Capstone Project – CNN Project: Dog Breed Classifier

Udacity – Machine Learning Engineer Nanodegree

Buu Son, Dinh
Stuttgart, Germany
May 27, 2020

Abstract

This project uses computer vision technique to predict dog breeds from images. The involved Convolutional Neural Networks (CNN) model is implemented to bridge the gap of traditional Machine Learning techniques in localization and classification in the real-world. Besides, the project also aims to deploy a REST API along with web or mobile app to process user-uploaded images.

Keywords: CNN, REST API

1. Definition

1.1 Project overview

Dog Breed Classifier [7] is one of elective final project in Udacity Machine Learning Engineer Course. Some of breeds share the same body features and structure, so identifying breed in an image is a challenging task. Even with our human eye, guessing correct dog breed is also impossible. For example, Labrador retrievers can be yellow, black or brown, which can confuse our intuition. Besides, noise such as background color can result in a misleading conclusion.

The dataset is taken from sklearn, containing 8351 dog images in 133 categories. This project takes an image from the user and provides the dog species prediction. As part of it, if human is involved in the picture, it will provide the resembling dog breed. By using Pytorch and OpenCV, provided models like Inception-V3, Resnet-50 and VGG-16 would become more powerful when the results are displayed beside the prediction and its augmented pictures. This project can also be applied in Cancer Tumor Processing or Motorcycle Classification.

1.2 Problem Statement

The purpose of the project is to create a model pipeline to predict the species of the dog in the user-provided image. The problem is defined as a multi-class classification task, which is based on user-input images to predict dog breed. First, I started with a CNN model from scratch with 50 epochs and the outcome was 22% accuracy, not so poor, but not so good either; it is however than guessing the breed of the dog with less than 1% chance.

After achieving basic foundation of CNN, a transfer learning technique with pre-trained model, Resnet-50, was a much more accuracy approach, around 4 times better than my scratch CNN model, 83% accuracy. In this technique, by reusing pre-trained knowledge and optimizing the classification part, the task become much easier with less computational resources and training time.

Finally, the prediction is made, dog breed or which dog the human most resembles depends on human or dog is involved.

1.3 Metrics

The model is evaluated on the provided shuffled test dataset over accuracy score and human-eye observation. In this way, the performance of this model can be guaranteed and improved, for example, data cleansing before training, add or remove layer to prevent overfitting. Although there is a bunch of evaluation methods like precision, recall, F1 or confusion matrix, they are useful for unbalanced dataset. Since our refined datasets (both human and dog) are quite balanced, simple accuracy score will be sufficient.

2. Analysis

2.1 Data Exploration and Visualization

There are two data files associated with this project:

- /dogImages: Dog images, consists of 8351 images in total, where train, test, and validation sets include 6680, 836 and 835 images respectively.
- /lfw: human images, consists of 13233 images.

Figure 1, 2, 3 describe the number of images by breed in the training, testing and validation set. As can be seen clearly, the classes are relatively balanced (around 50 per breed in the training set)

