report

January 5, 2020

1 CNN Project: Dog Breed Classifier

1.1 Machine Learning Engineer Nanodegree

1.2 Capstone Project

Kenneth Preston January 4th, 2020

1.3 I. Definition

1.3.1 Project Overview

In this section, look to provide a high-level overview of the project in layman's terms. Questions to ask yourself when writing this section: - *Has an overview of the project been provided, such as the problem domain, project origin, and related datasets or input data? - Has enough background information been given so that an uninformed reader would understand the problem domain and following problem statement?*

This project is udacity suggested. It involves training a custom deep learning model to identify dog breeds. It is an opportunity to explore the pytorch package and practice image classification.

Problem Domain The problem domain is image classification. So it is a classification problem, but also requires an approproach appropriate for working with images (convolutional layers). This is simply a computer vision problem. The problem is simply to classify an image of one subject (one dog, or one human) as a dog of a specific breed. The algorithm will view an image of a dog or a person and return a prediction of the type of dog it is (or what dog a human looks like).

Achieving accuracy in this classification task seemed to be a challenge because there would likely be a lot of similarities between the breeds. I assumed that the task would be more difficult than differientiating between a Cat and a Dog. I was interested in this problem mainly because I find computer vision the most facinating topic in artificial intelligence.

Project Origin This project comes from a udacity suggested problem. This specific project is a popular one and is used in other udacity nanodegree programs as well as by the AI community in general. I believe the origin is simply a multiclass classification problem for image classification. #### Datasets This project uses two datasets, but mainly relies on the dog dataset for training purposes. - Dog images: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip - Human images: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip The inputs are images of a single entity, either a dog (in the dog dataset) or a human (in the human dataset). They are not complicated

by multiple entities in one image (such as a mix of dogs and humans). The files provided are in the jpg format and easily machine readable at sensible image sizes (will outline this further below). The image sizes do vary so preprocessing is necessary to train or evaluate the model (i.e. produce predictions)

1.3.2 Problem Statement

In this section, you will want to clearly define the problem that you are trying to solve, including the strategy (outline of tasks) you will use to achieve the desired solution. You should also thoroughly discuss what the intended solution will be for this problem. Questions to ask yourself when writing this section: *- Is the problem statement clearly defined? Will the reader understand what you are expecting to solve? - Have you thoroughly discussed how you will attempt to solve the problem? - Is an anticipated solution clearly defined? Will the reader understand what results you are looking for?*

The Problem The problem I am trying to solve is to predict accurately the breed of a dog from a image. The range of dog breeds as defined by this project is 133 breeds. I would like to predict the one breed the image is most likely to be. A fun application of this derived model is to be able to predict which dog breed a human looks like when providing the model with human faces instead of dogs.

The Strategy The strategy I will employ to tackle this problem is as follows: - In practice I will follow the dog_app.ipynb notebook provided at the github address: https://github.com/udacity/deep-learning-v2-pytorch.git under the project-dog-classification folder. - I will use a training and validation set of images to train a multiclass image classification algorithm and then build a function that ingests an image and provides a dog breed prediction from that image. I will also use a dog and human face detector to determine whether the image is of a human or a dog and change the message shown to the user based on this information. - Although part of the notebook will have me attempting to build a classification model from scratch, the model I will put into the application will be one based on transfer learning using a pretrained model that will have all its layers frozen accept for the final layer used for classification. This way I can use the benefits of using a model trained on many more images than I can afford to do by myself. - In order to augment my understanding of this subject I will use the pytorch documentation heavily and also rely alot on google searches to trouble shoot the issues I come up to when developing my model from scratch and my transfer learning model. Although I have taken courses on this subject I will use online tutorials as inspiration when trying out different model architectures, preprocessing techniques, and finding solutions to issues such as when my model overfits or my accuracy level does not improve.

The Solution To solve this issues I will be standing on the shoulders of giants in computer vision. I will be using the hard work of others that have built accurate image classifiers and apply their solution to my problem through transfer learning. This will provide me with an accurate dog breed classifier that I can use in conjunction with a human face detector and a dog face detector. For the face detectors I can also rely on accurate models that are publicly available. For face detection I can simply used pretrained models already capable at performing this task. The resulting application will accomplish the following tasks: - Ingest an image file of jpg format. - Identify if a human or a dog is present. - Identify which breed of dog it thinks is present. - Return the image

with a text message letting the user know whether it is a dog or human in the image and then which breed it thinks the dog or human looks the most like.

