Machine Learning Engineer
Nanodegree

Capstone Project Report

Dog Breed Classifier with CNNs

Andreas Häuser

17th February 2020

Project Report

Contents

1 Definition

1.1 Poject Overview

If dog owners are unsure about their dog's breed, a dog-breed classifier can help them identify their dog's breed. This would be especially helpful for inexperienced dog owners. If a dog owner have doubts about the breed before buying a dog, i.e. there is a discrepancy between what the seller says and what you remember about the breed, you can quickly confirm your doubts with a dog breed classifier or get rid of the world.
The same problem hit me two years ago when I got my first dog called JJ. The dog breeder claimed it was a Chihuahua and I assumed that for a few months. Only shortly before I wanted to fly from Mexico to Germany with JJ and the customs papers had already been issued for a Chihuahua, a veterinarian informed me that he was actually a miniature pinscher. The upshot was that JJ had to spend two weeks in a dog hotel in Mexico.

These people and also me can be helped by developing a picture classifier that appreciates dog breeds. This is possible via the image classification in the area of deep learning, which is part of machine learning. Image classification refers to a process in computer vision that can classify an image according to its visual content. A convolutional neural network, CNN for short, is preferably used here.
CNN are a specialized type of neural network for processing data that have the structure of a grid. Image classification with CNN saw the light of day in 1994 with Yann LeCun's LeNet5 and received a huge boost in 2012 with AlexNet by Alex Krizhevsky who won the ImageNet competition.[1]

In this project I created a CNN model that can classify dog breeds. A dataset from Udacity is available for training the model, which includes more than 8000 images with exactly 133 dog breeds. In addition, I have implemented an additional dog breed. The Miniature Pinscher. So I'm going to implement 134 dog breeds in this project.
I also created a web application for the final model. With this web app you can upload a picture of your dog or yourself. The model will then return an estimate of the breed of dog. Or if you upload a picture of a human, the most similar dog breed is returned.

[1]Eugenio Culurciello. *The History of Neural Networks.* dataconomy.com. Apr. 19, 2017. URL: `https://dataconomy.com/2017/04/history-neural-networks/` (visited on 01/11/2020).

1.2 Problem Statement

In this project it is important to build a data processing pipeline to classify real-world, user-supplied images. On the basis of a image of a dog, an algorithm is supposed to give an estimate of the breed of the dog. If the image of a person is given, the algorithm should reproduce the most similar dog breed. In addition, an accuracy of 60 percent or greater should be achieved. To achieve this goal transfer learning is used . In transfer learning approach feature part of the network is freezed and only classifier is trained. Also a web application is being developed to facilitate access for end users.

The following therefore applies:

- It must be recognized on an image whether a person or a dog is present. If neither of the two is the case, an error message is issued
- The breed of the dog must be estimated
- An accuracy of 60 percent or greater should be achieved
- Transfer learning will be used to reach this goal
- Build a web application to facilitate access for end users

1.3 Metrics

The goal here is to compare the performance of my model with that of the benchmark model. Therefore, I would use accuracy as an evaluation metric. Also because the benchmark model only specifies the accuracy. The benchmark chosen for the project will be explained later in this report. We will see later that there is a slight data imbalance in the dataset. Therefore I will use also the F1 score as an additional metric.

2 Analysis

2.1 Data Exploration

The following datasets provided by Udacity will be used:

The dog dataset[2] with 8351 dog images

The human dataset[3] with 5750 human images

The original dataset provided by Udacity contains 133 dog breeds. Remember the one I mentioned in the definition under 1.1 Project Overview1.1 that I initially thought my dog JJ was a Chihuahua. His real dog breed is the miniature pinscher. And this breed is not included in the Udacity dataset.

So I searched the internet for dog images of this breed and found them at Kaggle[4]. This is a dataset with 120 dog breeds and originally comes from a dataset from Stanford University[5]. And in this one the breed miniature pinscher is included. So I extracted these images from the dataset and added them to the dog dataset folder structure.
The miniature pinscher now appears in the `train/test/valid folders` in the `134.miniature_pinscher` folder in the dogImages parent directory of my GitHub repository[6]. I have adjusted the distribution to match amount of the other dog breeds so that there is no additional imbalance generated. The number of images of this breed in each folder are

train: 76

test: 10

valid: 10

So I can now do justice to my situation described under 1.1 Domain Background1.1 and predict also the breed miniature pinscher.

[2]Udacity. *The dog dataset.* URL: `https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip` (visited on 01/10/2020).

[3]Udacity. *The human dataset.* URL: `https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip` (visited on 01/10/2020).

[4]Kaggle. *Dog Breed Identification.* URL: `https://www.kaggle.com/c/dog-breed-identification/overview` (visited on 02/15/2020).

[5]Stanford University. *Stanford Dogs Dataset.* URL: `http://vision.stanford.edu/aditya86/ImageNetDogs` (visited on 02/15/2020).

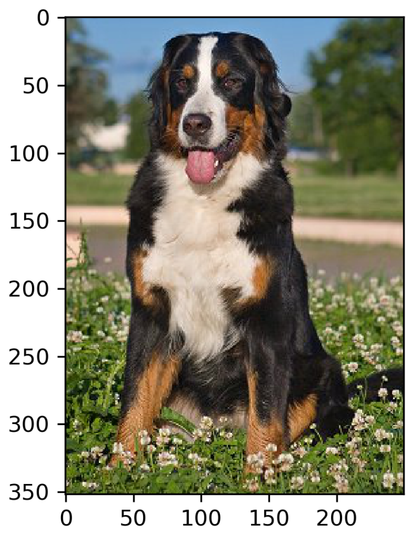
[6]Andreas Häuser. *UD-MLND-Dog-Breed-Classifier.* URL: `https://github.com/BayTown/UD-MLND-Dog-Breed-Classifier` (visited on 02/15/2020).

2.1.1 The dog dataset

The dataset contains 8,447 dog pictures, which are subdivided into training, test and validation data. So we have 6,756 images to train the model, 846 images for testing and 845 images for validation. The entire dataset contains 134 different breeds from the Afghan dog to the Yorkshire Terrier.

