Machine Learning Engineer Nanodegree

Capstone Project

Daisuke Ohba
May 23th, 2020

Project Definition

Project Overview

Some breeds have been established by breeders, enthusiasts, breed clubs, etc., as "breed standards" (standards) in order to maintain and reinforce a certain appearance and traits through the generations, and there are hundreds of them.

Until now, we have had to ask an expert to know the exact breed of our dog. However, with the help of machine learning, which has evolved by leaps and bounds in the last few years, we can easily identify the breed with a single photo of our dog.

This is the assignment that I chose for my Capstone project for "Udacity Machine Learning Engineer Nanodegree". (CNN Project Dog Breed Classifier)

Datasets and Inputs are provided in the Notebook by Udacity. We will download and use them. Also, these are images of dogs or humans, labeled.

Images of dogs

- 8351 images of dogs (train: 6680, test: 836, valid: 835)
- They are stored in folders named by breed, and there are 133 breeds.
- RGB color mode
- The sizes are not uniform, so use them after resizing.

Images of humans

- 13234 images of humans (5750 people)
- They are stored in folders named by their name.
- RGB color mode
- The sizes are uniform. (250 x 250)

Problem Statement

Identify the input image and output the following. Concretely, it outputs the following according to the input image.

- if a dog is detected in the image, return the predicted breed.
  Machine learning classifications (unsupervised) are used to identify dog breeds.
- if a human is detected in the image, return the resembling dog breed.
  For human detection, we use OpenCV.

- if neither is detected in the image, provide output that indicates an error.

Metrics

"F-measure" is used for comprehensive evaluation of precision and recall. It is calculated by the harmonic mean of precision and recall.

$$ Fmeasure = \frac{2 \times precision \times recall}{precision + recall} $$

Analysis

Data Exploration

Datasets are provided in the Notebook by Udacity.
We will download and use them. However if we are using the Udacity workspace, we don't need to re-download these - they can be found in the specified folder.

These are images of dogs or humans, labeled.

Images of dogs

- 8351 images of dogs (train: 6680, valid: 835, test: 836)
- They are stored in folders named by breed, and there are 133 breeds.
- RGB color mode
- The sizes are not uniform, so use them after resizing.



Images of humans

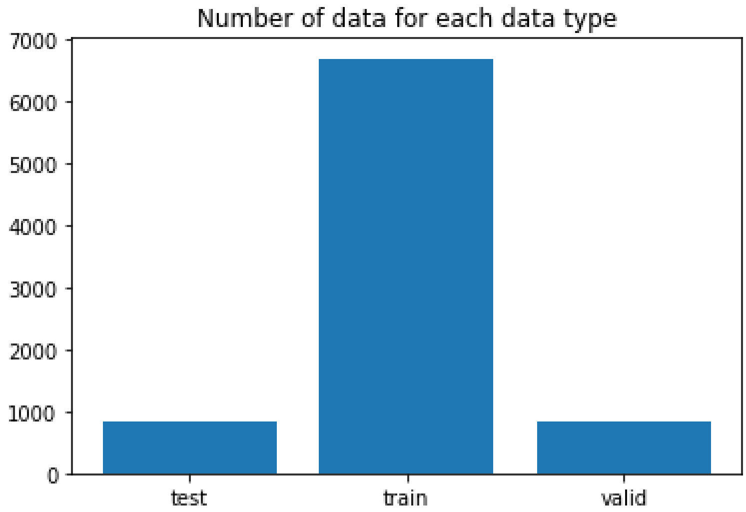
- 13233 images of humans
- They are stored in folders named by their name, and there are 5750 people.
- RGB color mode
- The sizes are uniform. (250 x 250)