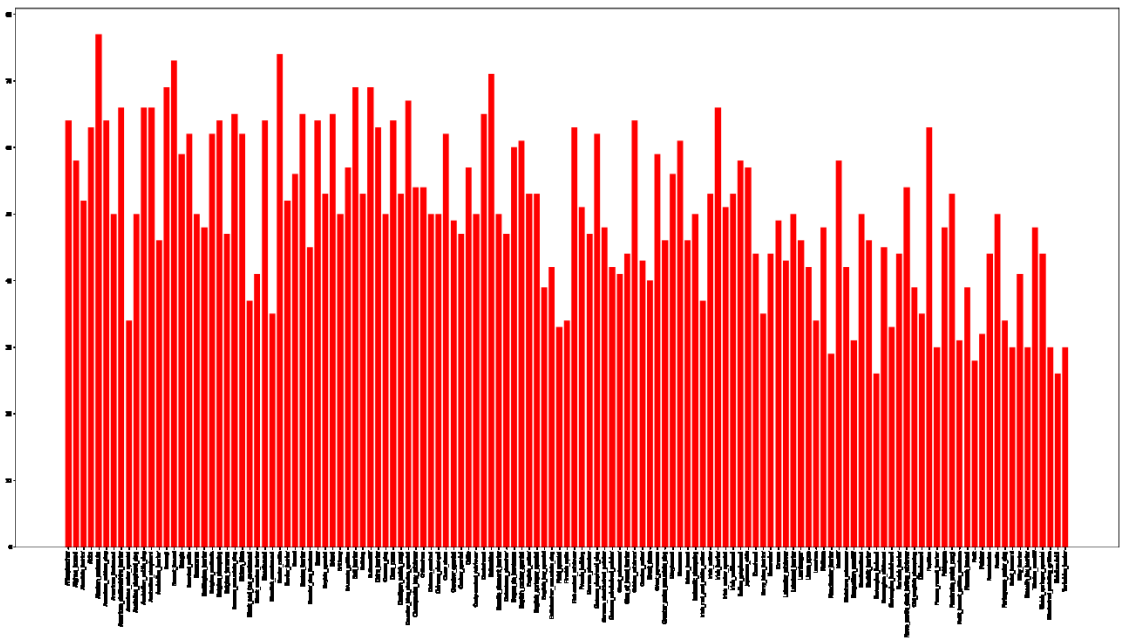

Figure 1: Breed distribution in the training set

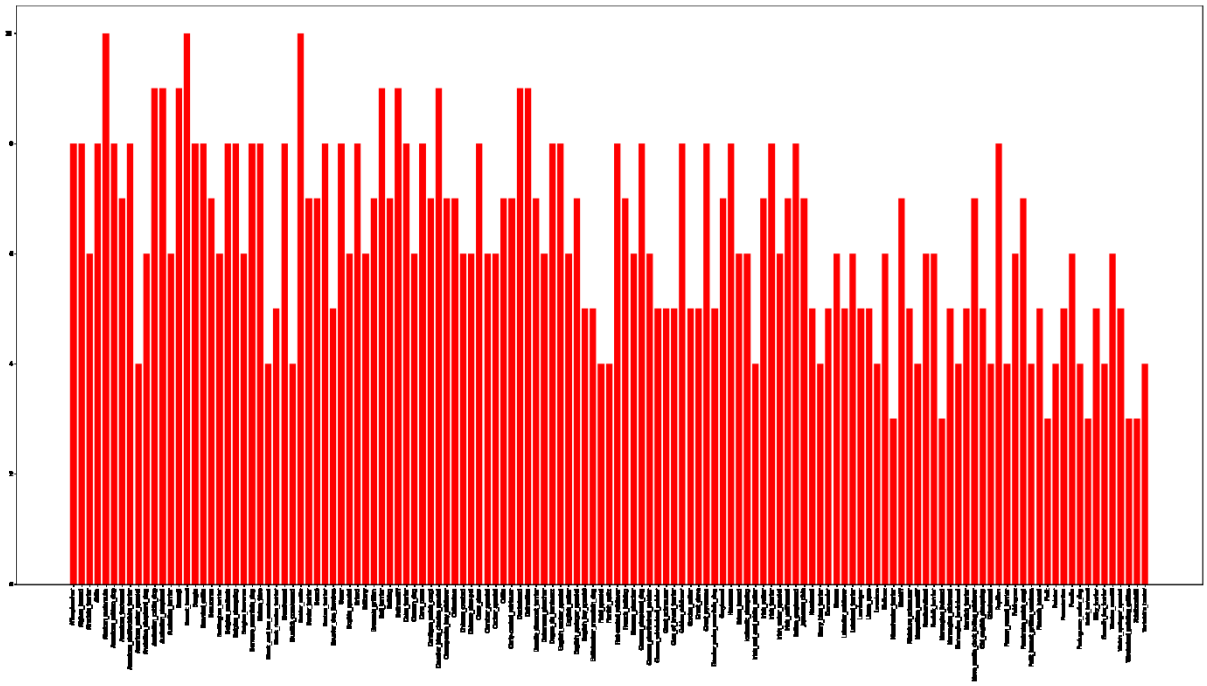

Figure 2: Breed distribution in the testing set

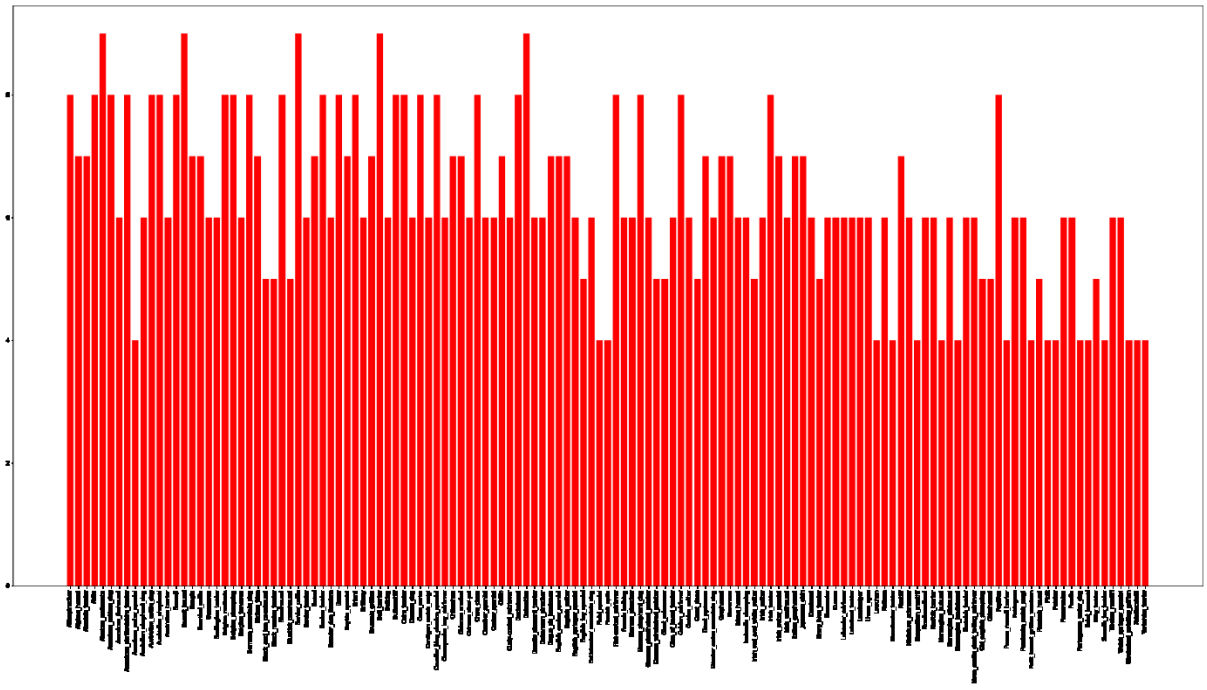

Figure 3: Breed distribution in the validation set

Multiple variations among the training set can be demonstrated in three figures below.

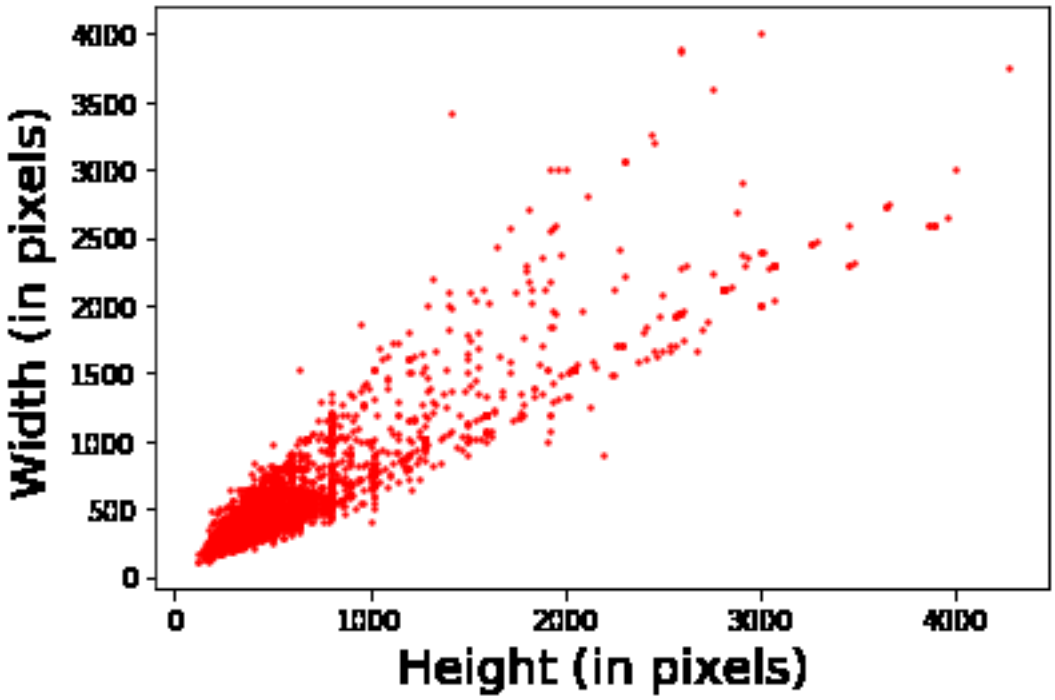

Figure 4: Height vs. Width (in pixels)