Here are two examples of very nice dogs from the dog dataset:

Chihuahua

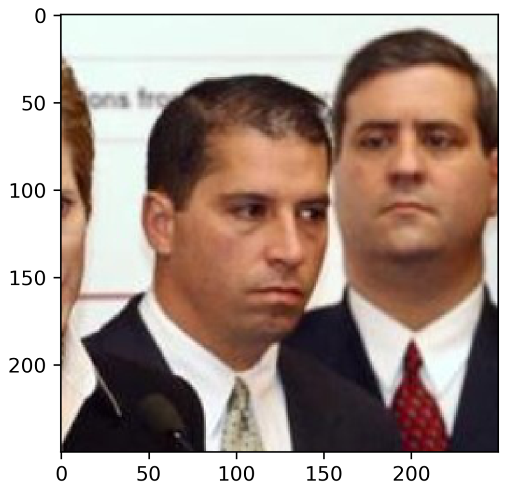


Bernese Mountain Dog

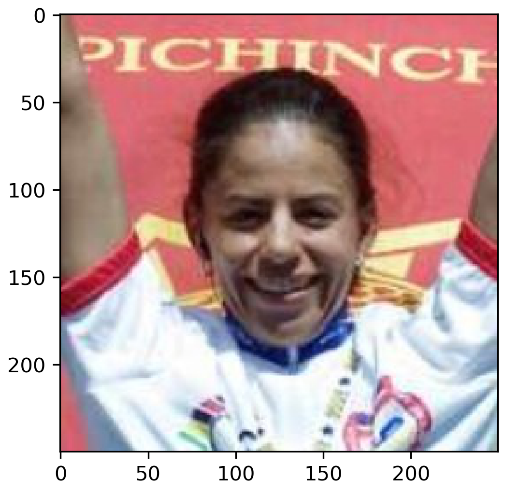
All pictures come in a .jpg format. The images have a mean of 565.85 pixel in width and a mean of 527.64 pixel in height. I will explain the distribution of the width and height of all images in section 2.2 Exploratory Visualization.

2.1.2 The human dataset

This dataset contains 5750 ordinary images of more or less ordinary people. These images are used to test the performance of the Human Face Detector. This will be the only use of these pictures in my project. Here are some sample images from the human dataset:



