Machine Learning Engineer Nanodegree Capstone Report:
Dog Breed Classifier using CNN

Nurbol Sakenov
August 10th, 2020

1. Definition

1.1 Project Overview
The interaction between humans and dogs can be traced as early as back to 9000 years ago [1]. With dogs being used for various purposes such as herding, guarding, scent detection, and hunting, different types of dogs have been breeded to fulfill these purposes. As of today, there are around 340 dog breeds known in the world [2].
With such a variety of dog breeds around the world, it is a challenge for humans to differentiate dog breeds from each other. This project aims to apply machine learning techniques to classify different breeds of dogs.

1.2 Problem Statement
The challenge created by a variety of dog breeds creates the opportunity for applying machine learning methods to solve it. In machine learning, image classification is accomplished by extracting key features from an image and supplying the features as inputs to the supervised model.
Convolutional neural networks (CNN) have gained a significant interest recently as it provides excellent results in image classification. The goal of the project is to create a CNN machine learning model to classify a dog breed given a real world image of a dog. In addition, if the model is supplied with an image of a human, the model should provide a resembling dog breed.

1.3 Metrics
The evaluation metric will be accuracy, which is calculated by comparing predictions to the true label and dividing the count of total correct results by the total number of predictions. The formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

TP - true positive

TN - true negative
FP - false positive
FN - false negative

2. Analysis

2.1. Data Exploration

The project dataset is provided by Udacity. Dog images dataset contains 8351 images of 133 dog breeds, from which there are 6680 training images, 835 validation images, 836 test images. The images in the dog dataset have mean height of 528 and mean width of 567; before building the model from scratch, the images will be resized.

```
*-----------------------------------------------------------------------------*
number of images                          |  8351

dtype                                     |  uint8
channels                                  |  [3]
extensions                                |  ['jpg']

min height                                |  113
max height                                |  4003
mean height                               |  529.0449048018202
median height                             |  467

min width                                 |  105
max width                                 |  4278
mean width                                |  567.0325709495869
median width                              |  500

mean height/width ratio                   |  0.9330062008886891
median height/width ratio                 |  0.934
recommended input size(by mean)           |  [528 568] (h x w, multiples of 8)
recommended input size(by mean)           |  [528 560] (h x w, multiples of 16)
recommended input size(by mean)           |  [544 576] (h x w, multiples of 32)

channel mean(0~1)                         |  [ 0.48671097  0.46614197  0.39680353]
channel std(0~1)                          |  [ 0.26631421  0.26078537  0.26648965]
*-----------------------------------------------------------------------------*
eda ended in 00 hours 01 minutes 59 seconds
```

