Dog Breed Classifier

by udacity student David Maisonnette Jentjens

I. Definition

a. Project Overview

Every day more and more people get familiar with the power of image classification and subsequent alterations, whether they are aware of it or not. Using snapchat and/or instagram A.R. filters require complex underlying neural networks built upon training on millions, and in somes cases **billions**, of images. Humans have integrated these tools to achieve things such as dog filters, recognition of faces in photo galleries.

b. Problem Statement

The Fédération Cynologique Internationale (FCI) [1], also known as the World Canine Organization, states there are at least 360 officially recognized dog breeds around the globe. With this many breeds, a human can't possibly be able to keep up with most dog breeds. Of course most people are able to recognize the Poodles and Chihuahuas, but what about the American Staffordshire Terriers? And the Lancashire Heelers? And what about the Kintamani-Bali dogs? That's too much for our monkey brains to remember. This is where A.I. can help us.

c. Project Outline (Goal)

The goal of the project will be to build an image classification model, in which images of dogs are classified based on their breeds. If the image of a human is provided instead, it will approximate the dog breed that looks the most similar. It will be using CNNs (Convolutional Neural Networks) to achieve this purpose (Fig. 1).

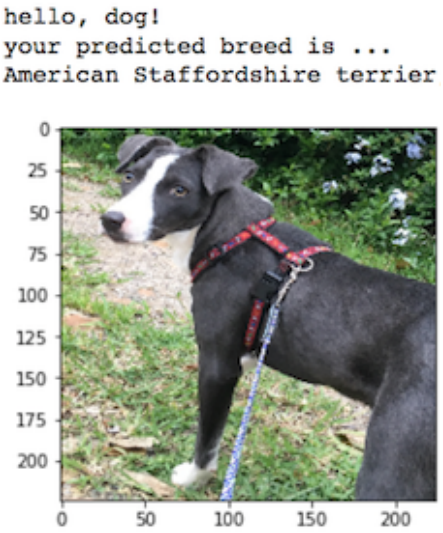

Fig. 1. Dog breed classification example.

d. Evaluation Metrics

The model will be tested on a section of the initial dataset, reserved for testing. The primary evaluation metric will be accuracy. Even though there is a small but relevant imbalance in the dogs dataset (mentioned in the Analysis chapter), accuracy is still a reliable enough metric to test our dataset on. However, the main reason accuracy was chosen is so that we are able to benchmark our model, a procedure which is discussed in the benchmark model section. This benchmark model uses accuracy as their metric for evaluation, so it makes sense to use it here.

Another way in which the model will be tested is by selecting images randomly and running detection on them. The project notebook will contain these tests, along with their expected outcomes. The following image contains an example of a test and its respective output (Fig. 2):

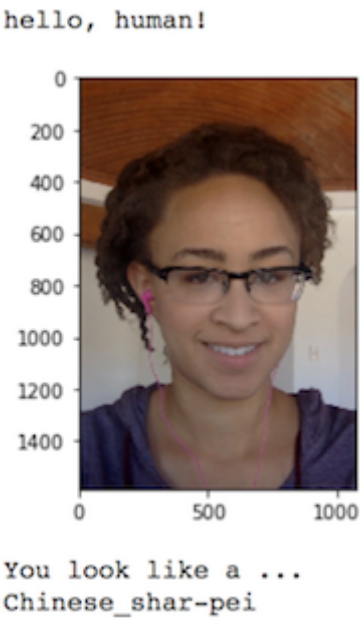

Fig. 2. This is an example of a test and its respective output:

II. Analysis

a. Data Exploration and Visualization

The base human and dog datasets are also provided in the project's repository, consisting of **13233 samples** of image data for the human dataset and **8351 samples** for the dog dataset. Each image consists of one or more of the species it is supposed to portray. For example, an image may contain more than one human face for the human dataset.