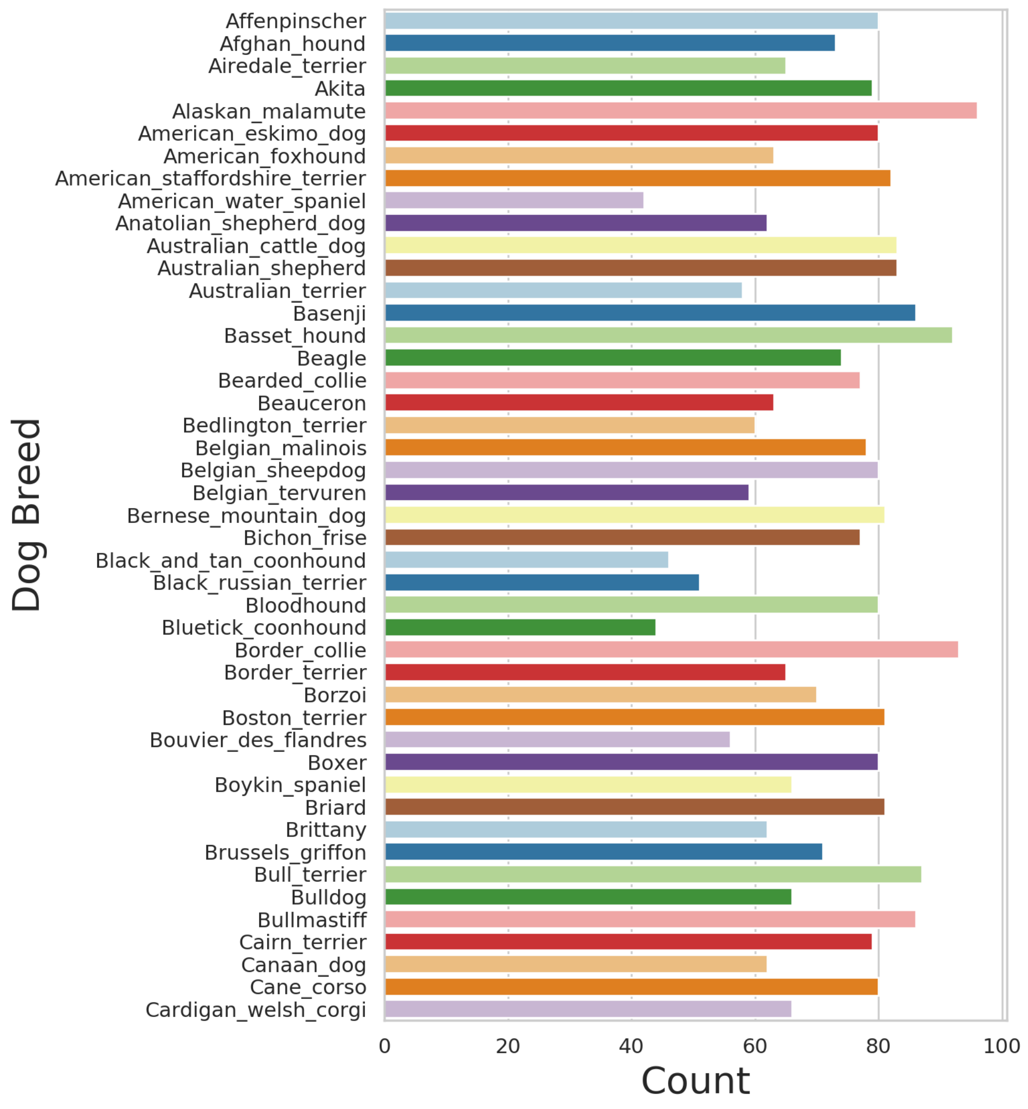
Person 1



Person 2

2.2 Exploratory Visualization

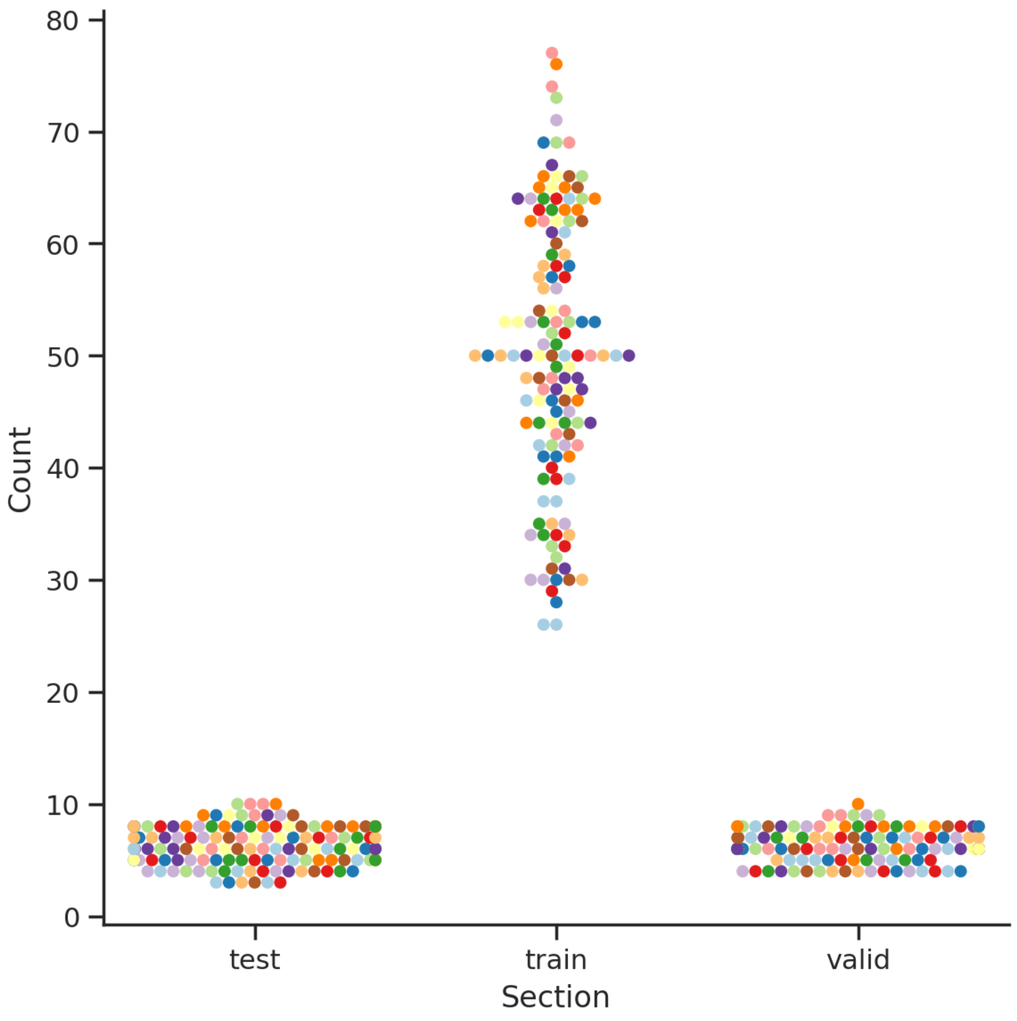
In this section we will explore the data and I will also provide plot graphics to get a feel for what is contained in the dataset and how. So that you can get an idea of what the dataset contains, i will show you a part of the distribution of the dog breeds over the complete dataset:



Dog Breed Distribution

Here you can see the distribution among the dog breeds from A to C across the sections train, test and valid in the dataset. This plot contains 45 dog breeds.

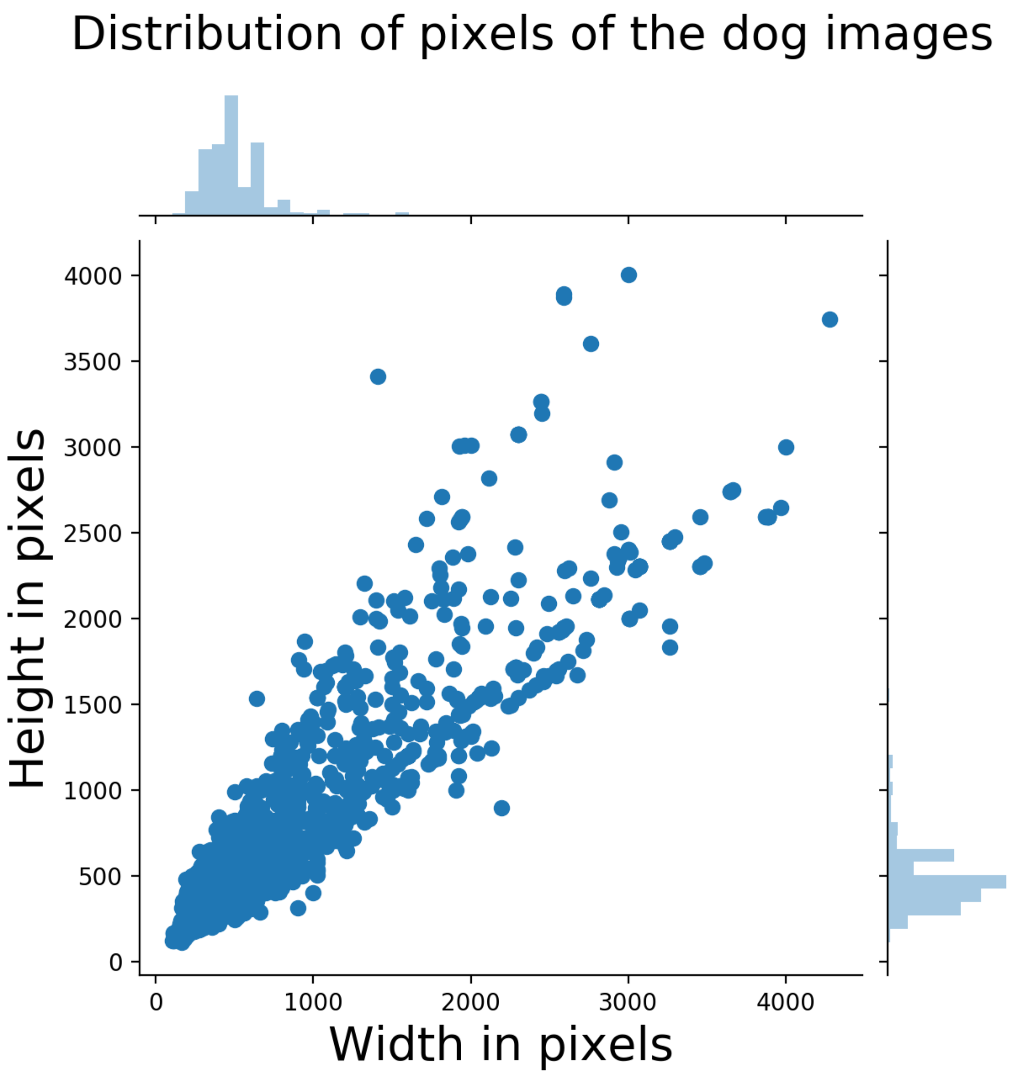
On the following plot you can see the count of images of each dog breed across the sections training, test and valid:



Dog Breed Count - Sections

The images in the sections test and valid are very well distributed because they are close together. In the section train we have a data imbalance. The dog breeds with the least pictures in this section has 26 images and the one with the most has 77 images. This data imbalance could become a problem. But fortunately there is Data Augmentation that can be used in the training process to reduce this problem.

On the following plot you can see the distribution of the image sizes over the whole dataset of the dog images:



Images - Pixel Distribution

But this has no direct relevance as I will reduce all images to the same size in the data preprocessing step. I only show this plot here if anyone is interested in how the image sizes are distributed over the dataset.

2.3 Algorithms and Techniques

I will divide the used algorithms into the following 5 sections as these are the skeleton of the project:

Human Face Detector Here I will use Cascade Classifiers from Opencv[7] to recognize the face of a person on an image. If a face is recognized this means that a person is present in the picture.

Dog detector To be able to recognize dogs on an image I will use a pretrained VGG16 model[8]. This will be implemented with the open source machine learning framework PyTorch[9]. The VGG16 model won the 2014 Imagenet competition and achieves 92.7% top-5 test accuracy in the ImageNet dataset, which is a dataset of over 14 million images belonging to 1000 classes. This pretrained model is only used for prediction and will not be trained or similar.