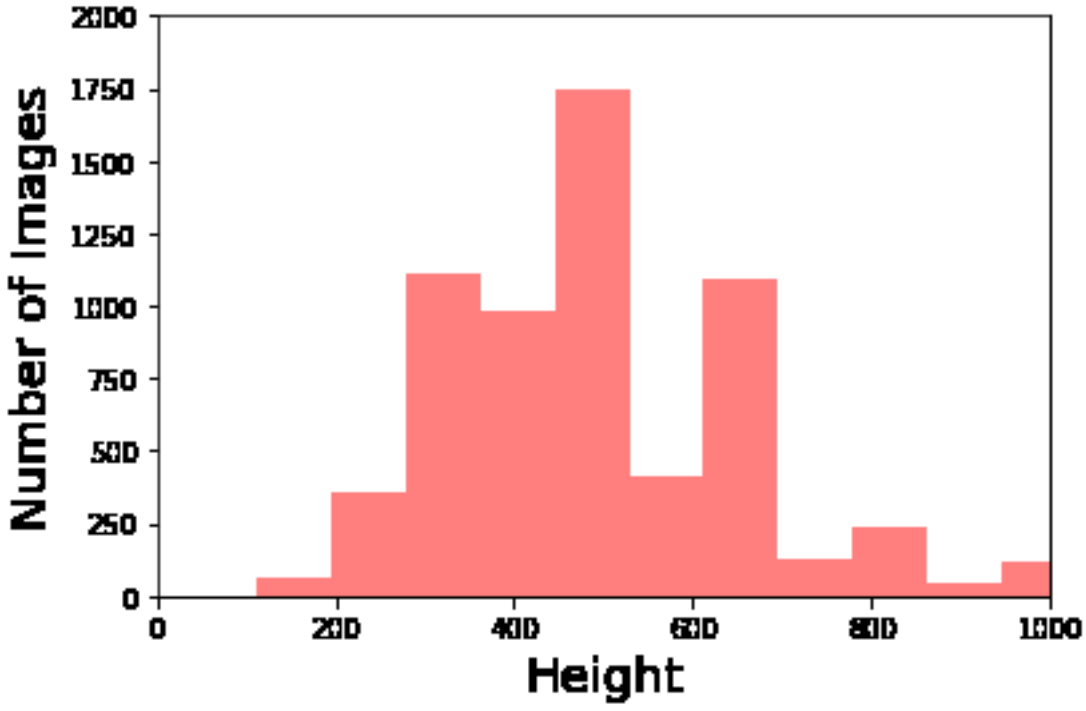

Figure 5: Distribution of image height among the training set

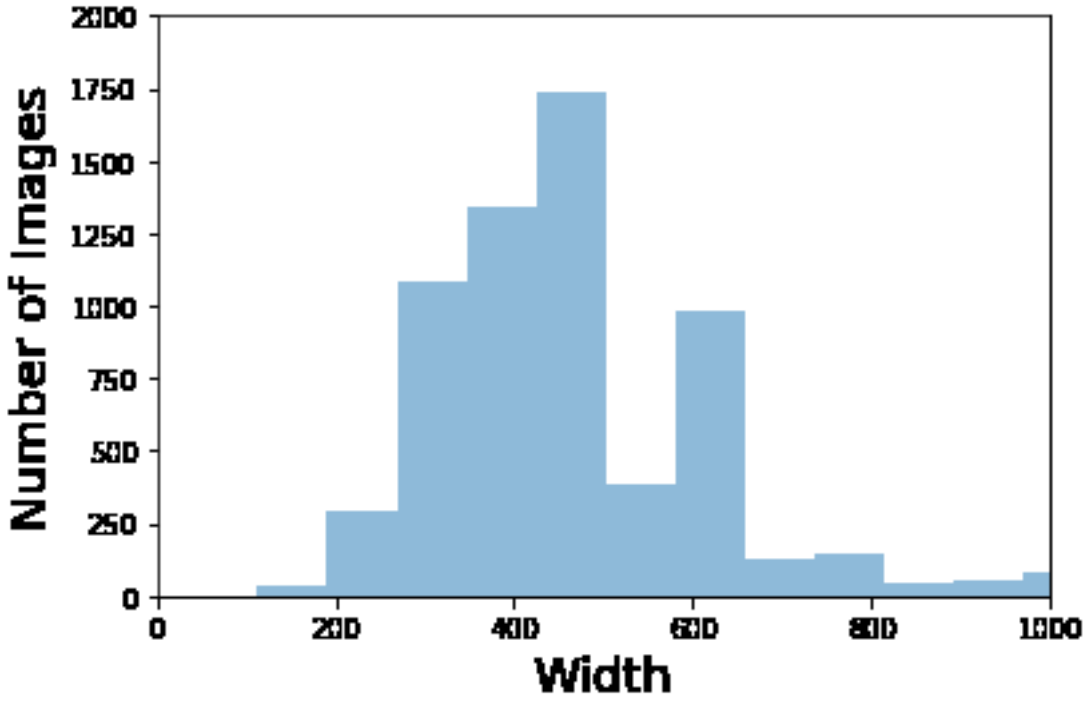

Figure 6: Distribution of image width among the training set

The classifier will take one out of 133 categories of dog. Besides, there are some facing problems:

- Image has more than one dog: this can be solved by implementing another model to detect each dog independently, and crop a bounding box around the dog to feed in our prediction model to find which breed is.

Figure 7: Four Alaskan Dogs in an image

- Noise in the image: cropping can also help

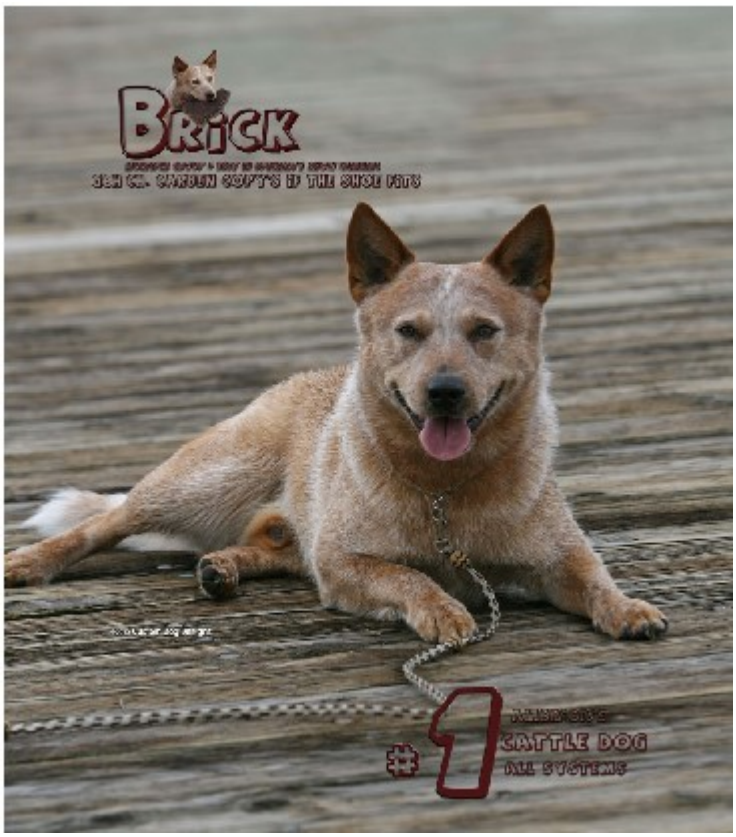

Figure 8: There are a couple of signatures around a dog

- Different species have minor differences in features, which might not be distinguished efficiently right away. In this example, who could figure correctly which one is a Border Collie, Aussie or English Shepherd.