The dog dataset can be obtained via the following link:
https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
The human dataset can be obtained via the following link:
http://vis-www.cs.umass.edu/lfw/lfw.tgz

I. Dog Dataset

The dog dataset provided in the link above is given in the form of a directory with training, validation and testing folders inside. These in turn contain a number of subfolders, where each subfolder is named after a given dog breed, and contains multiple images of dogs with that breed (Fig. 2).

Fig. 2. Structure of the dog input data

The average number of images inside each subfolder (size of class) is around 50, with the smallest class being of size 26 and the largest being 77. This signifies that there is a relatively high amount of class imbalance in our dataset, which poses a few challenges further on the Implementation section.

The class imbalance can be more clearly spotted in the graph below, in which class sizes are plotted along their frequencies in a countplot. The imbalance can be clearly spotted in the leftmost bars (Fig. 3).

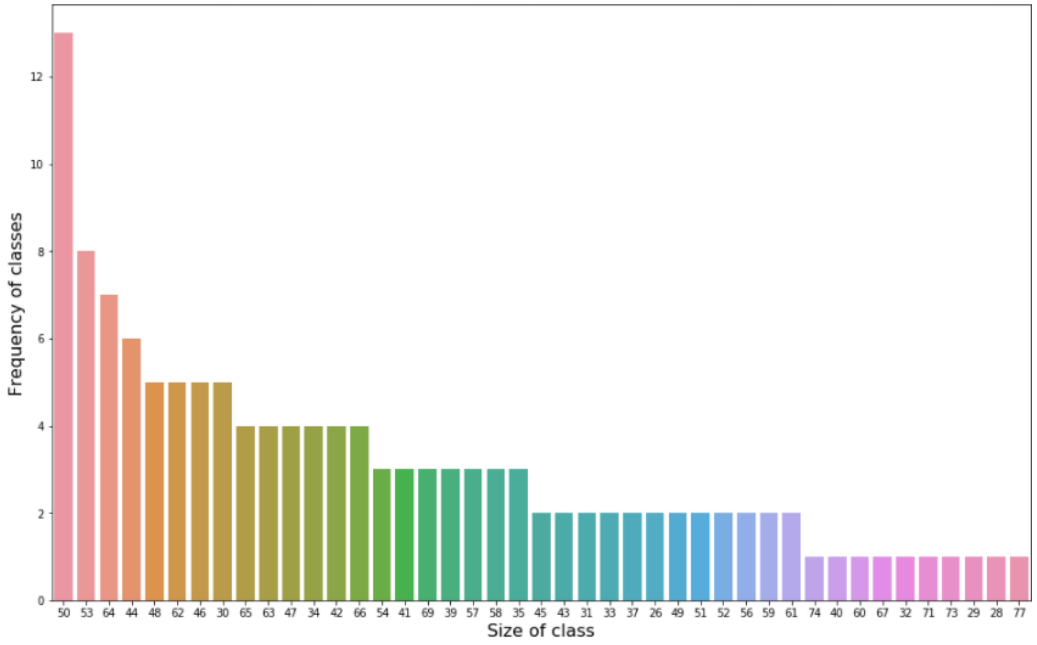

Fig. 3. Graph showing class imbalance in the dog dataset

II. Human Dataset

The human input data is a little different from the dog data, as it is not divided into training, testing and validation data. The input directory contains multiple subfolders, which themselves contain different images of the same person. Each subfolder is then named after the person that is represented in them. The subfolders contain at least one image of each person, but may contain many (Fig. 4).