Dog Breed Classifier from Scratch Here I will use PyTorch to design, train and test a model from scratch. The architecture of the model will be based on the maximum possible hardware of my PC.

Dog Breed Classifier with transfer learning In this section I will apply the technique transfer learning. In transfer learning, a pretrained model is used which is then only modified according to the intended use. In my case I will use the pretrained ResNeXt-101-32x8d model[10]. The model won the second place of the Imagenet competition in 2016. This will be implemented with PyTorch. I will freeze the feature part of the model and only modify and train the classifier again.

[7]OpenCV. *OpenCV.* URL: `https://opencv.org/` (visited on 02/16/2020).

[8]Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* URL: `https://arxiv.org/abs/1409.1556` (visited on 02/16/2020).

[9]PyTorch. *PyTorch.* URL: `https://pytorch.org/` (visited on 02/16/2020).

[10]Saining Xie — Ross Girshick — Piotr Dollár — Zhuowen Tu — Kaiming He. *Aggregated Residual Transformations for Deep Neural Networks.* URL: `https://arxiv.org/abs/1611.05431` (visited on 02/16/2020).

Running the Application In this section I will bring together everything above, except the Dog Breed Classifier from Scratch. Here, first of all it is recognized whether a dog or a human is present in a image. If both are not present in an image, an error message will be returned to the user. The image is then passed to the Dog Breed Classifier algorithm, which will then return the top 3 predictions of the most similar dog breeds. The top 3 predictions are then displayed with the image to the user. It is also shown whether the image is of a dog or a human.

Web Application I will develop this web application[11] with Flask[12] and make it available for testing on AWS Elastic Beanstalk[13] for a short time. Flask is a lightweight WSGI web application framework and AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications from Amazon. This Web App will contain all the features mentioned in the previous point and will be very easy for the user to use. The big advantage will be that the user does not need to deal with a Jupyter notebook or Python.

2.4 Benchmark

I found an interesting benchmark model on the Internet. It can be viewed here[14] On page 4 in Figure 5 the accuracy of different state of the art models is compared. So I choose the ResNet-50 + FT + DA shown in the table on page 4 as my benchmark model which has a test accuracy of 89.66%.

[11] Andreas Häuser. *Dog Breed Classifier*. URL: `http://dog-breed-classifier.us-east-1.elasticbeanstalk.com` (visited on 02/16/2020).

[12] Flask. *Flask*. URL: `https://www.palletsprojects.com/p/flask` (visited on 02/16/2020).

[13] AWS. *AWS Elastik Beanstalk*. URL: `https://aws.amazon.com/elasticbeanstalk` (visited on 02/16/2020).

[14] Ayanzadeh Aydin and Vahidnia Sahand. *Modified Deep Neural Networks for Dog Breeds Identification*. May 2018. URL: `https://www.researchgate.net/profile/Aydin_Ayanzadeh2/publication/325384896_Modified_Deep_Neural_Networks_for_Dog_Breeds_Identification/links/5cd0345ea6fdccc9dd90690c/Modified-Deep-Neural-Networks-for-Dog-Breeds-Identification.pdf` (visited on 01/20/2020).

3 Methodology

3.1 Data Preprocessing

As we have already noticed, not all images have the same resolution. Also the distribution of the images is not the same in the various dog breeds. Here there is a slight data imbalance. For the data preprocessing step the PyTorch module torchvision.transforms is used.

All images of the section test and valid are first reduced to 256x256 pixels. Then a CenterCrop to 224x224 pixels is performed.

All images of the section train are first reduced to 256x256 pixels. Afterwards, data augmentation is applied to counteract the data imbalance. After the images are reduced in size, a RandomResizedCrop to 224x224 pixels is performed. Then a RandomHorizontalFlip is performed and finally a RandomRotation of 10 degrees.

The mentioned reduction to 224x224 pixels serves as a preparation for the ResNeXt-101-32x8d model. Because this was trained to this input size of the images. In addition, the images are normalized also for the same reasons of preparation for the ResNeXt-101-32x8d model.

3.2 Implementation

In this section I will explain in depth which algorithms and techniques I have used. I will divide the used algorithms into the following 5 sections as these are the skeleton of the project:

Human Face Detector Here I used one of the Cascade Classifiers from Opencv to recognize the faces. This Cascade Classifiers checks whether there is a face of a person in a image. First I used the cascade classifier of the file `haarcascade_frontalface_alt.xml` and implemented it in the function `face_detector`. This already brought good results. At 100 test images with humans 99% were recognized as humans. After that I created another cascade classifier with the file `haarcascade_frontalface_alt2.xml` and implemented it into the function `face_detector_ext`. This one is able to recognize 100% of humans as humans. I will use the latter cascade classifier in the further course.

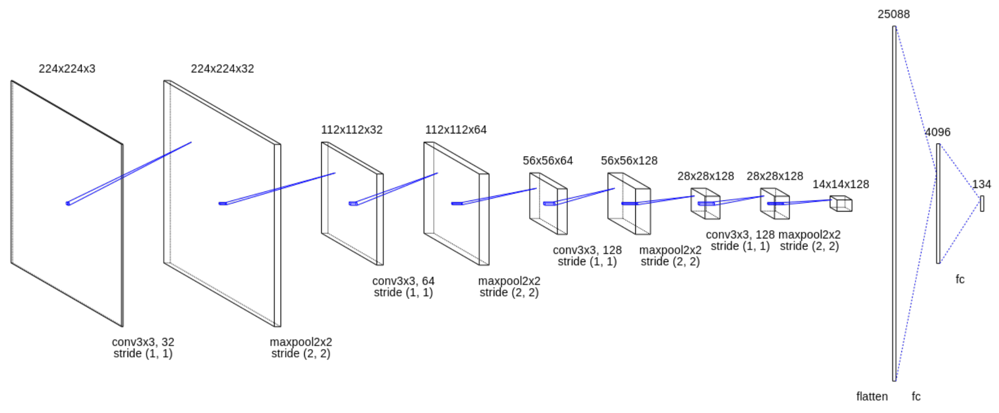
Dog detector To be able to recognize dogs on an image I used a pretrained VGG16 model. This is implemented with the open source machine learning framework PyTorch. This pretrained model is only used for prediction and will not be trained or similar. First of all, I transform the image for the VGG16 model to 224x224 pixels since it is the input size the model was trained for. After that the image is converted it to a PyTorch tensor and is normalized. The parameters for normalization are the same as those used when training the VGG16 model. Then I pass this image to the prediction and get an index value back. As a reminder, the VGG16 model was trained on the ImageNet dataset. The record has 1000 categories. The indexes between 151 and 268 are dog categories. This means that if the VGG16 model returns an index in this range, a dog is detected on the image. The function `VGG16_predict` takes care of the prediction and the function `dog_detector` evaluates the index and returns true or false accordingly.