Figure 9: Minor feature differences among different species [5]

2.2 Algorithms and Techniques

This project contains two main parts:

- Human or Dog Detector: The human face detector is built on OpenCV''s Haar Feature-based Cascade Classifier, which was trained on 13233 images. Besides, the dog detector was constructed on a pre-trained CNN model, VGG-16 was recommended in the built-in project, I however implemented Inception-v3 and Resnet-50 to compare their performance. The output of dog detector is an index number which in in range of 151-268 if a dog is found. The purpose of this part is to check if an image is about a dog or a human to further be fed into next part.
- Dog Breed Predictor: CNN is famous for fine-grained classification especially in image classification [1]. Before training the model which will be used later to predict the dog breed or the resembling dog breed depending on the output of Human/Dog Detector; we decided to cleanse the image by cropping, shuffling, and flipping. Once the pre-processing data is complete, it then is put in to training and used to predict on test dataset.

2.3 Benchmark

CNN model is one of the most popular methods to classify images, surpassing traditional machine learning model. In this context, we will restrict the epochs of CNN model to 50 with the help of transfer learning technique. The result then will be evaluated with accuracy score and human-eye observation.

3. Methodology

3.1 Data Preprocessing

Pytorch CNN models expect input images normalized in the same way, a RGB image needs to possess the shape of 3xHxW, where H and W is Height and Weigh respectively, which is required to be at least 224 [6]. The input images have to be loaded in to a range of [0, 1] and then normalized using mean = [0.485, 0.456. 0.406] and standard deviation = [0.229, 0.224, 0.225]. Besides, to make sure the data is randomly distributed and orientated, all the train dataset is shuffled and resized to 224 and cropped from the center, and then rotate at 2020 degrees (which is 220 degrees). The following code is demonstrated as follows:

```
import os
from torchvision import datasets
from PIL import ImageFile
from glob import glob

ImageFile.LOAD_TRUNCATED_IMAGES = True

### TODO: Write data loaders for training, validation, and test sets
## Specify appropriate transforms, and batch_sizes

min_resize = 224
min_crop = 224
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])
```

```
input_dir = 'dogImages'
train_dir = 'train'
test_dir = 'test'
valid_dir = 'valid'

#transform
train_tf = transforms.Compose([transforms.Resize(min_resize),

transforms.CenterCrop(min_crop),

transforms.RandomHorizontalFlip(),

transforms.RandomVerticalFlip(),

transforms.RandomRotation(2020),
                                    transforms.ToTensor(),
                                    normalize])

test_tf = transforms.Compose([transforms.Resize(min_resize),

transforms.CenterCrop(min_crop),
                                    transforms.ToTensor(),
                                    normalize])

valid_tf                =                test_tf                =
transforms.Compose([transforms.Resize(min_resize),

transforms.CenterCrop(min_crop),
                                    transforms.ToTensor(),
                                    normalize])

#load datasets
train_dt  =  datasets.ImageFolder(input_dir  +  '/'  +  train_dir,
transform=train_tf)
test_dt  =  datasets.ImageFolder(input_dir  +  '/'  +  test_dir,
transform=test_tf)
valid_dt  =  datasets.ImageFolder(input_dir  +  '/'  +  valid_dir,
transform=valid_tf)

print("Train data: {} images loaded".format(len(train_dt)))
print("Test data: {} images loaded".format(len(test_dt)))
print("Valid data: {} images loaded".format(len(valid_dt)))

loaders_scratch = {}
loaders_scratch[train_dir]  =  torch.utils.data.DataLoader(train_dt,
batch_size=32, shuffle=True)
loaders_scratch[test_dir]  =  torch.utils.data.DataLoader(test_dt,
batch_size=32)
loaders_scratch[valid_dir]  =  torch.utils.data.DataLoader(valid_dt,
batch_size=32)

print("Total categories: {}".format(len(train_dt.classes)))
print(train_dt.classes)
```

Figure 10: Data Preprocessing Stage

3.2 Implementation

- Human or Dog Detector: As discussed in Algorithms and Techniques section, the snipped code for Human Face Detector and Dog Detector is as follows:

```
# returns "True" if face is detected in image stored at img_path
def face_detector(img_path):
    img = cv2.imread(img_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray)
return len(faces) > 0
```

Figure 11: Human detector

```
from PIL import Image
import torchvision.transforms as transforms

# Set PIL to be tolerant of image files that are truncated.
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True

def VGG16_predict(img_path):
    '''
    Use pre-trained VGG-16 model to obtain index corresponding to
    predicted ImageNet class for image at specified path

    Args:
        img_path: path to an image

    Returns:
        Index corresponding to VGG-16 model's prediction
    '''
    # mini-batches of 3-channel RGB images of shape (3 x H x W),
    # where H and W are expected to be at least 224. The images
    # have to be loaded in to a range of [0, 1] and then normalized
    # using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224,
0.225]

    normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                     std=[0.229, 0.224, 0.225])
    tf = transforms.Compose([transforms.Resize(224),
transforms.CenterCrop(224), transforms.ToTensor(), normalize])

    img = tf(Image.open(img_path)).unsqueeze(0)
    if use_cuda:
        img = img.cuda()

    predictor = VGG16(img)

    return predictor.argmax() # predicted class index

def dog_detector(img_path):
    ## TODO: Complete the function.

    rs = VGG16_predict(img_path).cpu().numpy() #convert cuda instance
to cpu -> numpy

    return (rs >= 151 and rs <= 268)
```