Exploratory Visualization

Number of data for each data type

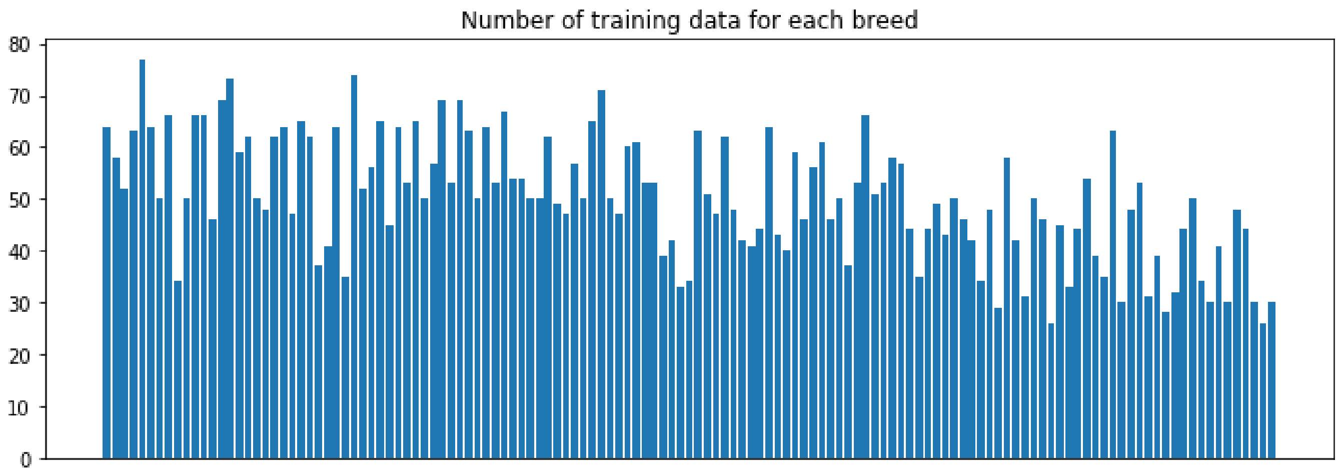
The amount of data is sufficient. The number of training data, evaluation data, and test data are well balanced.



Number of training data for each breed

There is a minimum of 26 training data for each breed. Sufficient for training.

Minimum: 26
Maximum: 77



Algorithms and Techniques

This project uses machine learning (supervised) to classify dog breeds. Specifically, transfer learning is performed based on the Pre-trained VGG16 Model. The VGG16 model is a CNN model consisting of 16 layers trained on a large-scale image dataset called "ImageNet", which was proposed by a research group at Oxford University and has achieved good results in 2014 ILSVR, which is a model suitable for image classification.