Dog Breed Classifier from Scratch Here I used PyTorch to design, train and test a model from scratch. The architecture of the model will be based on the maximum possible hardware of my PC. When designing the architecture of the model from scratch I followed the architecture of established models like VGG16. But I had to keep my model very simple, because I quickly reached the limits of my graphics card when experimenting and reconstructing VGG16. My PC has an Nvidia GTX 1080Ti with 11GB of graphics card memory and just loading the reconstruction of VGG16 has already used over 8GB of graphics memory. So the training was no longer possible. So I was forced to simplify my model and to add a pooling layer after each convolutional layer to reduce the features. And based on my available hardware I decided to use the following model:

- Convolutional Layer with 3 input dimensions and 32 output dimensions - kernel size 3
- Relu Activation function
- Pooling layer - kernel size 2
- Convolutional Layer with 32 input dimensions and 64 output dimensions - kernel size 3
- Relu Activation function
- Pooling layer - kernel size 2
- Convolutional Layer with 64 input dimensions and 128 output dimensions - kernel size 3
- Relu Activation function

- Pooling layer - kernel size 2
- Convolutional Layer with 128 input dimensions and 128 output dimensions - kernel size 3
- Relu Activation function
- Pooling layer - kernel size 2
- Flatten layer to convert the pooled feature maps to a single vector with a length of 25088
- Dropout with a probability of 0.25
- Fully connected Linear Layer with an input size of 25088 and an output size of 4096
- Relu Activation function
- Dropout with a probability of 0.25
- Fully connected Linear Layer with an input size of 4096 and an output size of 134

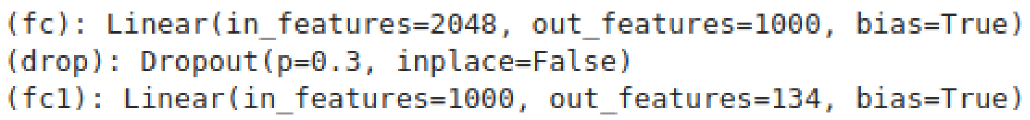
Here you can see the architecture of the model from scratch



Model from Scratch - Architecture

Dog Breed Classifier with transfer learning In this section I applied the technique transfer learning. In my case I will use the pretrained ResNeXt-101-32x8d model. This will be implemented with PyTorch. I will freeze the feature part of the model including the Adaptive Average Pooling layer and only replace and train the classifier part. First of all I loaded the pre-trained model and freezed all layers using `requires_grad = False`. Then I replaced the only fully connected layer `(fc)`, added a dropout layer `(drop)` and inserted a fully connected layer `(fc1)` as last layer. This last layer has 134 output neurons (output features). If layers are replaced in this way, they are automatically set back to `requires_grad = True` and can therefore be trained.

Here is an overview of the classifier section:



```
(fc): Linear(in_features=2048, out_features=1000, bias=True)
(drop): Dropout(p=0.3, inplace=False)
(fc1): Linear(in_features=1000, out_features=134, bias=True)
```

Model Transfer Learning - Classifier

Running the Application In this section the `run_app` function brings everything together. The Human Face Detector, the Dog Detector and the Dog Breed Classifier with transfer learning. Here, first of all it is recognized whether a dog or a human is present in a image. If both are not present in an image, an error message will be returned to the user. The image is then passed to the Dog Breed Classifier function `predict_breed_transfer` which transforms the image, makes the predictions, and returns the top 3 predictions of the most similar dog breeds. This is posible through the PyTorch-function `torch.topk()`. The top 3 predictions are then displayed to the user with the image. It is also shown whether the image is of a dog or a human.

Web Application I have developed this web application with Flask and made it available for testing on AWS Elastic Beanstalk for a short time until 02/19/2020. Here you can visit my Website[15]. Flask is a lightweight WSGI web application framework and AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications from Amazon. This Web App contains all the features mentioned in the previous point and is very easy for the user to use. The user only needs to upload an image and after the prediction process he will be redirected to the results page where the top 3 results will be displayed. To make sure the content of the website is displayed and arranged in a nice way I used the open source toolkit Bootstrap[16]. You can find more about this in my GitHub repo in the folder `webapp`.

[15]Andreas Häuser. *Dog Breed Classifier*. URL: `http://dog-breed-classifier.us-east-1.elasticbeanstalk.com` (visited on 02/16/2020).

[16]Bootstrap. *Bootstrap - open source toolkit*. URL: `https://getbootstrap.com/` (visited on 02/16/2020).

3.3 Refinement

In the section Dog Breed Classifier with Transfer Learning I used the ResNeXt-50 model at the beginning. But this had only a test accuracy of 83%. So I switched to the ResNeXt-101 model. This model has a better performance, as it turned out.

4 Results

4.1 Model Evaluation and Validation

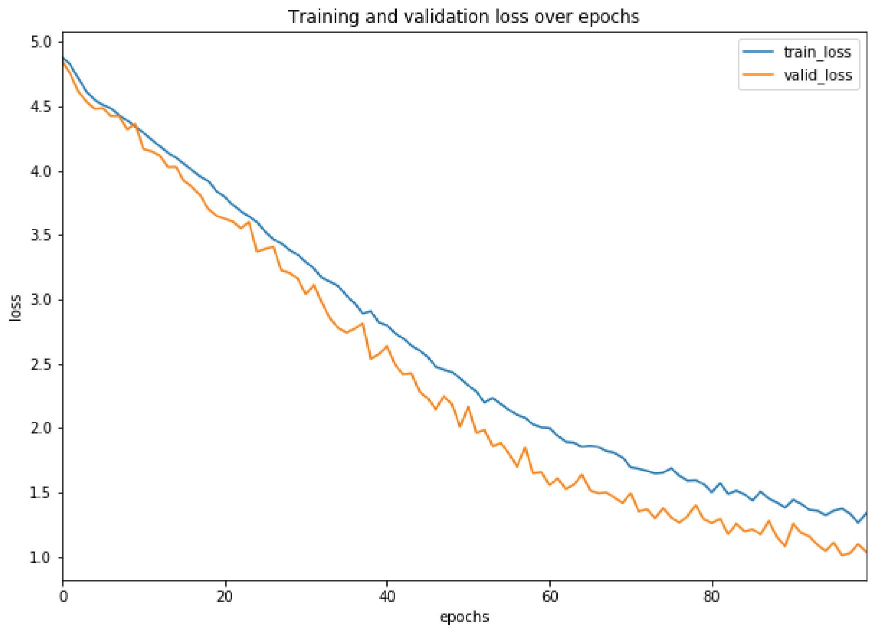
During the training of both models the model was validated with the Validation set after each epoch. After the training the models were tested with a test dataset. I will describe the training process for each model here.