Figure 12: Dog detector based on VGG-16

```
#define model
inception = models.inception_v3(pretrained=True)

#check cuda
if use_cuda:
    inception = inception.cuda()
    inception.eval()

def inception_predict(img_path):

    # mini-batches of 3-channel RGB images of shape (3 x H x W),
    # where H and W are expected to be at least 299. The images
    # have to be loaded in to a range of [0, 1] and then normalized
    # using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224,
0.225]

    normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                     std=[0.229, 0.224, 0.225])
    tf = transforms.Compose([transforms.Resize(299),
transforms.CenterCrop(299), transforms.ToTensor(), normalize])

    img = tf(Image.open(img_path).convert('RGB')).unsqueeze(0)

    if use_cuda:
        img = img.cuda()
    with torch.no_grad():
        predictor = inception(img)


    return predictor.argmax() # predicted class index

def dog_detector_inception(img_path):
    ## TODO: Complete the function.

    rs = inception_predict(img_path).cpu().numpy() #convert cuda
instance to cpu -> numpy

    return (rs >= 151 and rs <= 268)
```

Figure 13: Dog detector based on Inception-v3

```
resnet = models.wide_resnet50_2(pretrained=True)

#check cuda
if use_cuda:
    resnet = resnet.cuda()
    resnet.eval()

def resnet_predict(img_path):
    # mini-batches of 3-channel RGB images of shape (3 x H x W),
    # where H and W are expected to be at least 224. The images
    # have to be loaded in to a range of [0, 1] and then normalized
    # using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224,
0.225]

    normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                     std=[0.229, 0.224, 0.225])
    tf = transforms.Compose([transforms.Resize(224),
transforms.CenterCrop(224), transforms.ToTensor(), normalize])
```

```
    img = tf(Image.open(img_path).convert('RGB')).unsqueeze(0)

    if use_cuda:
        img = img.cuda()
    with torch.no_grad():
        predictor = resnet(img)


    return predictor.argmax() # predicted class index

### returns "True" if a dog is detected in the image stored at img_path
def dog_detector_resnet(img_path):
    ## TODO: Complete the function.

    rs = resnet_predict(img_path).cpu().numpy() #convert cuda instance
to cpu -> numpy

    return (rs >= 151 and rs <= 268)
```

Figure 14: Dog detector based on ResNet-50

- Dog Breed Predictor: For the basic CNN, I chose the following architecture:

```
Net(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False
)
  (fc1): Linear(in_features=25088, out_features=1000, bias=True)
  (fc2): Linear(in_features=1000, out_features=133, bias=True)
  (dropout): Dropout(p=0.4, inplace=False)
  (batch_norm): BatchNorm1d(1000, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
)
```

Figure 15: Scratch CNN architecture

- The CNN architecture is basically a feature extractor [8] [9]
- The feature extractor consists of 4 CNN layers, where each has a ReLU and a MaxPooling2D to convert all negative pixels values to 0 and reduce the parameter of layers by taking the maximum value
- The details for input size at each layer is as follows:
  - First layer: 224x224x3
  - Second layer: 112x112x16
  - Third layer: 56x56x32
  - Forth layer: 28x28x64, output 14x14x128
- After the CNN layers, 2 Dropout layer with probality of 0.4 are introduced beside with linear function to convert 14x14x128 -> 1000 -> 133
- For the criterion, cross-entropy loss was chosen to measure the performance of the classification model; it increases as the predicted label probability diverges from the true label. Our task was therefore, keep it low as possible. Additionally, by defining the learning as 0.01 and Stochastic Gradient Descent (SGD) as optimizer, the global optimal might be reached with enough epochs. Certainly, another optimizer, Adaptive Moment Estimation (Adam), is expected as good as SGD, by defining a good enough AdaBound, Adam can converge faster, it is however complicated to find this optimal AdaBound [3]. Furthermore, Adam has lower training error but not validation error as compared to SGD [2].

3. 3 Refinement

The scratch CNN was not so perfect, about 22% accuracy; still have more space to improve. Then, the pre-trained Resnet-50 was implemented, the last layer (Linear layer) was modified to match with the output size of 133, and all the pameters of feature extractor are deactivated, this is to make sure the parameters of the classifier get backprograted.

4. Results

4.1 Model Evaluation and Validation

In 100 images of dog and human (100 each), human detector achieved 96% for human data but there are 18% of dog was false detected. Dog detector performance based on VGG-16, Inception-v3 and ResNet-50 was summarized in the table below:

	Model		
	VGG-16	Inception-v3	ResNet-50
Human detected as dog	0%	0%	2%
Dog detected	95%	96%	95%

Figure 16: Two images got misclassified by the three pre-trained models.

As shown in figure, these two dogs got false prediction by all three models.

As for dog breed classification, the scratch model accuracy increased from 13% (20 epochs) to 22% (50 epochs) on test set. The transfer model on the other hand, rose from 71% (10 epochs) to 83%. This is to confirm that the more the model is trained, the better it behaves. Moreover, the precision, recall, f1-scorce and confusion matrix of 2 models – scratch CNN and transfer learning models are shown below.

	precision	recall	f1-score	support
Affenpinscher	0.50	0.25	0.33	8
Afghan hound	0.00	0.00	0.00	8
Airedale terrier	0.33	0.17	0.22	6
Akita	0.00	0.00	0.00	8
Alaskan malamute	0.50	0.20	0.29	10
American eskimo dog	0.22	0.25	0.24	8
American foxhound	0.00	0.00	0.00	7
American staffordshire terrier	0.33	0.12	0.18	8
American water spaniel	0.00	0.00	0.00	4
Anatolian shepherd dog	0.00	0.00	0.00	6
Australian cattle dog	0.00	0.00	0.00	9
Australian shepherd	0.50	0.33	0.40	9
Australian terrier	0.00	0.00	0.00	6
Basenji	0.17	0.11	0.13	9
Basset hound	0.29	0.50	0.37	10
Beagle	0.25	0.12	0.17	8
Bearded collie	0.33	0.25	0.29	8
Beauceron	0.40	0.29	0.33	7
Bedlington terrier	0.23	0.50	0.32	6
Belgian malinois	0.33	0.25	0.29	8
Belgian sheepdog	0.00	0.00	0.00	8
Belgian tervuren	0.00	0.00	0.00	6
Bernese mountain dog	0.38	0.38	0.38	8
Bichon frise	0.08	0.12	0.10	8
Black and tan coonhound	0.00	0.00	0.00	4
Black russian terrier	0.50	0.20	0.29	5
Bloodhound	0.30	0.38	0.33	8
Bluetick coonhound	0.00	0.00	0.00	4
Border collie	0.73	0.80	0.76	10
Border terrier	0.29	0.29	0.29	7
Borzoi	0.40	0.29	0.33	7
Boston terrier	0.28	0.62	0.38	8
Bouvier des flandres	0.27	0.60	0.37	5
Boxer	0.25	0.12	0.17	8
Boykin spaniel	0.13	0.33	0.19	6
Briard	0.33	0.25	0.29	8
Brittany	0.20	0.17	0.18	6
Brussels griffon	0.22	0.29	0.25	7
Bull terrier	1.00	0.11	0.20	9
Bulldog	0.40	0.57	0.47	7
Bullmastiff	0.40	0.22	0.29	9
Cairn terrier	0.11	0.12	0.12	8
Canaan dog	0.00	0.00	0.00	6
Cane corso	0.00	0.00	0.00	8
Cardigan welsh corgi	0.00	0.00	0.00	7
Cavalier king charles spaniel	0.22	0.22	0.22	9
Chesapeake bay retriever	0.50	0.43	0.46	7
Chihuahua	0.00	0.00	0.00	7
Chinese crested	0.67	0.33	0.44	6
Chinese shar-pei	0.00	0.00	0.00	6
Chow chow	0.00	0.00	0.00	8
Clumber spaniel	0.17	0.17	0.17	6
Cocker spaniel	0.17	0.17	0.17	6
Collie	0.14	0.14	0.14	7
Curly-coated retriever	0.33	0.14	0.20	7
Dachshund	0.22	0.22	0.22	9