Figure 1: Dog image dataset EDA summary

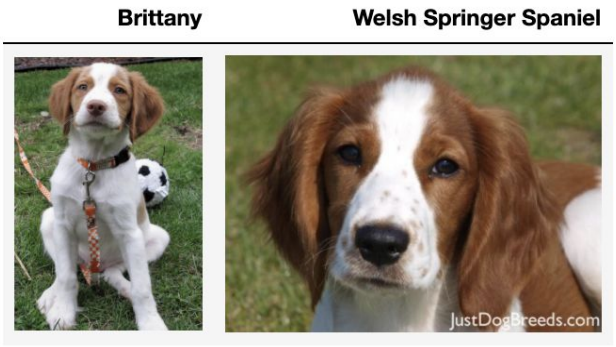

Figure 2: Sample images from dog dataset

There are 13233 images in total in the human images dataset. Compared to images in the dog dataset, it can be observed (figure 3) that the images in the human dataset have the same dimension of 250x250 within the entire dataset.

```
*-------------------------------------------------------------------------------*
number of images                          |  13233

dtype                                     |  uint8
channels                                  |  [3]
extensions                                |  ['jpg']

min height                                |  250
max height                                |  250
mean height                               |  250.0
median height                             |  250

min width                                 |  250
max width                                 |  250
mean width                                |  250.0
median width                              |  250

mean height/width ratio                   |  1.0
median height/width ratio                 |  1.0
recommended input size(by mean)           |  [248 248] (h x w, multiples of 8)
recommended input size(by mean)           |  [256 256] (h x w, multiples of 16)
recommended input size(by mean)           |  [256 256] (h x w, multiples of 32)

channel mean(0~1)                         |  [ 0.43918329  0.3830736   0.34242591]
channel std(0~1)                          |  [ 0.29703313  0.27359733  0.26828519]
*-------------------------------------------------------------------------------*
eda ended in 00 hours 00 minutes 27 seconds
```

Figure 3: Human faces dataset EDA summary

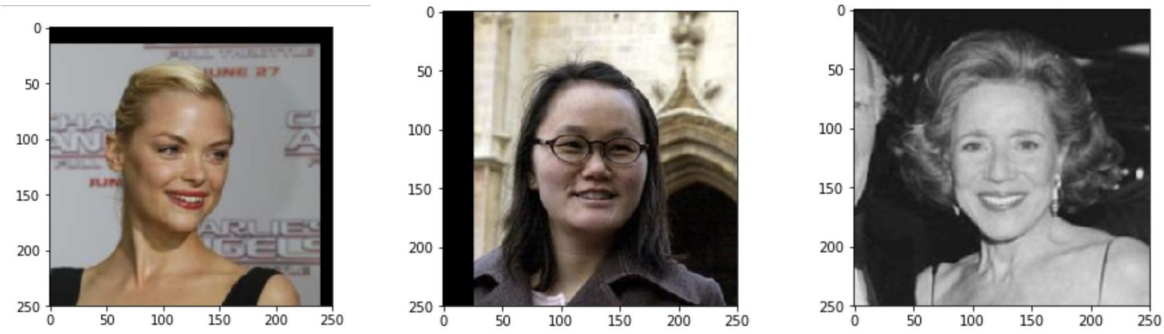

Figure 4: Sample images from human faces dataset

2.2 Algorithms and Techniques

One way to solve the dog classification problem is using CNN. CNNs belong to the class of neural networks commonly used in problems such as image classification, video recognition, and recommendation systems. The architecture of a CNN model has main input and output layers, in addition there are multiple hidden layers [3]. Due to their structure and high accuracy CNNs are commonly used in the image classification field.

In the project, the solution will involve the following steps:

1. In order to detect human images, OpenCV's implementation of Haar feature based cascade classifiers will be used.
2. In order to detect dog images, pretrained VGG16 model will be used.
3. CNN from scratch: the image will be supplied to the CNN model which will assign the best matching dog breed out of 133 dog breeds. The target is to achieve the accuracy of greater than 10%.
4. CNN using transfer learning: the model will be trained using transfer learning based step 3.

2.3 Benchmark

The benchmark model will be a simple CNN classifier created from scratch, which is expected to have accuracy of at least 10%. Then the model will be improved by learning on the train dataset and using transfer learning. After testing, the trained CNN model should have accuracy of at least 60% and above.

3. Methodology

3.1 Data Preprocessing

The training images are randomly rotated and horizontally flipped. Finally, the images from both datasets are resized to 224x224, transformed to tensors, and normalized.

```
#transforms
transform_train = transforms.Compose([transforms.Resize(224),
                                      transforms.RandomRotation(20),
                                      transforms.CenterCrop(224),
                                      transforms.RandomHorizontalFlip(),
                                      transforms.ToTensor(),
                                      transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])])

transform_test = transforms.Compose([transforms.Resize(224),
                                      transforms.CenterCrop(224),
                                      transforms.ToTensor(),
                                      transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])])
```

Figure 5: Data preprocessing applied to images

3.2 Human Face Detector

OpenCV's implementation of Haar feature-based cascade classifiers [8] is used to detect human faces in images. There are many pre-trained face detectors provided by OpenCV, the one used for this project is provided in the directory */haarcascades.* Before providing an image to the face detector, the image is converted to grayscale. After building the face detector, the classifier has been tested with one hundred human and dog images. Overall, the classifier correctly identified human faces with 99% accuracy and classified incorrectly 14% of dog images.