Benchmark

The results of the model created from the scratch benchmark are below.

```
Class   TP   TN   FP   TN Precision Recall F measure
    0    1    7   13  815 0.071429 0.125000 0.090909
    1    0    8    6  822 0.000000 0.000000 0.000000
    2    0    6    2  828 0.000000 0.000000 0.000000
    3    2    6    3  825 0.400000 0.250000 0.307692
    4    4    6   13  813 0.235294 0.400000 0.296296
    5    3    5   15  813 0.166667 0.375000 0.230769
    6    1    6    7  822 0.125000 0.142857 0.133333
    7    1    7    9  819 0.100000 0.125000 0.111111
    8    0    4    1  831 0.000000 0.000000 0.000000
    9    1    5    6  824 0.142857 0.166667 0.153846
   10    1    8    6  821 0.142857 0.111111 0.125000
   11    0    9    3  824 0.000000 0.000000 0.000000
   12    0    6    4  826 0.000000 0.000000 0.000000
   13    3    6   24  803 0.111111 0.333333 0.166667
   14    3    7   16  810 0.157895 0.300000 0.206897
   15    2    6    6  822 0.250000 0.250000 0.250000
   16    1    7    6  822 0.142857 0.125000 0.133333
   17    1    6    2  827 0.333333 0.142857 0.200000
   18    1    5    3  827 0.250000 0.166667 0.200000
   19    0    8    5  823 0.000000 0.000000 0.000000
   20    0    8    1  827 0.000000 0.000000 0.000000
   21    1    5   15  815 0.062500 0.166667 0.090909
   22    3    5    7  821 0.300000 0.375000 0.333333
   23    0    8    5  823 0.000000 0.000000 0.000000
   24    1    3    4  828 0.200000 0.250000 0.222222
   25    1    4   11  820 0.083333 0.200000 0.117647
   26    2    6   17  811 0.105263 0.250000 0.148148
   27    0    4    1  831 0.000000 0.000000 0.000000
   28    4    6    5  821 0.444444 0.400000 0.421053
   29    2    5    3  826 0.400000 0.285714 0.333333
   30    0    7    8  821 0.000000 0.000000 0.000000
   31    3    5    7  821 0.300000 0.375000 0.333333
   32    1    4    6  825 0.142857 0.200000 0.166667
   33    2    6   13  815 0.133333 0.250000 0.173913
   34    1    5    4  826 0.200000 0.166667 0.181818
   35    0    8    7  821 0.000000 0.000000 0.000000
   36    1    5    8  822 0.111111 0.166667 0.133333
   37    0    7    4  825 0.000000 0.000000 0.000000
   38    2    7    6  821 0.250000 0.222222 0.235294
   39    1    6    4  825 0.200000 0.142857 0.166667
   40    0    9   10  817 0.000000 0.000000 0.000000
   41    0    8    1  827 0.000000 0.000000 0.000000
   42    0    6    1  829 0.000000 0.000000 0.000000
   43    1    7    7  821 0.125000 0.125000 0.125000
   44    0    7    3  826 0.000000 0.000000 0.000000
   45    0    9   14  813 0.000000 0.000000 0.000000
   46    1    6    1  828 0.500000 0.142857 0.222222
   47    0    7    1  828 0.000000 0.000000 0.000000
   48    0    6    5  825 0.000000 0.000000 0.000000
   49    0    6    2  828 0.000000 0.000000 0.000000
   50    3    5    7  821 0.300000 0.375000 0.333333
   51    0    6    2  828 0.000000 0.000000 0.000000
```

```
      52    0    6    5  825 0.000000 0.000000 0.000000
      53    1    6    9  820 0.100000 0.142857 0.117647
      54    0    7    9  820 0.000000 0.000000 0.000000
      55    0    9    9  818 0.000000 0.000000 0.000000
      56    7    2   19  808 0.269231 0.777778 0.400000
      57    2    5    4  825 0.333333 0.285714 0.307692
      58    0    6    2  828 0.000000 0.000000 0.000000
      59    2    6    6  822 0.250000 0.250000 0.250000
      60    0    8    1  827 0.000000 0.000000 0.000000
      61    0    6    3  827 0.000000 0.000000 0.000000
      62    0    7    7  822 0.000000 0.000000 0.000000
      63    0    5    2  829 0.000000 0.000000 0.000000
      64    0    5    0  831 0.000000 0.000000 0.000000
      65    0    4    1  831 0.000000 0.000000 0.000000