Dalmatian	0.83	0.56	0.67	9
Dandie dinmont terrier	0.21	0.57	0.31	7
Doberman pinscher	0.22	0.33	0.27	6
Dogue de bordeaux	0.43	0.38	0.40	8
English cocker spaniel	0.25	0.38	0.30	8
English setter	0.33	0.17	0.22	6
English springer spaniel	0.20	0.29	0.24	7
English toy spaniel	0.00	0.00	0.00	5
Entlebucher mountain dog	0.00	0.00	0.00	5
Field spaniel	0.00	0.00	0.00	4
Finnish spitz	0.00	0.00	0.00	4
Flat-coated retriever	0.30	0.38	0.33	8
French bulldog	0.00	0.00	0.00	7
German pinscher	0.11	0.17	0.13	6
German shepherd dog	0.40	0.25	0.31	8
German shorthaired pointer	0.14	0.17	0.15	6
German wirehaired pointer	0.25	0.40	0.31	5
Giant schnauzer	0.00	0.00	0.00	5
Glen of imaal terrier	0.25	0.40	0.31	5
Golden retriever	0.11	0.12	0.12	8
Gordon setter	0.20	0.20	0.20	5
Great dane	0.50	0.20	0.29	5
Great pyrenees	0.11	0.12	0.12	8
Greater swiss mountain dog	0.12	0.20	0.15	5
Greyhound	0.25	0.14	0.18	7
Havanese	0.18	0.25	0.21	8
Ibizan hound	0.31	0.83	0.45	6
Icelandic sheepdog	0.00	0.00	0.00	6
Irish red and white setter	0.18	0.50	0.27	4
Irish setter	0.25	0.29	0.27	7
Irish terrier	0.21	0.50	0.30	8
Irish water spaniel	0.40	0.33	0.36	6
Irish wolfhound	0.12	0.14	0.13	7
Italian greyhound	0.33	0.38	0.35	8
Japanese chin	0.43	0.43	0.43	7
Keeshond	0.50	0.20	0.29	5
Kerry blue terrier	0.00	0.00	0.00	4
Komondor	0.57	0.80	0.67	5
Kuvasz	0.11	0.17	0.13	6
Labrador retriever	1.00	0.20	0.33	5
Lakeland terrier	1.00	0.17	0.29	6
Leonberger	0.22	0.40	0.29	5
Lhasa apso	0.00	0.00	0.00	5
Lowchen	0.00	0.00	0.00	4
Maltese	0.25	0.33	0.29	6
Manchester terrier	0.00	0.00	0.00	3
Mastiff	0.00	0.00	0.00	7
Miniature schnauzer	0.00	0.00	0.00	5
Neapolitan mastiff	0.00	0.00	0.00	4
Newfoundland	0.00	0.00	0.00	6
Norfolk terrier	0.12	0.17	0.14	6
Norwegian buhund	0.33	0.33	0.33	3
Norwegian elkhound	0.22	0.40	0.29	5
Norwegian lundehund	0.20	0.25	0.22	4
Norwich terrier	1.00	0.20	0.33	5
Nova scotia duck tolling retriever	0.20	0.29	0.24	7
Old english sheepdog	0.00	0.00	0.00	5
Otterhound	0.00	0.00	0.00	4

Papillon	0.22	0.25	0.24	8
Parson russell terrier	0.50	0.25	0.33	4
Pekingese	0.20	0.17	0.18	6
Pembroke welsh corgi	0.27	0.43	0.33	7
Petit basset griffon vendeen	0.33	0.25	0.29	4
Pharaoh hound	0.40	0.80	0.53	5
Plott	0.00	0.00	0.00	3
Pointer	0.25	0.25	0.25	4
Pomeranian	0.14	0.20	0.17	5
Poodle	0.00	0.00	0.00	6
Portuguese water dog	0.00	0.00	0.00	4
Saint bernard	0.00	0.00	0.00	3
Silky terrier	0.10	0.20	0.13	5
Smooth fox terrier	0.00	0.00	0.00	4
Tibetan mastiff	0.30	0.50	0.37	6
Welsh springer spaniel	0.00	0.00	0.00	5
Wirehaired pointing griffon	0.10	0.33	0.15	3
Xoloitzcuintli	0.22	0.67	0.33	3
Yorkshire terrier	0.50	0.25	0.33	4
accuracy			0.22	836
macro avg	0.23	0.21	0.20	836
weighted avg	0.24	0.22	0.21	836