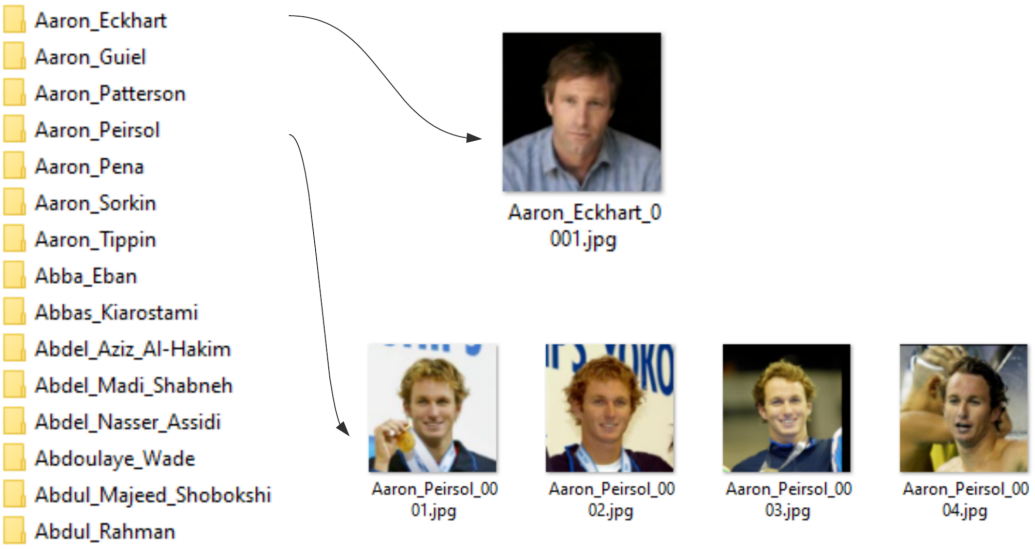

Fig. 4. Structure of the human input data (With two examples shown)

Here, classes are defined by the amount of faces detected in the images. Analogously to the dog dataset, there is also a lot of class imbalance present. Most images only contain one person, with only very few containing two or more. It can be stated that the class imbalance is even higher in this data, since less than 20% of images fall into the category of more than one face detected.

However, there won't be any problems with the class imbalance here though, since we will be using OpenCV's implementation of Haar feature-based cascade classifiers [2] to detect the human faces in images, a pre-trained model.

b. Algorithms and Techniques

This project will be built by following the steps provided in the notebook, available in this github repository. The final goal will be to have a working prediction model that can accurately classify dog breeds from different images of dogs, as well as approximate dog breeds to human faces.

Given an image, the model will first try to verify if the image is a dog or a human. It will try to detect a human face with the OpenCV Haar feature-based cascade pre-trained classifier, mentioned earlier. The algorithm will then operate under the following rules:

- If a dog is detected in the image, it will return the predicted breed.
- If a human is detected in the image, it will return the resembling dog breed.
- If neither is detected in the image, it will provide output that indicates an error.

Following these rules, it becomes clear we must implement three models. One that detects humans, another that detects dogs, and finally one that is able to correctly classify breeds.

The project notebook will contain a model built from scratch using PyTorch convolutional neural networks, and a transfer learning model, using the pre-trained CNN ResNet-50 [3] for image recognition (Fig. 5).

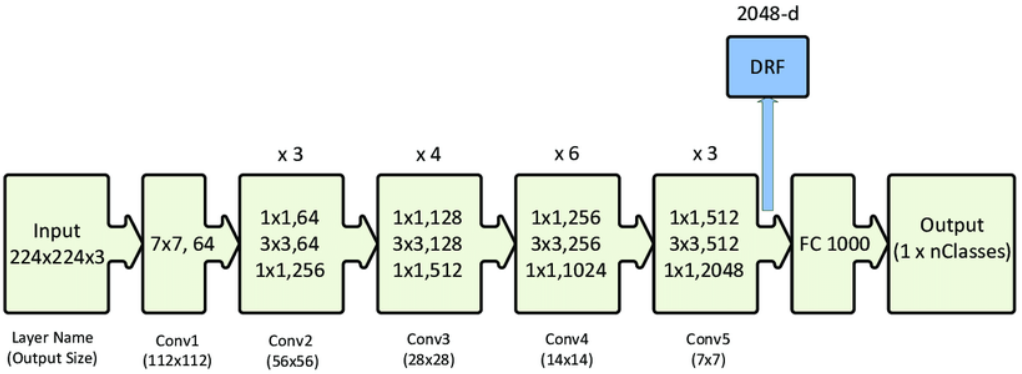

Fig. 5. Representation of the standard ResNet-50 network architecture.

c. Benchmark Model

The transfer learning model will be evaluated and compared to the model created by Whitney LaRow, Brian Mittl, Vijay Singh, documented in the oxford paper Dog Breed Identification [4]. Their model achieved a top accuracy of 90% in identifying dog breeds, which is already a pretty high score for such a complex problem.