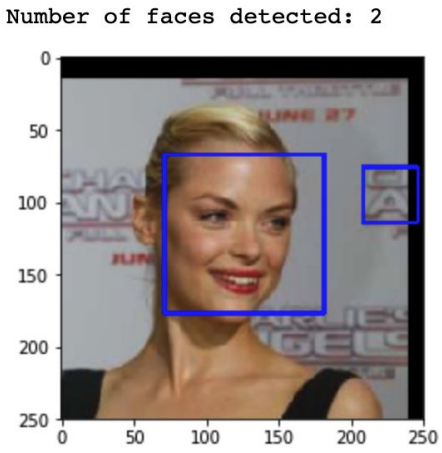

Figure 6: Human face detector result

3.3 Dog Face Detector

A pre-trained VGG-16 model is used to detect dog images. The model with weights is obtained from ImageNet [9], a popular database used for image recognition tasks. In total, the ImageNet database contains over 10 million images with URLs. When provided an image, the VGG-16 model returns an output that corresponds to 1000 categories from which dog images are classified within the range of 151 and 268

inclusive. Overall, the model identified 0% of human images as dogs. 100% of dog images have been correctly identified.

3.4 Implementing CNN from scratch

As a benchmark model, the CNN model with three convolutional layers have been built. Each convolutional layer has three kernels and one stride. The final convolutional layer has the output of size 128. The activation function is chosen as ReLU because it does not saturate and activate all the neurons at the same time [5]. Therefore, the ReLU activation is computationally more efficient in comparison to other activation functions such as sigmoid or tanh. The dropout rate of 0.2 is added to avoid overfitting. The final output size is 133.

In order to measure performance of the classifier during training, Cross-Entropy [6] was selected as a loss function. The function is easy to interpret: the loss increases as the more predicted value diverges from the actual label. The formula:

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

where:

- M - number of classes (dog, cat, fish)
- log - the natural log
- y - binary indicator (0 or 1) if class label c is the correct classification for observation o
- p - predicted probability observation o is of class c

After training the scratch CNN model on 25 epochs, the model could achieve the accuracy of 20%, which fulfills the initial requirement that had been set as 10%.

3.5 Transfer learning

The final model architecture was chosen as ResNet 101 based on the study from [7]. The study suggested that the deeper ResNet achieves a better result compared to the shallow network ResNet-50. However, it can also be observed in figure 5 that the improvement gain between ResNet-101 and ResNet-152 is low.

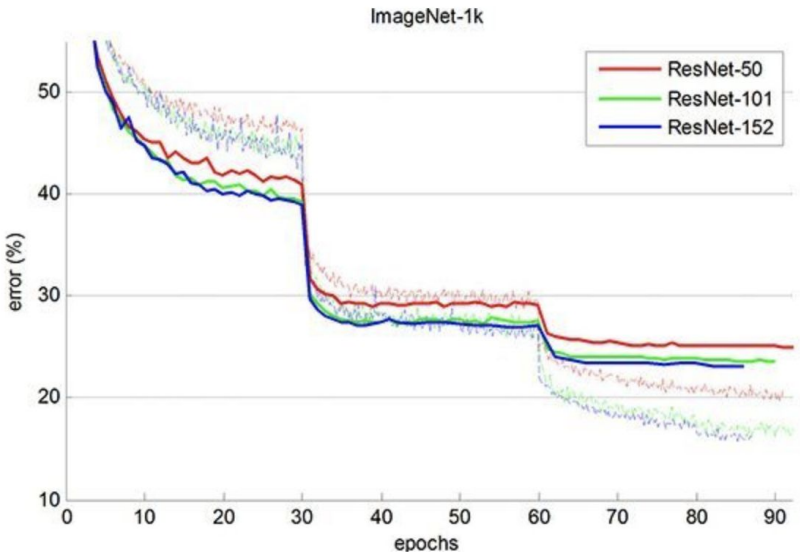

Figure 7: Performance comparison ResNet-50, ResNet 101, ResNet-152. [7]