Figure 17: Accuracy report of the scratch model

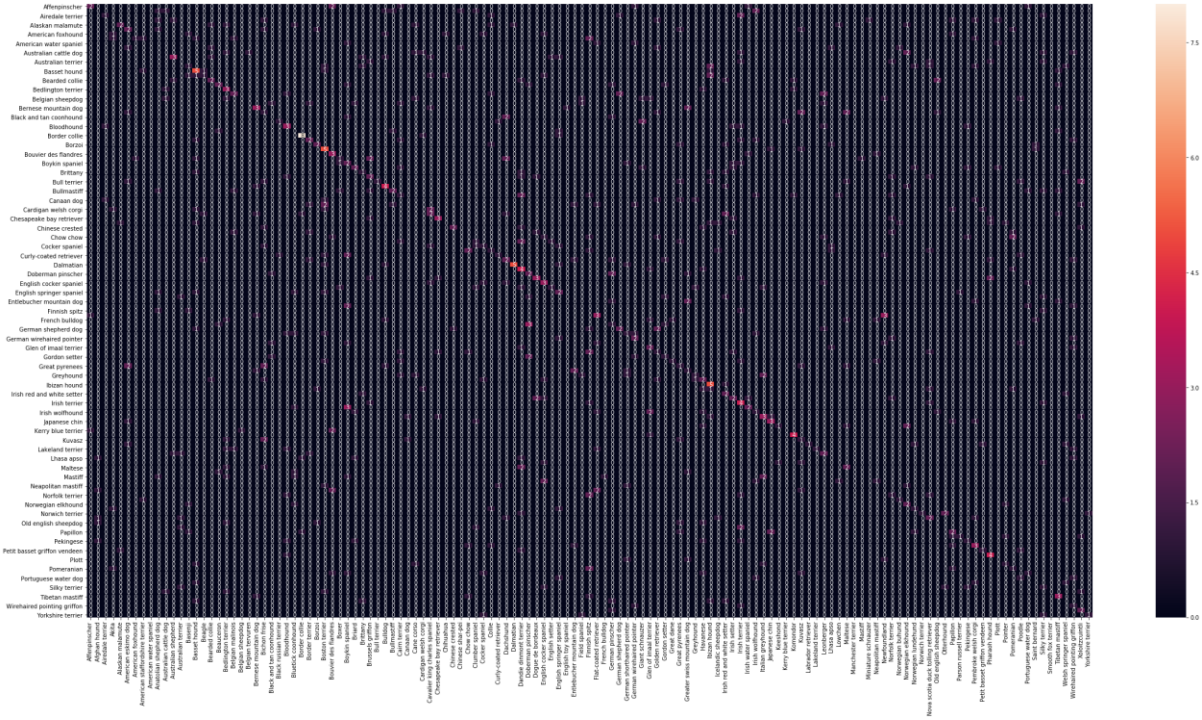

Figure 18: Confusion matrix of the scratch model

	precision	recall	f1-score	support
Affenpinscher	0.86	0.75	0.80	8
Afghan hound	1.00	0.88	0.93	8
Airedale terrier	0.86	1.00	0.92	6
Akita	1.00	0.75	0.86	8
Alaskan malamute	0.83	1.00	0.91	10

American eskimo dog	1.00	0.88	0.93	8
American foxhound	0.70	1.00	0.82	7
American staffordshire terrier	0.80	1.00	0.89	8
American water spaniel	1.00	0.50	0.67	4
Anatolian shepherd dog	0.80	0.67	0.73	6
Australian cattle dog	0.88	0.78	0.82	9
Australian shepherd	0.70	0.78	0.74	9
Australian terrier	1.00	1.00	1.00	6
Basenji	0.73	0.89	0.80	9
Basset hound	0.91	1.00	0.95	10
Beagle	1.00	0.75	0.86	8
Bearded collie	0.62	0.62	0.62	8
Beauceron	1.00	0.86	0.92	7
Bedlington terrier	1.00	1.00	1.00	6
Belgian malinois	0.73	1.00	0.84	8
Belgian sheepdog	0.89	1.00	0.94	8
Belgian tervuren	1.00	0.83	0.91	6
Bernese mountain dog	1.00	1.00	1.00	8
Bichon frise	0.89	1.00	0.94	8
Black and tan coonhound	1.00	0.75	0.86	4
Black russian terrier	0.57	0.80	0.67	5
Bloodhound	1.00	1.00	1.00	8
Bluetick coonhound	1.00	0.75	0.86	4
Border collie	1.00	1.00	1.00	10
Border terrier	1.00	1.00	1.00	7
Borzoi	1.00	0.86	0.92	7
Boston terrier	1.00	0.88	0.93	8
Bouvier des flandres	1.00	0.60	0.75	5
Boxer	0.89	1.00	0.94	8
Boykin spaniel	0.62	0.83	0.71	6
Briard	0.89	1.00	0.94	8
Brittany	0.71	0.83	0.77	6
Brussels griffon	0.88	1.00	0.93	7
Bull terrier	1.00	0.89	0.94	9
Bulldog	1.00	0.71	0.83	7
Bullmastiff	0.88	0.78	0.82	9
Cairn terrier	0.89	1.00	0.94	8
Canaan dog	0.50	0.83	0.62	6
Cane corso	0.78	0.88	0.82	8
Cardigan welsh corgi	0.83	0.71	0.77	7
Cavalier king charles spaniel	0.70	0.78	0.74	9
Chesapeake bay retriever	1.00	1.00	1.00	7
Chihuahua	0.88	1.00	0.93	7
Chinese crested	1.00	0.50	0.67	6
Chinese shar-pei	1.00	0.50	0.67	6
Chow chow	1.00	0.88	0.93	8
Clumber spaniel	1.00	1.00	1.00	6
Cocker spaniel	1.00	0.17	0.29	6
Collie	0.86	0.86	0.86	7
Curly-coated retriever	1.00	1.00	1.00	7
Dachshund	0.80	0.89	0.84	9
Dalmatian	1.00	1.00	1.00	9
Dandie dinmont terrier	1.00	0.86	0.92	7
Doberman pinscher	0.40	1.00	0.57	6
Dogue de bordeaux	0.80	1.00	0.89	8
English cocker spaniel	0.50	0.75	0.60	8
English setter	0.83	0.83	0.83	6
English springer spaniel	0.86	0.86	0.86	7

English toy spaniel	0.50	0.40	0.44	5
Entlebucher mountain dog	0.56	1.00	0.71	5
Field spaniel	1.00	0.25	0.40	4
Finnish spitz	0.00	0.00	0.00	4
Flat-coated retriever	0.89	1.00	0.94	8
French bulldog	0.78	1.00	0.88	7
German pinscher	0.00	0.00	0.00	6
German shepherd dog	1.00	0.88	0.93	8
German shorthaired pointer	0.67	1.00	0.80	6
German wirehaired pointer	0.60	0.60	0.60	5
Giant schnauzer	0.57	0.80	0.67	5
Glen of imaal terrier	1.00	1.00	1.00	5
Golden retriever	0.80	1.00	0.89	8
Gordon setter	0.62	1.00	0.77	5
Great dane	1.00	1.00	1.00	5
Great pyrenees	0.71	0.62	0.67	8
Greater swiss mountain dog	1.00	0.40	0.57	5
Greyhound	0.86	0.86	0.86	7
Havanese	0.71	0.62	0.67	8
Ibizan hound	0.75	1.00	0.86	6
Icelandic sheepdog	0.67	0.33	0.44	6
Irish red and white setter	0.60	0.75	0.67	4
Irish setter	1.00	0.86	0.92	7
Irish terrier	1.00	0.88	0.93	8
Irish water spaniel	0.75	1.00	0.86	6
Irish wolfhound	1.00	0.86	0.92	7
Italian greyhound	0.75	0.75	0.75	8
Japanese chin	0.88	1.00	0.93	7
Keeshond	1.00	1.00	1.00	5
Kerry blue terrier	1.00	0.75	0.86	4
Komondor	1.00	1.00	1.00	5
Kuvasz	0.62	0.83	0.71	6
Labrador retriever	1.00	0.80	0.89	5
Lakeland terrier	1.00	0.83	0.91	6
Leonberger	1.00	1.00	1.00	5
Lhasa apso	0.80	0.80	0.80	5
Lowchen	0.33	0.25	0.29	4
Maltese	0.86	1.00	0.92	6
Manchester terrier	1.00	0.67	0.80	3
Mastiff	0.67	0.86	0.75	7
Miniature schnauzer	0.71	1.00	0.83	5
Neapolitan mastiff	1.00	1.00	1.00	4
Newfoundland	0.86	1.00	0.92	6
Norfolk terrier	1.00	1.00	1.00	6
Norwegian buhund	0.00	0.00	0.00	3
Norwegian elkhound	0.83	1.00	0.91	5
Norwegian lundehund	0.75	0.75	0.75	4
Norwich terrier	1.00	1.00	1.00	
Nova scotia duck tolling retriever	0.88	1.00	0.93	7
Old english sheepdog	0.80	0.80	0.80	5
Otterhound	1.00	0.75	0.86	4
Papillon	0.89	1.00	0.94	8
Parson russell terrier	1.00	0.75	0.86	4
Pekingese	1.00	1.00	1.00	6
Pembroke welsh corgi	0.64	1.00	0.78	7
Petit basset griffon vendeen	1.00	0.50	0.67	4
Pharaoh hound	1.00	0.40	0.57	5
Plott	1.00	0.67	0.80	3