4.1.1 Model from Scratch

The final architecture and the hyperparameters were based on the maximum hardware available to me. The architecture is described in detail under point 3.2 Implementation in the sub section 'Dog Breed Classifier form Scratch'.
I would like to mention again that the last fully connected layer has 134 ouput features, which represent the 134 dog breeds (classes).
Here I would like to discuss the selected hypoparameters:

- A learning rate of 0.02 worked very well here
- 100 epochs were enough for this task
- As optimizer I chose the stochastic gradient descent (SGD) optimizer because it works very well for image classification tasks. It turned out to work also very well for the given task

The following plot shows the progress of the training loss and the validation loss over the epochs during training:



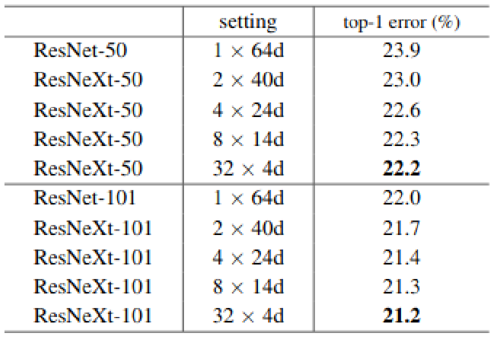
Training and validation loss over the epochs

On this plot you can see very well how the training loss and the validation loss decrease.
In the training function the model and the weights of the one with the lowest validation loss for an epoch are stored.

To verify the robustness of the final model this model was tested with 846 pictures of all 134 dog breeds.

4.1.2 Model with transfer learning

I have chosen the ResNeXt-101 model because it is one of the best performing pre-trained models available. It performs better than the ResNet50 model in the Top-1 error comparison[17] based on the ImageNet dataset:



	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

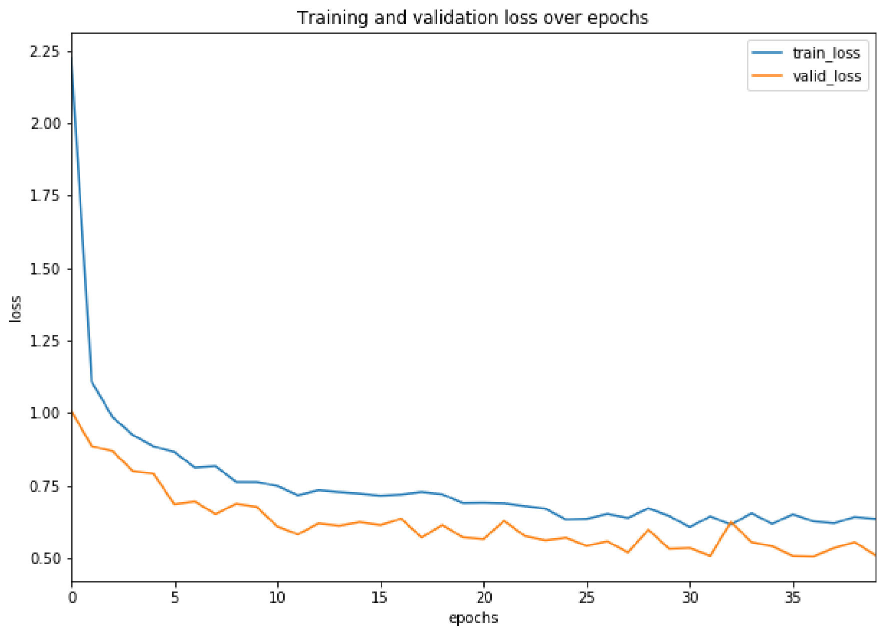
Training and validation loss over the epochs

The architecture of this model is described in detail under point 3.2 Implementation in the sub section 'Dog Breed Classifier with transfer learning'. I would like to mention again that the last fully connected layer has 134 ouput features, which represent the 134 dog breeds (classes).
Here I would like to discuss the selected hypoparameters:

- A learning rate of 0.01 worked very well here
- A momentum of 0.9 helps for faster converging
- 40 epochs were enough for this task
- As optimizer I chose the stochastic gradient descent (SGD) optimizer because it works very well for image classification tasks. It turned out to work also very well for the given task

[17]Saining Xie — Ross Girshick — Piotr Dollár — Zhuowen Tu — Kaiming He. *Aggregated Residual Transformations for Deep Neural Networks.* URL: `https://arxiv.org/abs/1611.05431` (visited on 02/16/2020).

The following plot shows the progress of the training loss and the validation loss over the epochs during training:



Training and validation loss over the epochs

On this plot you can see very well how the training loss and the validation loss decrease.
In the training function the model and the weights of the one with the lowest validation loss for an epoch are stored.

To verify the robustness of the final model this model was tested with 846 pictures of all 134 dog breeds.

4.2 Justification

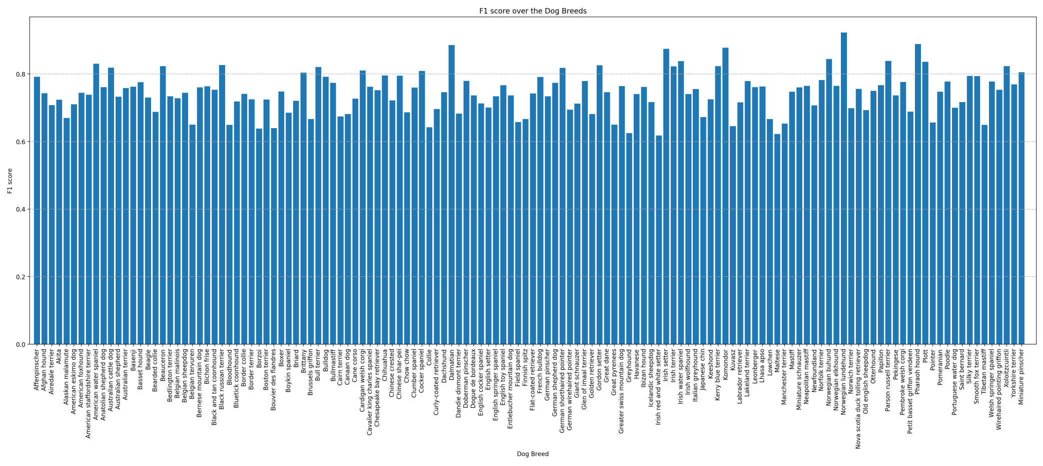
Here I will show the results of both models and finally give a comparison to the chosen benchmark:

4.2.1 Model from Scratch

To be honest, the results obtained in testing the model are better than expected. I will list the results here:

- Test Accuracy: 74.85%
- Test F1 score: 0.7523

The accuracy and the F1 score are very close together. These are both very good values for the Model from Scratch and this means that I was able to compensate the data imbalance with the data augmentation. The following list shows the F1-scores of each dog breed (class):