III. Methodology

a. Data Preprocessing

There are 3 types of detections run on the overall data, and all of them require a fair amount of data processing. As stated earlier, we need to be able to detect humans, detect dogs, and if a dog is detected, we need to be able to predict its breed.

I. Detecting Humans

For detecting humans, the pre-trained OpenCV model was used. The model was downloaded as a XML file from github. from Before using it, the human images were simply converted into grayscale, following standard procedure described in the OpenCV: Cascade Classifier documentation [5].

II. Detecting Dogs

For detecting dogs in images, the VGG-16 model, from the Oxford article “Very Deep Convolutional Networks for Large-Scale Image Recognition” [6]. was used. The model and its weights were pre-trained on the ImageNet [7] dataset. Before being fed to the

model, the images are read in as RGB, resized to a resolution of 224 x 224 pixels, following specifications mentioned in the VGG-16 model paper, and are also center cropped. Then the red, green and blue channels of the images are normalized to the mean values of (0.485, 0.456, 0.406) and the standard deviation values of (0.229, 0.224, 0.225) respectively. These are the values recommended in the torchvision.transforms section of the PyTorch documentation [8].

III. Predicting breeds

For predicting the breeds, the same transformations were done as in the previous section, along with an extra horizontal flip and a random rotation. The extra transformations are added because we want to add variability to prevent overfitting, since here a model will be trained from scratch (Fig. 6).

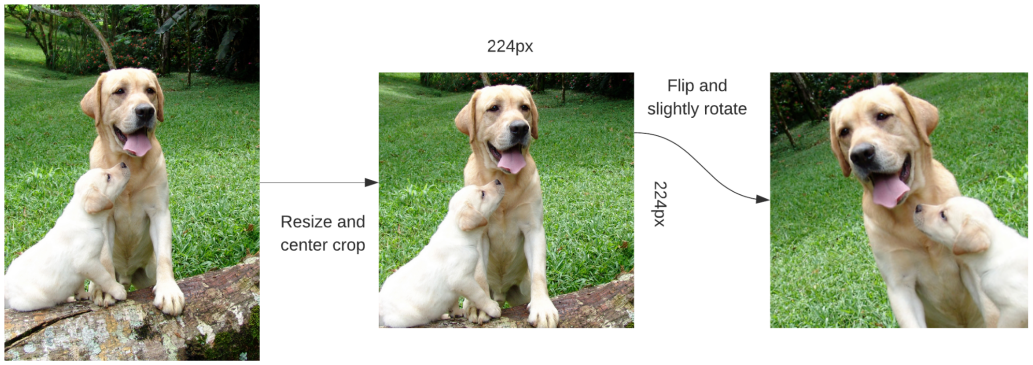

Fig. 6. Transformations done on the images before feeding them into the model.

IV. Predicting breeds (Using transfer learning)

The pre-processing done on this step is exactly the same as in the scratch model.

b. Implementation

The implementation of the project can be divided in accordance with the previous section, therefore approaching human detection, dog detection and breed prediction.

I. Detecting Humans

After the data processing step, the images were fed to the model, predicting how many faces it detected, if any. A conditional was added to simplify the prediction output, asking if the number of faces detected was higher than 0, so as to only verify if any faces are present, rather than saying how many were found. This process was relatively simple, since there was very little preprocessing done and the model used was pre-trained.

