** and to reimplement it with the transfer learning and get a **60%** accuracy or higher . foom this after creating and implementing the two models, train and validate it . All the models trained meet the expectation end the results are:

- For the pre-trained VGG16 the dogs detected in dog images is 87.0%
- The CNN implemented from Scratch provided a test accuracy of 15 % (131/836)
- The Transfer Learning VGG16 model provides a test accuracy of 79 % (663/836)

Also for human face_detector the accuracy of the face detection is: 98.0 %
As we see my results is very good compared to the minimum accuracy required.

Justification

At least, Both models performed way better than I expected and in version 0.1 I got lower accuracy but I worked on another version and got these result that I am proud of what I learned throughout this project.

References

[1] Statista. 2021. *Europe: dog population, by country 2018 | Statista*. [online] Available at: <https://www.statista.com/statistics/414956/dog-population-european-union-eu-by-country/> [Accessed 20 March 2021].

[2] https://medium.com/nanonets/how-to-easily-build-a-dog-breed-image-classification-model-2fd214419cde

[3] https://www.mdpi.com/2227-9717/8/1/24/htm

[4] https://medium.com/@ayushraghuwanshi98/dog-breed-classifier-capstone-project-a59a0b12293d

[5] https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

[6] Michelucci, U., n.d. *Applied Deep Learning*.

[7] https://medium.com/analytics-vidhya/implementing-cnn-in-pytorch-with-custom-dataset-and-transfer-learning-1864daac14cc

[8] https://www.kaggle.com/carloalbertobarbano/vgg16-transfer-learning-pytorch

[9] https://www.kaggle.com/c/dog-breed-identification/discussion

[10] https://en.wikipedia.org/wiki/ImageNet

s