      66    0    4   13  819 0.000000 0.000000 0.000000
      67    0    8    3  825 0.000000 0.000000 0.000000
      68    0    7    6  823 0.000000 0.000000 0.000000
      69    1    5   10  820 0.090909 0.166667 0.117647
      70    0    8   24  804 0.000000 0.000000 0.000000
      71    0    6    7  823 0.000000 0.000000 0.000000
      72    0    5    3  828 0.000000 0.000000 0.000000
      73    1    4    0  831 1.000000 0.200000 0.333333
      74    0    5    2  829 0.000000 0.000000 0.000000
      75    0    8    1  827 0.000000 0.000000 0.000000
      76    1    4   13  818 0.071429 0.200000 0.105263
      77    0    5    1  830 0.000000 0.000000 0.000000
      78    1    7   15  813 0.062500 0.125000 0.083333
      79    0    5    5  826 0.000000 0.000000 0.000000
      80    1    6    3  826 0.250000 0.142857 0.181818
      81    0    8    2  826 0.000000 0.000000 0.000000
      82    4    2    4  826 0.500000 0.666667 0.571429
      83    0    6    8  822 0.000000 0.000000 0.000000
      84    0    4    0  832 0.000000 0.000000 0.000000
      85    2    5   17  812 0.105263 0.285714 0.153846
      86    0    8   12  816 0.000000 0.000000 0.000000
      87    3    3    8  822 0.272727 0.500000 0.352941
      88    1    6    2  827 0.333333 0.142857 0.200000
      89    0    8    2  826 0.000000 0.000000 0.000000
      90    2    5    3  826 0.400000 0.285714 0.333333
      91    0    5    7  824 0.000000 0.000000 0.000000
      92    0    4    1  831 0.000000 0.000000 0.000000
      93    3    2    5  826 0.375000 0.600000 0.461538
      94    0    6    5  825 0.000000 0.000000 0.000000
      95    0    5    1  830 0.000000 0.000000 0.000000
      96    0    6    0  830 0.000000 0.000000 0.000000
      97    0    5    6  825 0.000000 0.000000 0.000000
      98    0    5    7  824 0.000000 0.000000 0.000000
      99    0    4    2  830 0.000000 0.000000 0.000000
     100    1    5    5  825 0.166667 0.166667 0.166667
     101    0    3    3  830 0.000000 0.000000 0.000000
     102    0    7    5  824 0.000000 0.000000 0.000000
     103    0    5    2  829 0.000000 0.000000 0.000000
     104    0    4    2  830 0.000000 0.000000 0.000000
     105    0    6    2  828 0.000000 0.000000 0.000000
```

```
    106    0    6    4  826 0.000000 0.000000 0.000000
    107    0    3    0  833 0.000000 0.000000 0.000000
    108    1    4    8  823 0.111111 0.200000 0.142857
    109    1    3    6  826 0.142857 0.250000 0.181818
    110    0    5    0  831 0.000000 0.000000 0.000000
    111    1    6    6  823 0.142857 0.142857 0.142857
    112    1    4    2  829 0.333333 0.200000 0.250000
    113    0    4    1  831 0.000000 0.000000 0.000000
    114    4    4   15  813 0.210526 0.500000 0.296296
    115    1    3    7  825 0.125000 0.250000 0.166667
    116    0    6    8  822 0.000000 0.000000 0.000000
    117    1    6    4  825 0.200000 0.142857 0.166667
    118    1    3    2  830 0.333333 0.250000 0.285714
    119    2    3    1  830 0.666667 0.400000 0.500000
    120    0    3    2  831 0.000000 0.000000 0.000000
    121    0    4    0  832 0.000000 0.000000 0.000000
    122    0    5    4  827 0.000000 0.000000 0.000000
    123    0    6    0  830 0.000000 0.000000 0.000000
    124    0    4    1  831 0.000000 0.000000 0.000000
    125    1    2    0  833 1.000000 0.333333 0.500000
    126    0    5    5  826 0.000000 0.000000 0.000000
    127    0    4    0  832 0.000000 0.000000 0.000000
    128    0    6    6  824 0.000000 0.000000 0.000000
    129    2    3   11  820 0.153846 0.400000 0.222222
    130    0    3    1  832 0.000000 0.000000 0.000000
    131    0    3    0  833 0.000000 0.000000 0.000000
    132    2    2    2  830 0.500000 0.500000 0.500000
```