1.3.3 Metrics

In this section, you will need to clearly define the metrics or calculations you will use to measure performance of a model or result in your project. These calculations and metrics should be justified based on the characteristics of the problem and problem domain. Questions to ask yourself when writing this section: *- Are the metrics you've chosen to measure the performance of your models clearly discussed and defined? - Have you provided reasonable justification for the metrics chosen based on the problem and solution?*

The metrics I will be using to measure the performance of the model I will be building to identify dog breeds will include: - During training I will be using a loss function called multi-class cross-entropy loss as applied to the validation set. This will help me understand how well my model is improving as the training is proceeding. This is used as an appropriate loss function for multi-class classification as it is often the default loss function for this type of problem (https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/). A full explaination of this loss function can be found here: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy. In laymen terms this loss function produces a lower loss when the correct answer is more confidently predicted (i.e. probability approaches 1) while also increasing the loss when confidently predicting (i.e. closer to 1) an incorrect classification. I believe this is an obvious choice of loss function for the task of multi-class classification. - When evaluating the model's performance after training I will be relying on a simple test accuracy score as defined in the provided notebook. The math is simple: Total # of Correct Predictions / Total Number of Images Tested . I believe this is an appropriate measure of performance as it clearly reveals how often the model is correct in its prediction. Analyses of false negatives or positives is not necessary because this classification task if trivial. If we were conducting multi-class classification in something like cancer screening we would want to ensure we focus more on metrics that look at false positives and false negatives as these mistakes would cost lives or create worry for users. Incorrectly predicting a dog breed of a dog or human is a trivial task and I believe we can focus solely on just improving accuracy and not weigh or evaluate false negatives or false positives.

1.4 II. Analysis

1.4.1 Data Exploration

In this section, you will be expected to analyze the data you are using for the problem. This data can either be in the form of a dataset (or datasets), input data (or input files), or even an environment. The type of data should be thoroughly described and, if possible, have basic statistics and information presented (such as discussion of input features or defining characteristics about the input or environment). Any abnormalities or interesting qualities about the data that may need to be addressed have been identified (such as features that need to be transformed or the possibility of outliers). Questions to ask yourself when writing this section: *- If a dataset is present for this problem, have you thoroughly discussed certain features about the dataset? Has a data sample been provided to the reader? - If a dataset is present for this problem, are statistics about the dataset calculated and reported? Have any relevant results from this calculation been discussed? - If a dataset is **not** present for this problem, has discussion been made about the input space or input data for your problem? - Are there*

any abnormalities or characteristics about the input space or dataset that need to be addressed? (categorical variables, missing values, outliers, etc.)

The main dataset used for this problem is a dog image dataset that was provided for this project. This project is available here for interested readers: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip. The dataset includes 8351 jpg format dog images organized by dog breed folders for a total of 133 folders. Each folder has between 26 and 77 images of a specific breed of dog. From a limited look through the images we can see that the images range in size on both height and width. This suggests that we will need to transform these images to a standard size to use for training, validation, and testing data. Otherwise the data looks very clean and currated for this project.

```
In [4]: import numpy as np
        from glob import glob

        dog_files = np.array(glob("/data/dog_images/*/*/*"))
        print('There are %d total dog images.' % len(dog_files))
```

```
There are 8351 total dog images.
```

Below are the folders included in the training data. The same folders are found in the validation and test folders as well.

```
In [56]: ls /data/dog_images/train/
```

```
001.Affenpinscher/                  068.Flat-coated_retriever/
002.Afghan_hound/                   069.French_bulldog/
003.Airedale_terrier/               070.German_pinscher/
004.Akita/                          071.German_shepherd_dog/
005.Alaskan_malamute/               072.German_shorthaired_pointer/
006.American_eskimo_dog/            073.German_wirehaired_pointer/
007.American_foxhound/              074.Giant_schnauzer/
008.American_staffordshire_terrier/ 075.Glen_of_imaal_terrier/
009.American_water_spaniel/         076.Golden_retriever/
010.Anatolian_shepherd_dog/         077.Gordon_setter/
011.Australian_cattle_dog/          078.Great_dane/
012.Australian_shepherd/            079.Great_pyrenees/
013.Australian_terrier/             080.Greater_swiss_mountain_dog/
014.Basenji/                        081.Greyhound/
015.Basset_hound/                   082.Havanese/
016.Beagle/                         083.Ibizan_hound/
017.Bearded_collie/                 084.Icelandic_sheepdog/
018.Beauceron/                      085.Irish_red_and_white_setter/
019.Bedlington_terrier/             086.Irish_setter/
020.Belgian_malinois/               087.Irish_terrier/
021.Belgian_sheepdog/               088.Irish_water_spaniel/
022.Belgian_tervuren/               089.Irish_wolfhound/
023.Bernese_mountain_dog/           090.Italian_greyhound/
024.Bichon_frise/                   091.Japanese_chin/
```

```
025.Black_and_tan_coonhound/
026.Black_russian_terrier/
027.Bloodhound/
028.Bluetick_coonhound/
029.Border_collie/
030.Border_terrier/
031.Borzoi/
032.Boston_terrier/
033.Bouvier_des_flandres/
034.Boxer/
035.Boykin_spaniel/
036.Briard/
037.Brittany/
038.Brussels_griffon/
039.Bull_terrier/
040.Bulldog/
041.Bullmastiff/
042.Cairn_terrier/
043.Canaan_dog/
044.Cane_corso/
045.Cardigan_welsh_corgi/
046.Cavalier_king_charles_spaniel/
047.Chesapeake_bay_retriever/
048.Chihuahua/
049.Chinese_crested/
050.Chinese_shar-pei/
051.Chow_chow/
052.Clumber_spaniel/
053.Cocker_spaniel/
054.Collie/
055.Curly-coated_retriever/
056.Dachshund/
057.Dalmatian/
058.Dandie_dinmont_terrier/
059.Doberman_pinscher/
060.Dogue_de_bordeaux/
061.English_cocker_spaniel/
062.English_setter/
063.English_springer_spaniel/
064.English_toy_spaniel/
065.Entlebucher_mountain_dog/
066.Field_spaniel/
067.Finnish_spitz/
092.Keeshond/
093.Kerry_blue_terrier/
094.Komondor/
095.Kuvasz/
096.Labrador_retriever/
097.Lakeland_terrier/
098.Leonberger/
099.Lhasa_apso/
100.Lowchen/
101.Maltese/
102.Manchester_terrier/
103.Mastiff/
104.Miniature_schnauzer/
105.Neapolitan_mastiff/
106.Newfoundland/
107.Norfolk_terrier/
108.Norwegian_buhund/
109.Norwegian_elkhound/
110.Norwegian_lundehund/
111.Norwich_terrier/
112.Nova_scotia_duck_tolling_retriever/
113.Old_english_sheepdog/
114.Otterhound/
115.Papillon/
116.Parson_russell_terrier/
117.Pekingese/
118.Pembroke_welsh_corgi/
119.Petit_basset_griffon_vendeen/
120.Pharaoh_hound/
121.Plott/
122.Pointer/
123.Pomeranian/
124.Poodle/
125.Portuguese_water_dog/
126.Saint_bernard/
127.Silky_terrier/
128.Smooth_fox_terrier/
129.Tibetan_mastiff/
130.Welsh_springer_spaniel/
131.Wirehaired_pointing_griffon/
132.Xoloitzcuintli/
133.Yorkshire_terrier/
```

