Machine Learning Engineer

Nanodegree

Capstone Project Report

Dog Breed Classifier With CNNs

Ji-woong Choi

18th September 2020

Project Report

CONTENTS

1 DEFINITION

1.1 PROJECT OVERVIEW

Starting with AlexNet, which first appeared in 2012, the Convolutional Neural Network has grown very steeply, and now it is showing more than an expert beyond the ability to distinguish between ordinary people. As the research progressed repeatedly, networks that showed quite good performance in the classification field appeared, and after that, the data set itself became the criterion for determining the quality rather than the development of the network. This means that the dataset it self is a competitive era.

So, What I want to do is to classify dog breed. Because The dog breed classifier is a well-known problem in ML. It is the good starting point to understand classification problem. In this proposal, we propose a model that classifies breeds by CNN using the Dog Face and Human Face datasets.

1.2 PROBLEM STATEMENT

In this project it is important to build a dataset processing pipeline to classify real-world, user-supplied images. The algorithm will identify an estimate of the dog's breed given an image. It has to perform two tasks:

Dog Face Detector : Given an image of a dog, the algorithm will identify an estimate of the canine's breed.

Human Face Detector : If supplied an image of a human, the code will identify the resembling dog breed.

If neither of the two is detected, then an error message will be issued.

In addition, an accuracy of 75 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.

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 slight data imbalance in the dataset. Therefore I will use also the F1 score as an additional metric.

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{t}p}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}.$$

2 ANALYSIS

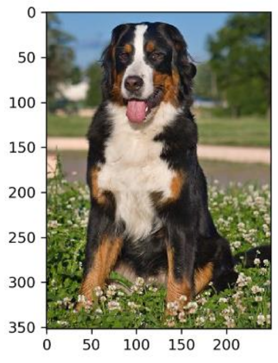
2.1 DATA EXPLORATION

All the datasets used to train, test, and validate the CNN model are provided by Udacity. The dataset consists of human and dog faces.

2.1.1 THE DOG DATASET

The dog image dataset has 8351 total images which are sorted into train (6,680 images), test (836 images) and valid (835 images) directories. Each of this directory (train, test, valid) have 133 folders corresponding to dog breeds. The images are of different sizes and different backgrounds, some images are not full-sized. The data is not balanced because the number of images provided for each breed varies. Few have 4 images while some have 8 images.