- Accuracy: 12% (106/836)
- F measure: 0.104027

Methodology

Data Preprocessing

Since the sizes of the training data are not uniform, we resized with a square of 250 sides and crop the center with a square of 224 sides as preprocessing of the data. We also normalize using the average and standard values of Imagenet. These are common techniques that uses the pretraining model we use.

Implementation

Transfer learning based on the Pre-trained VGG16 Model.
Since the final linear regression layer is classified into 1000 classes, it is changed to a layer classified into 133 classes for breeds.

The modified model is shown below.

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace)
```

```
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU(inplace)
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU(inplace)
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27): ReLU(inplace)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace)
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (classifier): 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)
  )
)
```

The loss function is CrossEntropyLoss, and the optimization function is SGD. These are common choices in classification.

Refinement

To prevent over fitting, learning was stopped early when the evaluation value in the validation data began to deteriorate, and the best-performing parameters were adopted.

In the initial solution, when the evaluation loss increased, the training was interrupted to give the final model. In the final solution, the training was continued until the evaluation loss increased continuously (5 times).

```
Epoch: 1    Training Loss: 1.477154     Validation Loss: 0.623285
Validation loss decreased (inf --> 0.623285).  Saving model ...
Epoch: 2    Training Loss: 0.624448     Validation Loss: 0.513815
Validation loss decreased (0.623285 --> 0.513815).  Saving model ...
Epoch: 3    Training Loss: 0.434888     Validation Loss: 0.474862
Validation loss decreased (0.513815 --> 0.474862).  Saving model ...
Epoch: 4    Training Loss: 0.366307     Validation Loss: 0.445833
Validation loss decreased (0.474862 --> 0.445833).  Saving model ...
Epoch: 5    Training Loss: 0.285812     Validation Loss: 0.445519
Validation loss decreased (0.445833 --> 0.445519).  Saving model ...
Epoch: 6    Training Loss: 0.240889     Validation Loss: 0.438481
Validation loss decreased (0.445519 --> 0.438481).  Saving model ...
Epoch: 7    Training Loss: 0.202925     Validation Loss: 0.417384
Validation loss decreased (0.438481 --> 0.417384).  Saving model ...
Epoch: 8    Training Loss: 0.164683     Validation Loss: 0.488066
Epoch: 9    Training Loss: 0.149371     Validation Loss: 0.464932
Epoch: 10   Training Loss: 0.137464     Validation Loss: 0.413841
Validation loss decreased (0.417384 --> 0.413841).  Saving model ...
Epoch: 11   Training Loss: 0.119226     Validation Loss: 0.428468
Epoch: 12   Training Loss: 0.109266     Validation Loss: 0.463171
Epoch: 13   Training Loss: 0.097856     Validation Loss: 0.473736
Epoch: 14   Training Loss: 0.078440     Validation Loss: 0.424771
Epoch: 15   Training Loss: 0.075797     Validation Loss: 0.492575
```

Results

Model Evaluation and Validation

```
Class   TP   TN   FP   TN Precision Recall F measure
    0    8    0    0  828 1.000000 1.000000 1.000000
    1    8    0    0  828 1.000000 1.000000 1.000000
    2    6    0    2  828 0.750000 1.000000 0.857143
    3    7    1    0  828 1.000000 0.875000 0.933333
    4   10    0    2  824 0.833333 1.000000 0.909091
    5    7    1    0  828 1.000000 0.875000 0.933333
    6    4    3    2  827 0.666667 0.571429 0.615385
    7    8    0    0  828 1.000000 1.000000 1.000000
    8    3    1    1  831 0.750000 0.750000 0.750000
    9    5    1    0  830 1.000000 0.833333 0.909091
   10    7    2    0  827 1.000000 0.777778 0.875000
   11    7    2    0  827 1.000000 0.777778 0.875000
   12    5    1    0  830 1.000000 0.833333 0.909091
   13    8    1    1  826 0.888889 0.888889 0.888889
   14    9    1    0  826 1.000000 0.900000 0.947368
   15    6    2    1  827 0.857143 0.750000 0.800000
   16    7    1    1  827 0.875000 0.875000 0.875000
   17    7    0    0  829 1.000000 1.000000 1.000000
   18    6    0    0  830 1.000000 1.000000 1.000000
   19    8    0    3  825 0.727273 1.000000 0.842105
   20    8    0    0  828 1.000000 1.000000 1.000000
   21    6    0    0  830 1.000000 1.000000 1.000000
   22    8    0    0  828 1.000000 1.000000 1.000000
   23    6    2    0  828 1.000000 0.750000 0.857143
   24    3    1    0  832 1.000000 0.750000 0.857143
   25    3    2    0  831 1.000000 0.600000 0.750000
   26    8    0    2  826 0.800000 1.000000 0.888889
   27    3    1    0  832 1.000000 0.750000 0.857143
   28   10    0    0  826 1.000000 1.000000 1.000000
   29    7    0    0  829 1.000000 1.000000 1.000000
   30    6    1    0  829 1.000000 0.857143 0.923077
   31    8    0    0  828 1.000000 1.000000 1.000000
   32    5    0    0  831 1.000000 1.000000 1.000000
   33    8    0    1  827 0.888889 1.000000 0.941176
   34    4    2    2  828 0.666667 0.666667 0.666667
   35    8    0    0  828 1.000000 1.000000 1.000000
   36    4    2    0  830 1.000000 0.666667 0.800000
   37    7    0    0  829 1.000000 1.000000 1.000000
   38    9    0    0  827 1.000000 1.000000 1.000000
   39    7    0    0  829 1.000000 1.000000 1.000000
   40    8    1    2  825 0.800000 0.888889 0.842105
   41    8    0    1  827 0.888889 1.000000 0.941176
   42    6    0    3  827 0.666667 1.000000 0.800000
   43    7    1    1  827 0.875000 0.875000 0.875000
   44    4    3    0  829 1.000000 0.571429 0.727273
   45    7    2    3  824 0.700000 0.777778 0.736842
   46    7    0    1  828 0.875000 1.000000 0.933333
   47    6    1    0  829 1.000000 0.857143 0.923077
   48    6    0    0  830 1.000000 1.000000 1.000000
   49    5    1    0  830 1.000000 0.833333 0.909091
   50    8    0    1  827 0.888889 1.000000 0.941176
   51    5    1    1  829 0.833333 0.833333 0.833333
```