Training and validation loss over the epochs

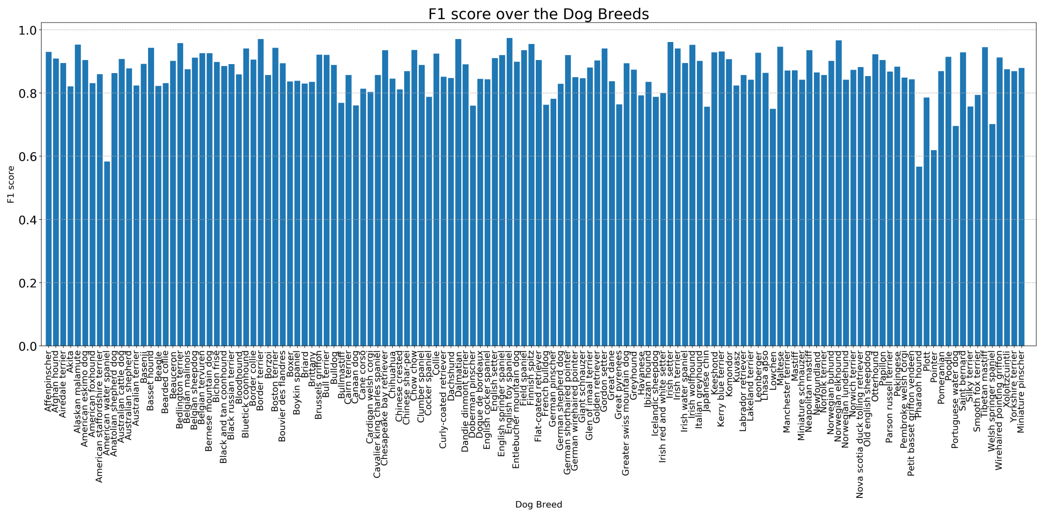
On this plot you can see the individual F1 scores. For a better view you can zoom in. The best scores are for 'Norwegian lundehund' and the worst for 'Irish red and white setter'.

4.2.2 Model with transfer learning

With this model the results are also better than expected. I will list the results here:

- Test Accuracy: 86.26%
- Test F1 score: 0.8594

The accuracy and the F1 score are very close together. These are both very good values for the Model with transfer learning and this means that in this case I also was able to compensate the data imbalance with the data augmentation. The following list shows the F1-scores of each dog breed (class):



Training and validation loss over the epochs

On this plot you can see the individual F1 scores. For a better view you can zoom in. The best scores are for 'English toy spaniel' and the worst for 'Pharaoh hound'.

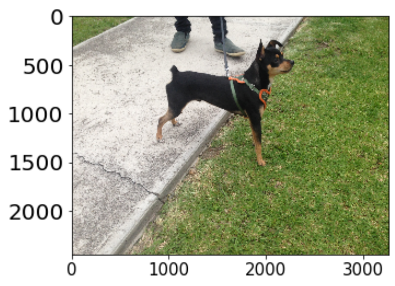
4.2.3 Benchmark Comparison

The Bechmark Model ResNet-50 + FT + DA has a test accuracy of 89.66%. Unfortunately I could not reach this accuracy with my two models. In the Model with transfer learning I only just missed this goal. This is most likely due to the fact that the Benchmark Model had 20580 images for training and testing with only 120 dog breeds. This is more than twice as many images as I have available.

5 Conclusion

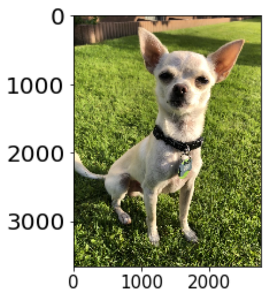
5.1 Free-Form Visualization

On this occasion I would like to show some outputs of the developed function `predict_breed_transfer`:



```
Hello dog! Here are the predictions for your dog breed
Top  1 prediction:  92.28% - Miniature pinscher
Top  2 prediction:   4.37% - Manchester terrier
Top  3 prediction:   3.35% - German pinscher
```

JJ in prediction

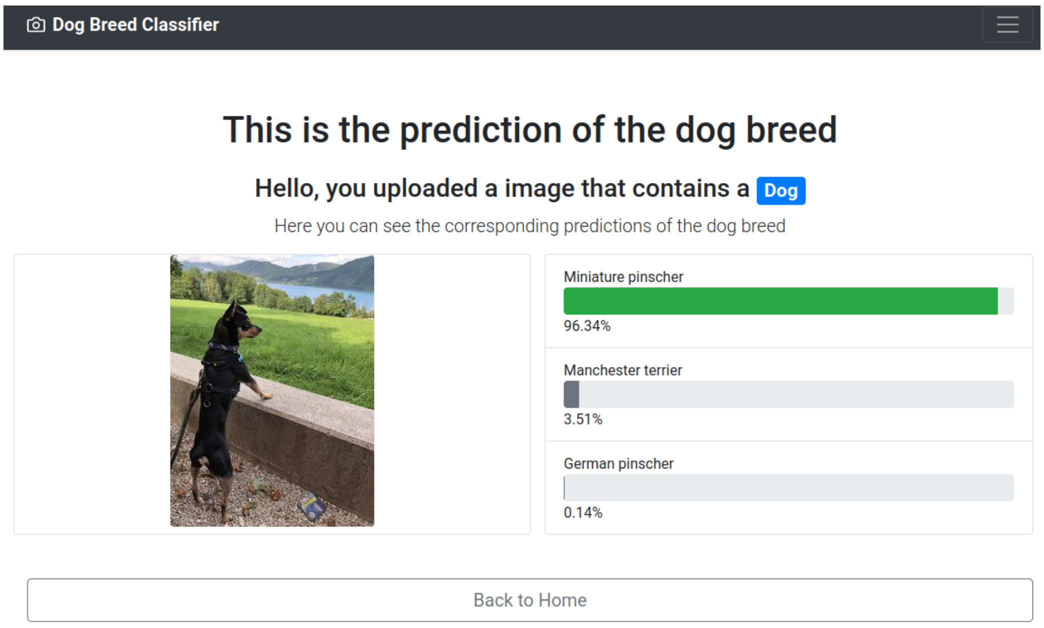


```
Hello dog! Here are the predictions for your dog breed
Top  1 prediction: 100.00% - Chihuahua
Top  2 prediction:   0.00% - Norwegian buhund
Top  3 prediction:   0.00% - Chinese crested
```

Coco in prediction

Here you can see my two dogs JJ and Coco, as they are output by the above mentioned function. You can see that they are both recognized as dogs. Actually I should insert a smiley here. You can also see the top 3 predictions with the respective percentages.

Now to my website which can be reached here. There you can test the algorithm directly. The website will be available for testing until 02/19/2020. On the next page I have added two screenshots of the website.