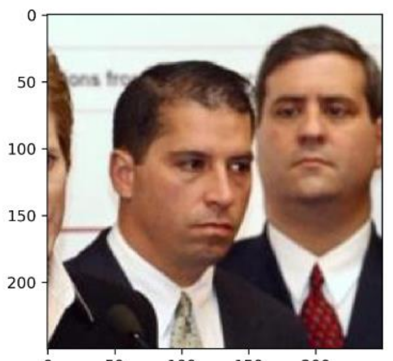
Chihuahua

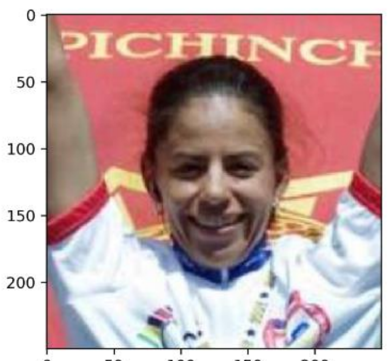


Bernese Mountain Dog

2.1.2 THE HUMAN DATASET

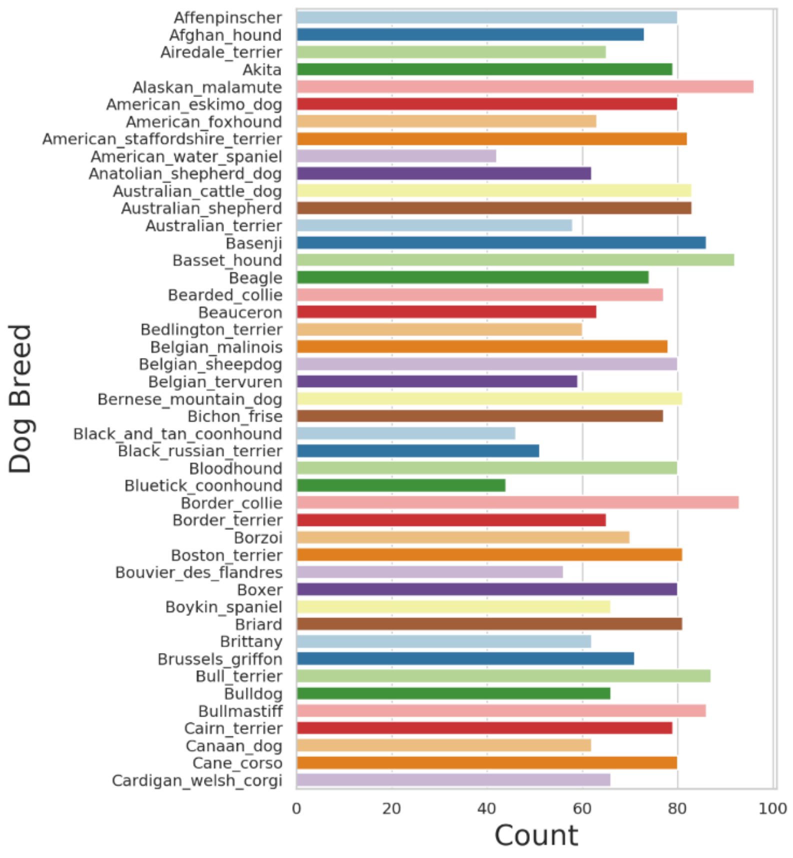
The human dataset contains 13,233 images. Here are some sample images.


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. This is a part of the distribution of the dog breeds over the complete dataset.



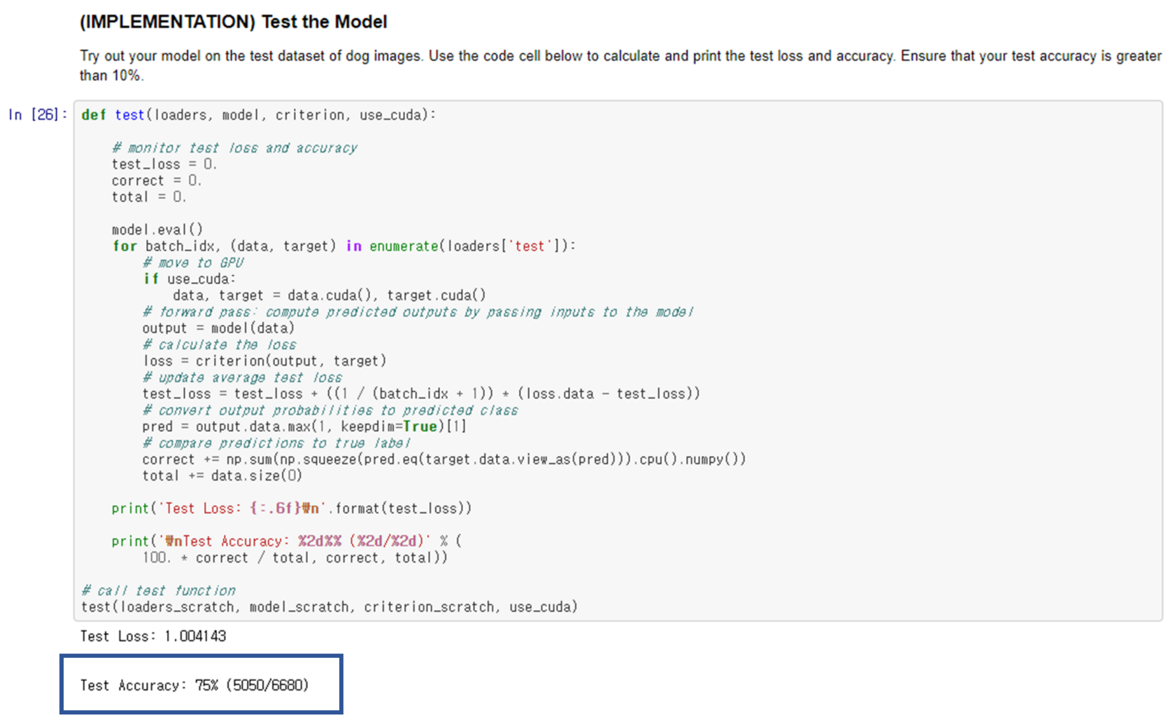
2.3 ALGORITHMS AND TECHNIQUES

For performing this multiclass classification, we can use Convolutional Neural Network (CNN) to solve the problem. A Convolutional Neural Network(CNN) is a

Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The solution involves three steps. First, to detect human images, we can use existing algorithm like OpenCV's implementation of Haar feature based cascade classifiers. Second, to detect dog-images we will use a pretrained ResNext101 model. Finally, after the image is identified as dog/human, we can pass this image to an CNN model which will process the image and predict the breed that matches the best out of 133 breeds.