After importing the pre-trained model defined in the chapter 3.4 using transfer learning, the following steps were followed:

- changing the output features to 133 to predict dog breeds
- assigning CrossEntropyLoss as loss function
- Setting the number of epochs to train the model as 7

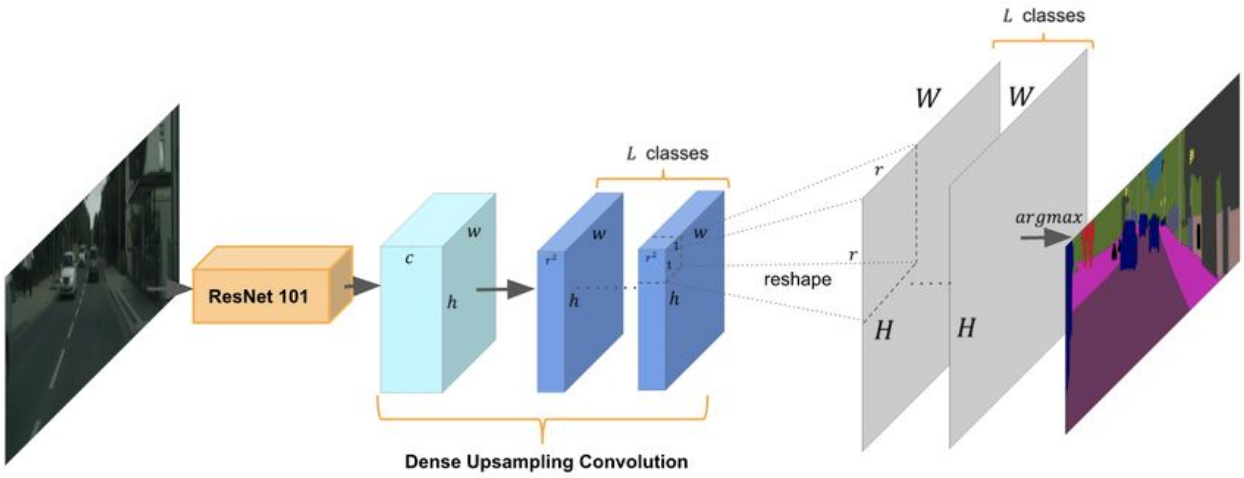

Figure 8: ResNet 101 architecture [10]

4. Results & Conclusions

4.1 Model Evaluation

The ResNet-101 that was trained using transfer learning as the final model had achieved the accuracy of 85%, which surpasses the target of 60%. The model had been

trained with 7 epochs, training loss: 0.549016 and validation loss: 0.539660. Below are sample images from the model output.