This is the prediction of the dog breed

Hello, you uploaded a image that contains a Dog

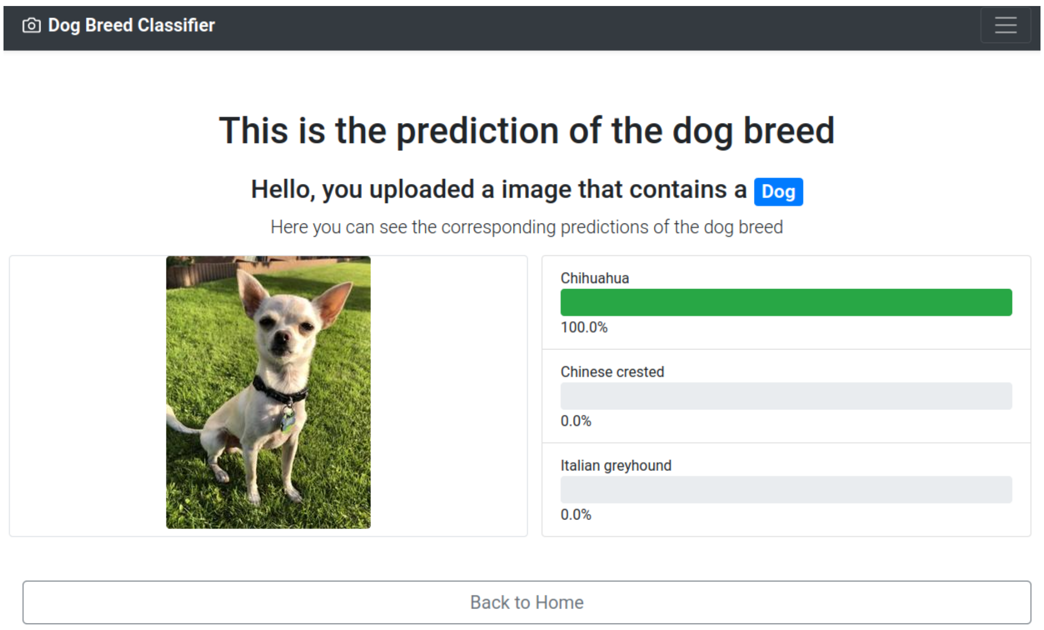
Here you can see the corresponding predictions of the dog breed

Miniature pinscher
96.34%

Manchester terrier
3.51%

German pinscher
0.14%

JJ Prediction on the Webiste



This is the prediction of the dog breed

Hello, you uploaded a image that contains a Dog

Here you can see the corresponding predictions of the dog breed

Chihuahua
100.0%

Chinese crested
0.0%

Italian greyhound
0.0%

Coco Prediction on the Webiste

5.2 Reflection

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

1. An initial problem and relevant public dataset of images were found
2. The images was downloaded
3. A Human Face Detector was developed
4. A dog detector was developed
5. The images were preprocessed and data augmentation was applied
6. A Model from Scratch was developed, trained and tested
7. A Model with transfer learning was developed, trained and tested
8. An application was designed and developed to show the user the top 3 predictions for a given image
9. A website was designed and developed to show the user the top 3 predictions for a given image
10. Every function and every part of the project has been clearly documented
11. The GitHub repository has been clearly documented and provided with README files

Step 3 was one of the most difficult chapters for me, as I first tried to reconstruct the VGG16 model. Unfortunately this did not work because of my hardware. I also found the development of the web application very difficult, because these were my first steps with Flask and the handling of AWS Elastic Beanstalk.

5.3 Improvement

- The entire training, validation and test dataset contains only 134 dog breeds. The FCI[18] currently recognizes over 352 dog breeds. If this diversity were represented in the dataset, the algorithm would be able to provide an estimate for all these dog breeds.
- In relation to the website the Inference could be better. The inference could have been designed with a RESTful-API but unfortunately I didn't had the time. This is also a question of cost, because with AWS you need one instance for the inference and one instance for the web server. At the moment the web server and the Inference are running on the same instance.

[18]FCI. *FCI Website.* URL: `http://www.fci.be/en/Presentation-of-our-organisation-4.html` (visited on 02/17/2020).

References

AWS. *AWS Elastik Beanstalk.* URL: https://aws.amazon.com/elasticbeanstalk (visited on 02/16/2020).

Aydin, Ayanzadeh and Vahidnia Sahand. *Modified Deep Neural Networks for Dog Breeds Identification.* May 2018. URL: https://www.researchgate.net/profile/Aydin_Ayanzadeh2/publication/325384896_Modified_Deep_Neural_Networks_for_Dog_Breeds_Identification/links/5cd0345ea6fdccc9dd90690c/Modified-Deep-Neural-Networks-for-Dog-Breeds-Identification.pdf (visited on 01/20/2020).

Bootstrap. *Bootstrap - open source toolkit.* URL: https://getbootstrap.com/ (visited on 02/16/2020).

Culurciello, Eugenio. *The History of Neural Networks.* dataconomy.com. Apr. 19, 2017. URL: https://dataconomy.com/2017/04/history-neural-networks/ (visited on 01/11/2020).

FCI. *FCI Website.* URL: http://www.fci.be/en/Presentation-of-our-organisation-4.html (visited on 02/17/2020).

Flask. *Flask.* URL: https://www.palletsprojects.com/p/flask (visited on 02/16/2020).

Häuser, Andreas. *Dog Breed Classifier.* URL: http://dog-breed-classifier.us-east-1.elasticbeanstalk.com (visited on 02/16/2020).

— *Dog Breed Classifier.* URL: http://dog-breed-classifier.us-east-1.elasticbeanstalk.com (visited on 02/16/2020).

— *UD-MLND-Dog-Breed-Classifier.* URL: https://github.com/BayTown/UD-MLND-Dog-Breed-Classifier (visited on 02/15/2020).

He, Saining Xie — Ross Girshick — Piotr Dollár — Zhuowen Tu — Kaiming. *Aggregated Residual Transformations for Deep Neural Networks.* URL: https://arxiv.org/abs/1611.05431 (visited on 02/16/2020).

— *Aggregated Residual Transformations for Deep Neural Networks.* URL: https://arxiv.org/abs/1611.05431 (visited on 02/16/2020).

Kaggle. *Dog Breed Identification.* URL: https://www.kaggle.com/c/dog-breed-identification/overview (visited on 02/15/2020).

OpenCV. *OpenCV.* URL: https://opencv.org/ (visited on 02/16/2020).

PyTorch. *PyTorch.* URL: https://pytorch.org/ (visited on 02/16/2020).

Simonyan, Karen and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* URL: https://arxiv.org/abs/1409.1556 (visited on 02/16/2020).

Udacity. *The dog dataset.* URL: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip (visited on 01/10/2020).

— *The human dataset.* URL: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip (visited on 01/10/2020).

University, Stanford. *Stanford Dogs Dataset*. URL: http://vision.stanford.edu/aditya86/ImageNetDogs (visited on 02/15/2020).