2.4 BENCHMARK

For our benchmark model, The Convolutional Neural Networks (CNN) model created from scratch must have accuracy of at least 10%. This should be enough to confirm that our model is working because random guess would be 1 in 133 breeds which are less than 1% if we don't consider unbalanced data for our dog images.



(IMPLEMENTATION) Test the Model

Try out your model on the test dataset of dog images. Use the code cell below to calculate and print the test loss and accuracy. Ensure that your test accuracy is greater than 10%.

```
In [26]: def test(loaders, model, criterion, use_cuda):

             # monitor test loss and accuracy
             test_loss = 0.
             correct = 0.
             total = 0.

             model.eval()
             for batch_idx, (data, target) in enumerate(loaders['test']):
                 # move to GPU
                 if use_cuda:
                     data, target = data.cuda(), target.cuda()
                 # forward pass: compute predicted outputs by passing inputs to the model
                 output = model(data)
                 # calculate the loss
                 loss = criterion(output, target)
                 # update average test loss
                 test_loss = test_loss + ((1 / (batch_idx + 1)) * (loss.data - test_loss))
                 # convert output probabilities to predicted class
                 pred = output.data.max(1, keepdim=True)[1]
                 # compare predictions to true label
                 correct += np.sum(np.squeeze(pred.eq(target.data.view_as(pred))).cpu().numpy())
                 total += data.size(0)

             print('Test Loss: {:.6f}₩n'.format(test_loss))

             print('₩nTest Accuracy: %2d%% (%2d/%2d)' % (
                 100. * correct / total, correct, total))

         # call test function
         test(loaders_scratch, model_scratch, criterion_scratch, use_cuda)
```

```
Test Loss: 1.004143


Test Accuracy: 75% (5050/6680)
```

Anyway, This Benchmark model show me the result of 75% finally. It is also enough score.

3 METHODOLOGY

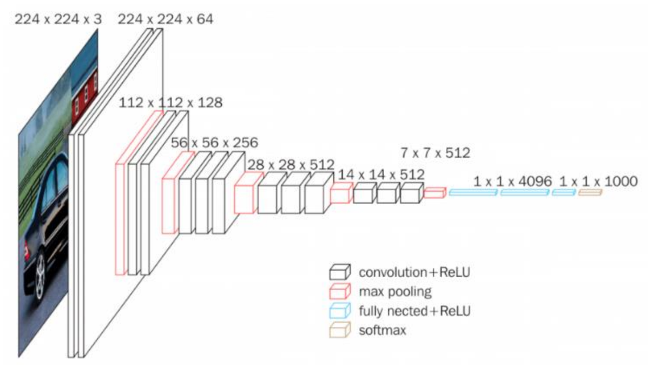
3.1 DATA PREPROCESSING

All the images are resized to 224 * 224, then normalization is applied to all images (train, valid and test datasets). For the training data, Image augmentation is done to reduce overfitting. The train data images are randomly rotated and random horizontal flip is applied. Finally, all the images are converted into tensor before passing into the model.

3.2 IMPLEMENTATION

I divide the project into 3 parts

- Pre-processing : In this step I imported all the required datasets and libraries. After that I will pre-process the data.
- Data Splitting : In this step I spliced the input data into train, validate and test datasets and then performed image augmentation on train/validate/test sets.
- Model training and evaluation: This step comprised of the following.
  - Created a dog detector using VGG16 model.



  - Trained, Tested and validated a CNN model built from scratch.

```
## Define CNN layers
self.conv1 = nn.Conv2d( 3, 32, 3, stride=1, padding=1)
self.conv2 = nn.Conv2d(32, 64, 3, stride=1, padding=1)
self.conv3 = nn.Conv2d( 64, 128, 3, stride=1, padding=1)
self.conv4 = nn.Conv2d(128, 128, 3, stride=1, padding=1)

# Pooling
self.pool = nn.MaxPool2d(2, 2)

# Define fully connected layers
self.fc1 = nn.Linear(14*14*128, 4096)
self.fc2 = nn.Linear(4096, 134)

# drop-out
self.dropout = nn.Dropout(0.25)
```

- Again created a CNN model, but this time using transfer learning with ResNext 101 model, and finally trained, tested and validated the data on this model.

```
model_transfer = models.resnext101_32x8d(pretrained=True)
model_transfer.add_module('drop', nn.Dropout(0.3))
model_transfer.add_module('fc1', nn.Linear(in_features=1000, out_features=134, bias=True))

# Freeze training for all parameters
for param in model_transfer.parameters():
    param.requires_grad = False

# Replacing the last 3 layers for fine tuning
# Parameters of newly constructed modules have requires_grad=True by default
model_transfer.fc = nn.Linear(2048, 1000, bias=True)
model_transfer.drop = nn.Dropout(0.3)
model_transfer.fc1 = nn.Linear(in_features=1000, out_features=134, bias=True)
```

- Evaluated the model by displaying of appropriate messages as output.