```
  52    5    1    2  828 0.714286 0.833333 0.769231
  53    6    1    2  827 0.750000 0.857143 0.800000
  54    7    0    1  828 0.875000 1.000000 0.933333
  55    8    1    1  826 0.888889 0.888889 0.888889
  56    8    1    0  827 1.000000 0.888889 0.941176
  57    7    0    0  829 1.000000 1.000000 1.000000
  58    4    2    2  828 0.666667 0.666667 0.666667
  59    8    0    1  827 0.888889 1.000000 0.941176
  60    3    5    0  828 1.000000 0.375000 0.545455
  61    5    1    2  828 0.714286 0.833333 0.769231
  62    6    1    4  825 0.600000 0.857143 0.705882
  63    2    3    0  831 1.000000 0.400000 0.571429
  64    5    0    1  830 0.833333 1.000000 0.909091
  65    3    1    2  830 0.600000 0.750000 0.666667
  66    2    2    1  831 0.666667 0.500000 0.571429
  67    7    1    0  828 1.000000 0.875000 0.933333
  68    7    0    0  829 1.000000 1.000000 1.000000
  69    3    3    3  827 0.500000 0.500000 0.500000
  70    7    1    0  828 1.000000 0.875000 0.933333
  71    4    2    2  828 0.666667 0.666667 0.666667
  72    1    4    0  831 1.000000 0.200000 0.333333
  73    4    1    2  829 0.666667 0.800000 0.727273
  74    4    1    0  831 1.000000 0.800000 0.888889
  75    8    0    1  827 0.888889 1.000000 0.941176
  76    5    0    5  826 0.500000 1.000000 0.666667
  77    3    2    1  830 0.750000 0.600000 0.666667
  78    6    2    3  825 0.666667 0.750000 0.705882
  79    4    1    0  831 1.000000 0.800000 0.888889
  80    6    1    0  829 1.000000 0.857143 0.923077
  81    5    3    2  826 0.714286 0.625000 0.666667
  82    6    0    0  830 1.000000 1.000000 1.000000
  83    5    1    2  828 0.714286 0.833333 0.769231
  84    2    2    0  832 1.000000 0.500000 0.666667
  85    7    0    0  829 1.000000 1.000000 1.000000
  86    8    0    1  827 0.888889 1.000000 0.941176
  87    4    2    1  829 0.800000 0.666667 0.727273
  88    6    1    1  828 0.857143 0.857143 0.857143
  89    7    1    1  827 0.875000 0.875000 0.875000
  90    7    0    1  828 0.875000 1.000000 0.933333
  91    5    0    0  831 1.000000 1.000000 1.000000
  92    4    0    1  831 0.800000 1.000000 0.888889
  93    5    0    0  831 1.000000 1.000000 1.000000
  94    3    3    2  828 0.600000 0.500000 0.545455
  95    4    1    1  830 0.800000 0.800000 0.800000
  96    4    2    0  830 1.000000 0.666667 0.800000
  97    5    0    0  831 1.000000 1.000000 1.000000
  98    4    1    1  830 0.800000 0.800000 0.800000
  99    2    2    1  831 0.666667 0.500000 0.571429
 100    6    0    2  828 0.750000 1.000000 0.857143
 101    3    0    2  831 0.600000 1.000000 0.750000
 102    5    2    0  829 1.000000 0.714286 0.833333
 103    5    0    1  830 0.833333 1.000000 0.909091
 104    4    0    1  831 0.800000 1.000000 0.888889
 105    6    0    1  829 0.857143 1.000000 0.923077
```