Figure 9: Final model results with human images

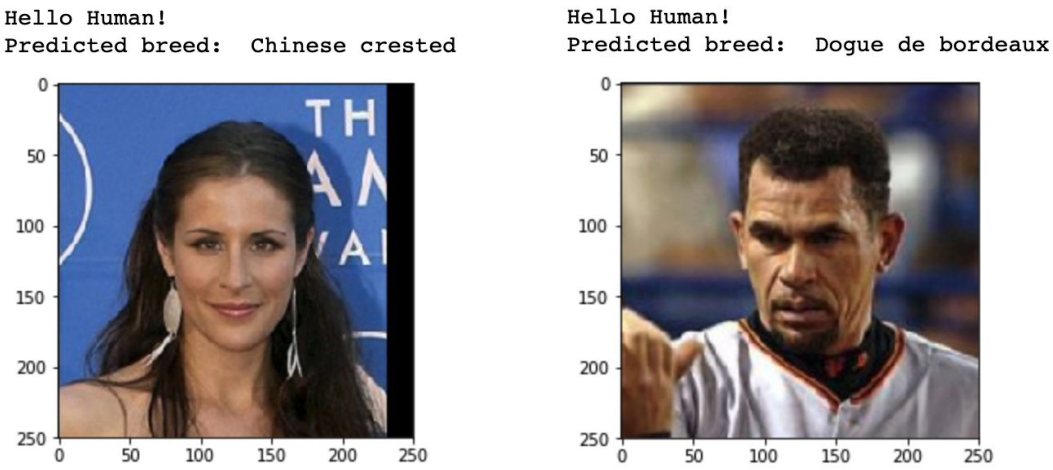

Figure 10: Final model results with human images

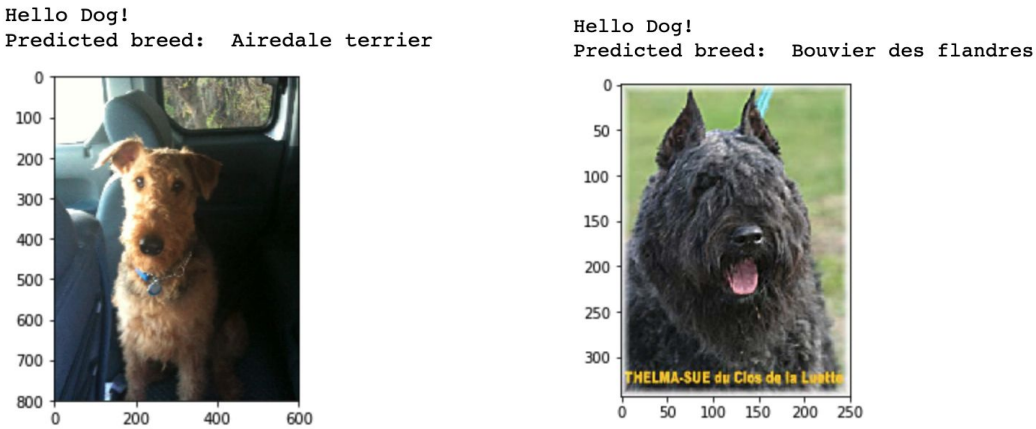

Figure 11: Final model results with dog images

4.2 Justification

The author believes that the accuracy of the final model can still be improved by tuning and training with more epochs. However, even with 7 epochs the final was able to surpass the target accuracy of 60%.

4.3 Improvement

Developing the project in a more modular way and using python files as imports could significantly improve readability and future maintenance of the project. The accuracy can still be improved by more training epochs and tuning of the model. During tests, the model predicted different dog breeds for the same person which could be avoided by providing a percentage similarity to different dog breeds.

Bibliography

[1] *“Dog Breed.”* Wikipedia, Aug. 2020, en.wikipedia.org/wiki/Dog_breed.

[2] *“Dog Breeds - Types Of Dogs*.” American Kennel Club, www.akc.org/dog-breeds/.

[3] “*Convolutional Neural Network.*” Wikipedia, 2 Aug. 2020, en.wikipedia.org/wiki/Convolutional_neural_network.

[4] Y. LeCun, Y. Bengio, et al. “Convolutional Neural Networks: an Overview and Application in Radiology.” *Insights into Imaging*, SpringerOpen, 1 Jan. 1970, link.springer.com/article/10.1007/s13244-018-0639-9.

[5] *“Activation Functions: Fundamentals Of Deep Learning.” Analytics Vidhya*, 19 July 2020, www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/.

[6] *Loss Functions - ML Glossary Documentation*, ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html.

[7] *ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks*. 25 Jan. 2019, neurohive.io/en/popular-networks/resnet/.

[8] *OpenCV:* Face detection using haar cascades. (n.d.). OpenCV documentation index. https://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

[9] *ImageNet*. https://www.image-net.org/

[10] *Understanding convolution for semantic segmentation.* (2017, February 27). https://www.researchgate.net/publication/314115448_Understanding_Convolution_for_Semantic_Segmentation