3.3 REFINEMENT

The CNN created from scratch have accuracy of 75%, Though it meets the benchmarking, the model can be significantly improved by using transfer learning. To create CNN with transfer learning, I have selected the ResNext101 architecture which is pre-trained on ImageNet dataset. The last convolutional output of ResNext101 is fed as input to our model. We only need to add a fully connected layer to produce 133-dimensional output. The model performed extremely well when compared to CNN from scratch.

```
test(loaders_transfer, model_transfer, criterion_transfer, use_cuda)
```

```
Test Loss: 0.501793


Test Accuracy: 86% (5773/6680)
```

Result of Transfer Learning of ResNext 101 model

4 RESULTS

4.1 MODEL EVALUATION AND VALIDATION

4.1.1 MODEL FROM SCRATCH

The final architecture and the hyperparameters were based on the maximum hardware available to me. Here I would like to discuss the selected hyperparameters:

- A learning rate of 0.03
- 100 epochs
- Stochastic Gradient Descent as optimizer.

4.1.2 MODEL WITH TRANSFER LEARNING

Here I would like to mention that the last fully connected layer has 133 output features, which represent the 133 dog breeds. Below is the selected hyperparameters:

- A learning rate of 0.01
- momentum of 0.9
- 40 epochs
- Stochastic Gradient Descent as optimizer.

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

The results obtained in testing the model are better than expected.

- Test Accuracy : 75.59%
- Test F1 Score : 0.7538

The accuracy and the F1 score are very close together. These are both very good values for the Model from Scratch.

4.2.2 MODEL WITH TRANSFER LEARNING

With this model the results are not better than expected. I will list the results here:

- Test Accuracy : 86.42%

- Test F1 Score : 0.8621

4.2.3 Benchmark Comparison

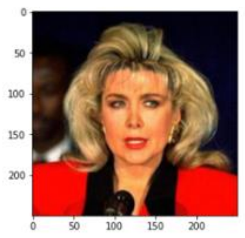
The model created using transfer learning have an accuracy 86.42% compared to the CNN model created from scratch had an accuracy of 75.59%. Both result are well.

5 CONCLUSION

5.1 FREE-FORM VISUALIZATION

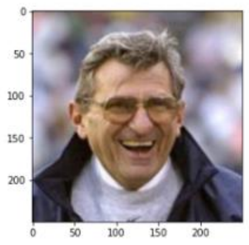
On this occasion I would like to show some outputs of the developed function predict_breed_transfer:

```
Hello dog! Here are the predictions for your dog breed
Top  1 prediction: 100.00% - Curly-coated retriever
Top  2 prediction:   0.00% - American water spaniel
Top  3 prediction:   0.00% - Chesapeake bay retriever
```



```
Hello human! If you were a dog, you could look like
Top  1 prediction:  95.30% - Dachshund
Top  2 prediction:   3.46% - American staffordshire terrier
Top  3 prediction:   1.24% - Dandie dinmont terrier
```

```
Hello dog! Here are the predictions for your dog breed
Top  1 prediction:  99.92% - Curly-coated retriever
Top  2 prediction:   0.07% - Chesapeake bay retriever
Top  3 prediction:   0.01% - American water spaniel
```



```
Hello human! If you were a dog, you could look like
Top  1 prediction:  66.97% - Australian shepherd
Top  2 prediction:  23.05% - Canaan dog
Top  3 prediction:   9.97% - Akita
```

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 image 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. Every function and every part of the project has been clearly documented
9. The GitHub repository has been clearly documented and provided with README files.

5.4 IMPROVEMENT

The model can be improved by adding more training and test data, currently the model is created using only 133 breeds of dog. Also, by performing more image augmentation, we can avoid overfitting and improve the accuracy. May be the model can be improved using different architecture.

REFERENCES

1. Original repo for Project : https://github.com/udacity/deep-learning-v2-pytorch/blob/master/project-dog-classification/
2. ResNext101 : https://pytorch.org/hub/pytorch_vision_resnext/
3. F1 Score : https://en.wikipedia.org/wiki/F1_score
4. PyTorch Documentation : https://pytorch.org/docs/master