II. Detecting Dogs

For detecting dogs, the pre-processed images were fed to the pre-trained VGG-16 model, which is preferably run on the GPU, using cuda. According to the ImageNet category dictionary [9], dogs are represented by categories from keys 151-268, inclusive. So to determine if the model correctly detected a dog, all that needs to be done is check whether the predicted category falls within this interval.

III. Predicting breeds (From Scratch)

This is the most challenging part of the project, since it involves creating a model from scratch that is able to correctly classify more than 100 different breeds, which is a monumentally complex task, even for a human.

The chosen model is a Convolutional Neural Network (CNN), inspired by the VGG-16 model. Due to the exceptional performance of CNNs in pattern recognition, they tend to produce great results detecting objects in images.

The model's architecture is composed of 4 convolutional layers, attached to two fully connected layers. Every convolutional layer is followed by a ReLU activation function and a max pooling filter. The max pooling filter is used to down-sample the images, thereby highlighting its most predominant features. It also helps to prevent overfitting and reduce computation time.

This is a relatively simple model that is still able to offer enough complexity to classify images correctly, to a given extent. The final architecture is pictured in the refinement section (Fig. 7) (Note how it follows the structure of the more complex VGG-16 model).

A cross entropy loss function was used during the training process. Stochastic gradient was used as an optimizer for the model, since it is specially suited for high-dimensionality optimization problems because of its ability to reduce computational load. The training process consists of a batched loss minimization, accompanied by a validation step, saving the model if the validation has decreased. It is run on a set number of epochs.

IV. Predicting breeds (Using transfer learning)

To be able to obtain better results in a complex task such as this one, a transfer model was used. The model that was chosen is the ResNet-50 convolutional neural network. It was chosen because it is a very light (compared to VGG-16, for example) and yet extremely powerful model. ResNet or Residual Network uses residual learning instead of trying to learn some features. It was built in order to help mitigate the problem of Accuracy Saturation(degradation) and vanishing gradients, by introducing skips between certain layers of the network. The research behind it is very extensively documented in the Microsoft article where it is first described, "Deep Residual Learning for Image Recognition" [10].

Besides being very performant, Resnet-50 is also excellent at image classification, having won the ImageNet challenge in 2015[11]. For implementing it with our data, the final fully connected layer of the model was replaced by a custom output layer of 133 neurons, representing the different kinds of breeds available to classify from.

c. Refinement

In an attempt to reduce overfitting on the scratch model, a dropout layer was added to the end of the model's structure. The initial accuracy obtained in testing was around 11%. After adding the dropout layer, the result was increased to 14%. Pictured below is the final structure of the model.

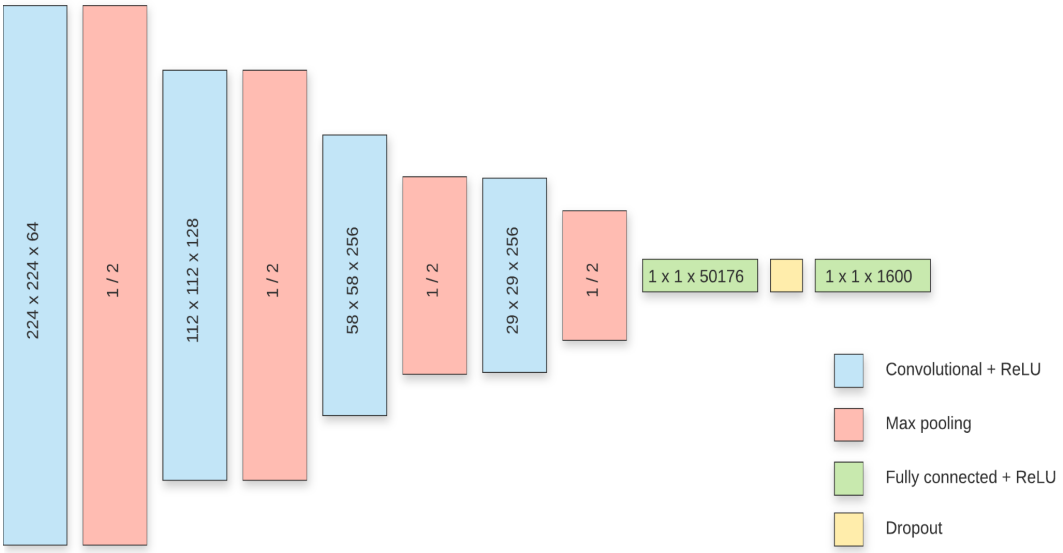

Fig. 7. 2D representation of the breed prediction scratch model architecture.