```
    106    5    1    1  829 0.833333 0.833333 0.833333
    107    1    2    1  832 0.500000 0.333333 0.400000
    108    4    1    1  830 0.800000 0.800000 0.800000
    109    4    0    3  829 0.571429 1.000000 0.727273
    110    5    0    1  830 0.833333 1.000000 0.909091
    111    7    0    1  828 0.875000 1.000000 0.933333
    112    5    0    0  831 1.000000 1.000000 1.000000
    113    4    0    1  831 0.800000 1.000000 0.888889
    114    8    0    0  828 1.000000 1.000000 1.000000
    115    3    1    0  832 1.000000 0.750000 0.857143
    116    6    0    0  830 1.000000 1.000000 1.000000
    117    6    1    1  828 0.857143 0.857143 0.857143
    118    4    0    1  831 0.800000 1.000000 0.888889
    119    5    0    0  831 1.000000 1.000000 1.000000
    120    3    0    0  833 1.000000 1.000000 1.000000
    121    2    2    2  830 0.500000 0.500000 0.500000
    122    4    1    1  830 0.800000 0.800000 0.800000
    123    5    1    1  829 0.833333 0.833333 0.833333
    124    3    1    1  831 0.750000 0.750000 0.750000
    125    3    0    0  833 1.000000 1.000000 1.000000
    126    3    2    1  830 0.750000 0.600000 0.666667
    127    4    0    1  831 0.800000 1.000000 0.888889
    128    4    2    0  830 1.000000 0.666667 0.800000
    129    5    0    4  827 0.555556 1.000000 0.714286