Pointer	1.00	0.25	0.40	4
Pomeranian	0.80	0.80	0.80	5
Poodle	1.00	0.83	0.91	6
Portuguese water dog	1.00	0.25	0.40	4
Saint bernard	1.00	1.00	1.00	3
Silky terrier	0.50	0.80	0.62	5
Smooth fox terrier	1.00	0.75	0.86	4
Tibetan mastiff	1.00	0.83	0.91	6
Welsh springer spaniel	1.00	0.60	0.75	5
Wirehaired pointing griffon	0.33	0.33	0.33	3
Xoloitzcuintli	1.00	1.00	1.00	3
Yorkshire terrier	0.67	0.50	0.57	4
accuracy			0.83	836
macro avg	0.84	0.81	0.80	836
weighted avg	0.85	0.83	0.82	836

Figure 19: Accuracy report of ResNet-50 model.

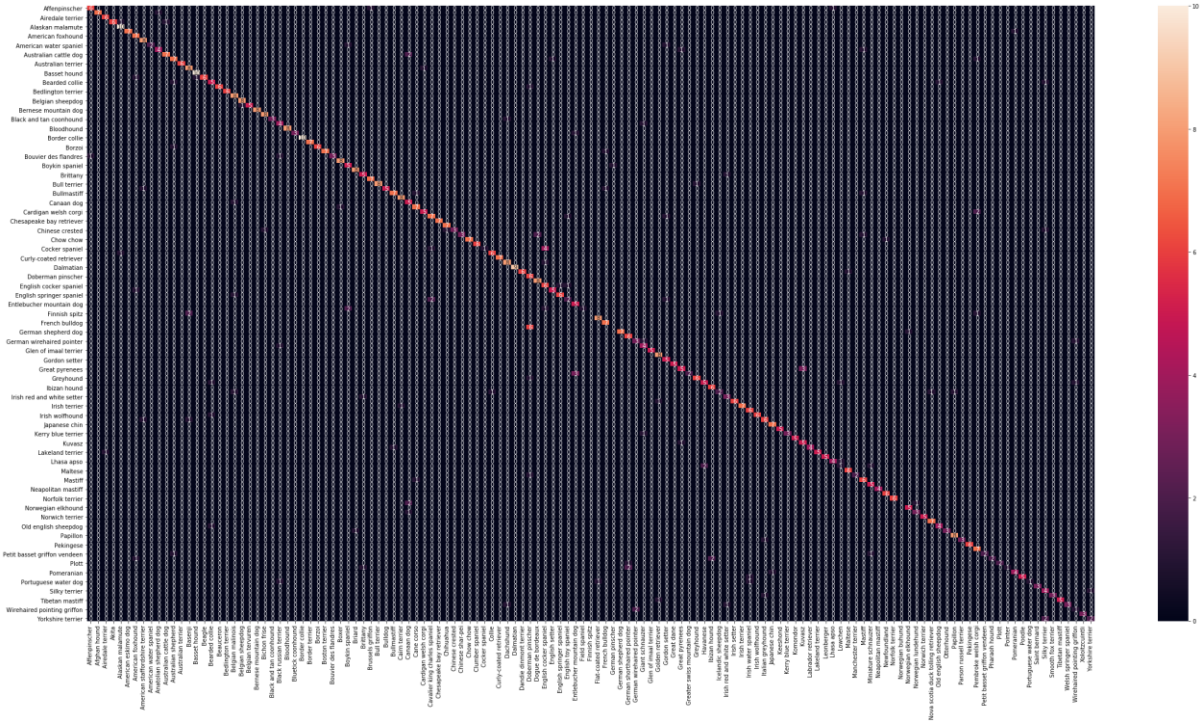

Figure 20: Confusion matrix of ResNet-50.

Below are several samples test taken from this test function in both human and dog images, and the last image was misclassified because the dog detector could not see it as a dog while the human detector did not see the dog as human, the breed classifier therefor was not applied.

Figure 21: Sample output.

4.2 Justification

The overfiting is likely to occur since the model is relatively small, only 133 categories among 8500 images. Preprocessing the data further can somehow figure out some interesting feature could help the model to generalize better. In addition, improving the performance of the human/dog detector helps the classifier work more efficiently. .

4.3 Improvement

- Try other pre-trained models available in the project such as Alexnet, Xception, VGG-19, etc with larger epoch number (thousand)
- Improve the detector performance (it was 96% accuracy now), because the human/dog_detector was built over dog_breed_classifier, as in my sample test case above, if the prior detector was failed to detect human or dog, even if the image is about a dog, the classifier would never work.
- Classify multiple dogs in the same image: implement YOLO-3 [4] to detect each dog and crop each out, then feed them into the classifier.
- Classify dogs and human in the same image: in the same manner, individual dog and human is fed into the classifier.
- As for the problems of dogs overlapping each other, such an extremely challenging task, hopefully will be solved in the future.

5. References:

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

[2] https://shaoanlu.wordpress.com/2017/05/29/sgd-all-which-one-is-the-best-optimizer-dogs-vs-cats-toy-experiment/

[3] https://medium.com/syncedreview/iclr-2019-fast-as-adam-good-as-sgd-new-optimizer-has-both-78e37e8f9a34

[4] https://pjreddie.com/darknet/yolo/

[5] https://www.border-wars.com/2010/12/uncanny-minor-differences.html

[6] https://pytorch.org

[7] https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification

[8] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[9] https://algorithmia.com/blog/convolutional-neural-nets-in-pytorch