IV. Results

a. Model Evaluation and Validation

Once again, this section will be divided into the previously mentioned detection and prediction sections.

I. Detecting Humans

Here, performance was measured in accuracy, testing if the algorithm detected humans in provided images of humans. Since we are using a powerful pre-trained model, we should expect a reasonably high accuracy, and we got 98% in a subset of 100 images, which agrees with our expectations.

Secondly, the algorithm was run on images of dogs, so a low percentage of face detections should be expected. Of course there is always the possibility that there are human faces present in the images, but this shouldn't affect the result by much. The percentage of face detected was 17%. This is probably because the algorithm is more optimized towards accuracy, rather than precision or recall (which means that it will tolerate a few misclassifications to prevent missing a face when it is there).

II. Detecting Dogs

Similar to the human detector evaluation, the model was tested on 100 human images and 100 dog images. Since the powerful VGG-16 is being used here, a very high accuracy would be expected on the images of dogs. An extremely high accuracy was obtained on the dog images, namely 100%. And while classifying the human images, only one was misclassified as being a dog. The following image shows the image that was misclassified, and it is possible to see why that might be (Fig 8).

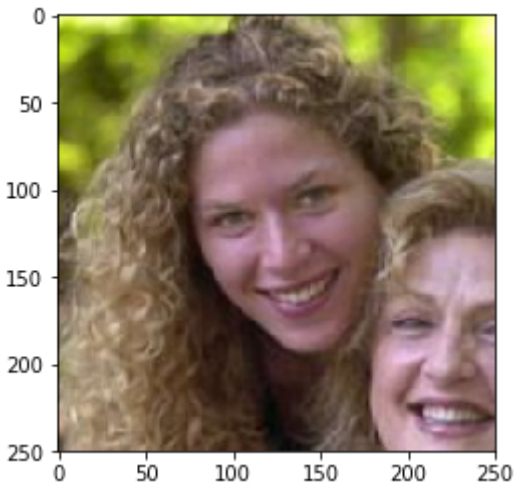

Fig. 8. Image that was misclassified as being a dog.

III. Predicting breeds (From Scratch)

Since the task of differentiating dog breeds is difficult even for a human, the threshold of acceptance in this task is extremely low, at about a minimum of 10% accuracy. The model was tested on the test dataset, which was already separated from the training and validation sets. The obtained accuracy was approximately 11% (before adding the dropout layer), which is acceptable, given the incredible complexity of the problem.

IV. Predicting breeds (Using transfer learning)

The model was run on 30 epochs, as to minimize the validation error as much as possible. As one might expect, using the ResNet-50 model did not only train much faster (taking into account that it was run on double the number of epochs), but it also produced significantly better results. It ended up with an accuracy of 75%, trumping the previously obtained 14%.

b. Justification

I. Detecting Humans

The accuracy was exceptionally high here because a highly optimized face recognition model was used, that had already been trained on a much larger number of samples.

II. Detecting Dogs

The accuracy was high in this model for the same reason the human detector achieved great results, it is an extremely robust model, trained on around 14 million images [12].

III. Predicting breeds (From Scratch)

The task of differentiating dog breeds is a very difficult one, and even humans may have trouble seeing the difference between some of them, so it is expected that a model built from scratch and trained on a limited number of images wouldn't present the best results. So the obtained result of 14% accuracy is not only anticipated but actually pretty acceptable in this scenario.