    130    3    0    3  830 0.500000 1.000000 0.666667
    131    2    1    0  833 1.000000 0.666667 0.800000
    132    3    1    0  832 1.000000 0.750000 0.857143
```

- Accuracy: 86% (722/836)
- F measure: 0.847775

Justification

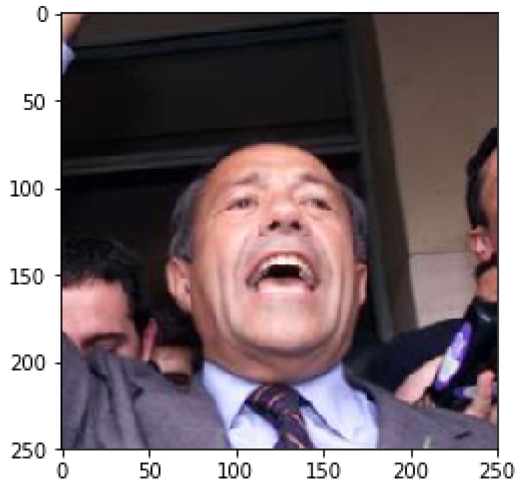
We were able to identify them with a very high degree of accuracy compared to the models we created from scratch. F-measure is also high, so both reproducibility and accuracy can be optimized.

The model created from scratch was not practical, but the final model turned out to be practical. Transition learning has been found to be very effective because it allows for the creation of highly accurate models in a short training time.

Finally, here are some of the test results. Correctly classified.

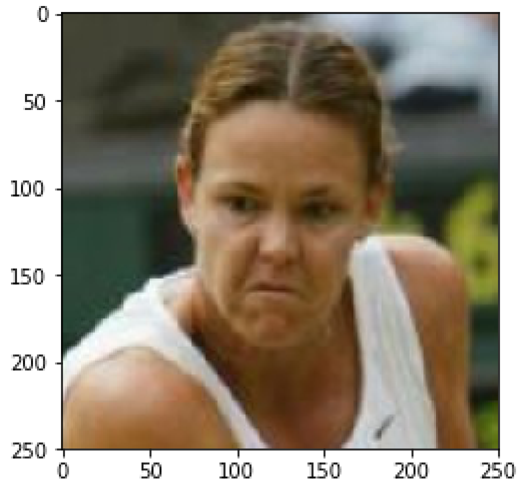
Hello, human!

You look like a ...
Welsh springer spaniel



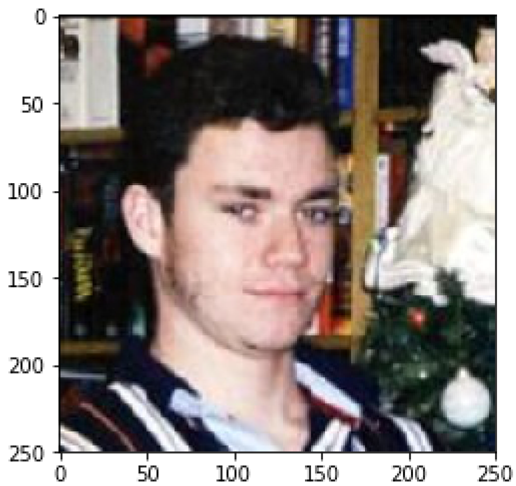
Hello, human!

You look like a ...
Wirehaired pointing griffon



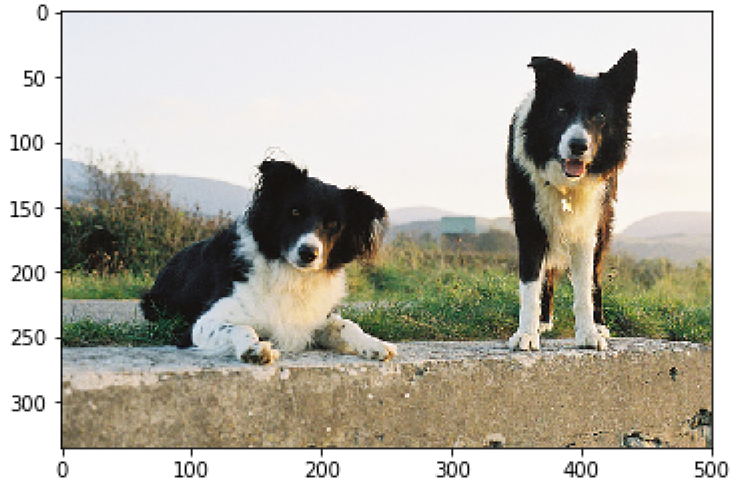
Hello, human!

You look like a ...
Chinese crested



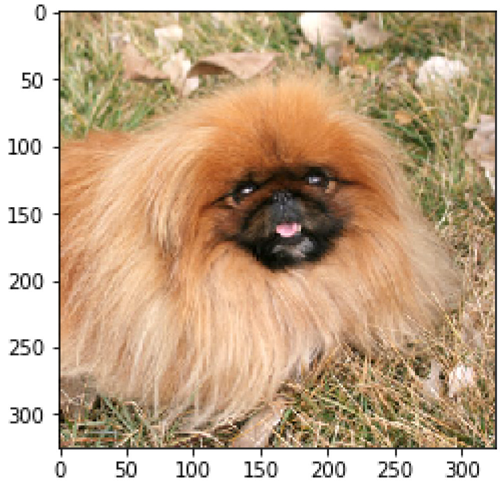
```
Hello, dog!

Your predicted breed is ...
Border collie
```




```
Hello, dog!

Your predicted breed is ...
Pekingese
```



```
Hello, dog!

Your predicted breed is ...
Lakeland terrier
```