IV. Predicting breeds (Using transfer learning)

The ResNet-50 model uses residual mapping, as a means of being able to solve the issue of extremely deep NNs having worse performance than shallower ones, also called the degradation problem [13]. It was developed by a team of highly educated individuals, and has been tested and improved by its community for years. It is pretty clear why it would perform much better, and it came close enough to our benchmark model's accuracy, which means we achieved our goal.

V. Conclusion

a. Feature Visualization

Upon analysing some of the algorithm's results, we can find some interesting conclusions of how the algorithm is able to make out patterns on the images of humans, to determine similarities between specific complexions and certain dog breeds. On most classifications of dog breeds on human faces, the patterns are maybe too cloudy and not much similarity can be seen. But on some, especially the ones where hair is featured more prominently, the dog breed classification suddenly doesn't feel so random anymore. In the image seen below (Fig. 9), which was already mentioned earlier, the woman's curly hair resembles the furry head of a American water spaniel, which is what the model classifies the human as being most similar to.

Fig. 9. Similarities between human hair and dog's fur.

This gives us a little visual insight into the kinds of features the algorithm is looking for when deciding whether a given image gets classified as a certain dog breed.

b. Reflection

This project has been able to solve a very interesting problem, by exploring two different approaches. If expanded and modeled more like VGG-16, the scratch model could potentially achieve much better results. But then it would make more sense to use VGG itself as a transfer learning model.

One interesting note is that the final model could easily be used for other kinds of image classification. The human face predictor might be usable only to detect faces, but the ResNet transfer learning model could be trained on other datasets, given a few modifications, providing the foundations of a solid multi-purpose image classifier.

c. Improvement

This project could be improved in many ways. One of the ways, more on the technical side, would be to attach further convolutional layers to the transfer learning (resnet) model, to increase its complexity, and help it find even more complex patterns on the images. Also, the model did not achieve it's benchmark goal entirely, so a few adjustments might be made to make the algorithm reach closer to its goal.

The transfer model VGG-16, was considered to be used instead of ResNet-50, since it inspired the scratch model. It was tested on the data and achieved relatively poor results. Perhaps there were some issues not correctly taken into account there, and maybe the model could perform much better given some adjustments.

Another interesting improvement, that could extend the different uses of the model, is to create a model that can try to merge a human face with the breed of the dog it has identified the face as being similar to. Essentially a more complex kind of dog filter, which is able to take advantage of a person's facial features. An example of what that might look like is pictured below (Fig. 10).

Fig. 10. Human face merged into a dog.

References

[1] Fédération Cynologique Internationale (FCI):
http://www.fci.be/en/

[2] Haar feature-based cascade classifiers:
https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html

[3] ResNet-50 convolutional neural network:
https://www.mathworks.com/help/deeplearning/ref/resnet50.html;jsessionid=652a8f1b9f7738fff4da167599bb

[4] Dog Breed Identification:
https://web.stanford.edu/class/cs231a/prev_projects_2016/output%20(1).pdf

[5] OpenCV - Cascade Classifier documentation:
https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html

[6] Very deep convolutional networks for large-scale image recognition:
https://arxiv.org/pdf/1409.1556v6.pdf

[7] ImageNet dataset:
https://www.image-net.org/

[8] Pytorch Documentation:
https://pytorch.org/vision/stable/transforms.html

[9] ImageNet category dictionary:
https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a

[10] Deep Residual Learning for Image Recognition:
https://arxiv.org/abs/1512.03385v1

[11] ImageNet Large Scale Visual Recognition Challenge 2015 (ILSVRC2015):
https://image-net.org/challenges/LSVRC/2015/

[12] Hands-on Transfer Learning with Keras and the VGG16 Model:
https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/#:~:text=VGG16%20is%20a%20convolutional%20neural,for%20Large%2DScale%20Image%20Recognition.

[13] What exactly is the degradation problem that Deep Residual Networks try to alleviate?:
https://www.quora.com/What-exactly-is-the-degradation-problem-that-Deep-Residual-Networks-try-to-alleviate