In this code chunk below I am showing the number of files in each folder range between 26 to 77 images in the training set. There are between 4 and 9 in the validation set. The test set has between 4 and 10.

```
In [54]: folderimnum = []
         folderval = []
         foldertest=[]
         dogtrain = glob("/data/dog_images/train/*")
         dogval = glob("/data/dog_images/valid/*")
         dogtest = glob("/data/dog_images/test/*")
         for i in dogtrain:
             folderimnum.append(len(glob(i+"/*")))
         for i in dogval:
             folderval.append(len(glob(i+"/*")))
         for i in dogtest:
             foldertest.append(len(glob(i+"/*")))
         print("Training: There are between {} and {} images in each folder of a breed of dog".f
         print("Validation: There are between {} and {} images in each folder of a breed of dog"
         print("Test: There are between {} and {} images in each folder of a breed of dog".forma
```

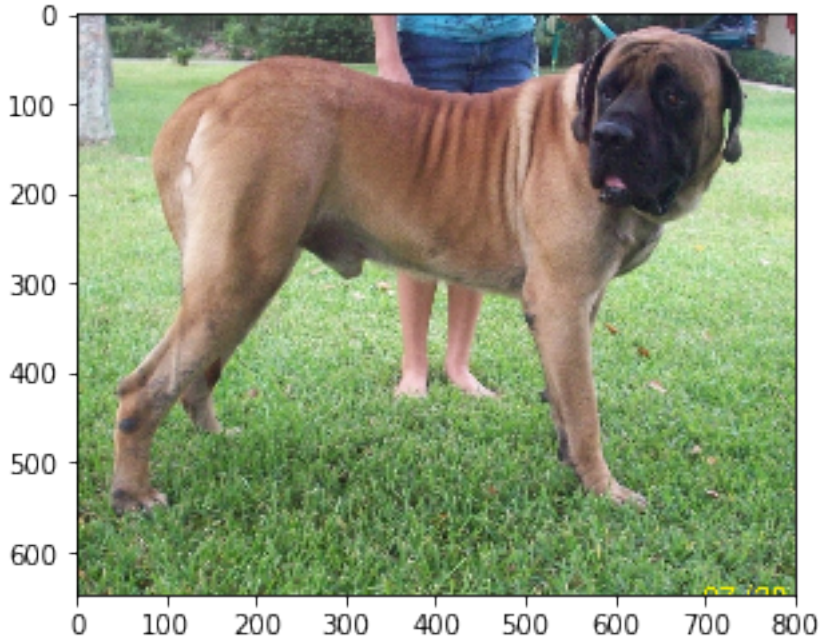
```
Training: There are between 26 and 77 images in each folder of a breed of dog
Validation: There are between 4 and 9 images in each folder of a breed of dog
Test: There are between 3 and 10 images in each folder of a breed of dog
```

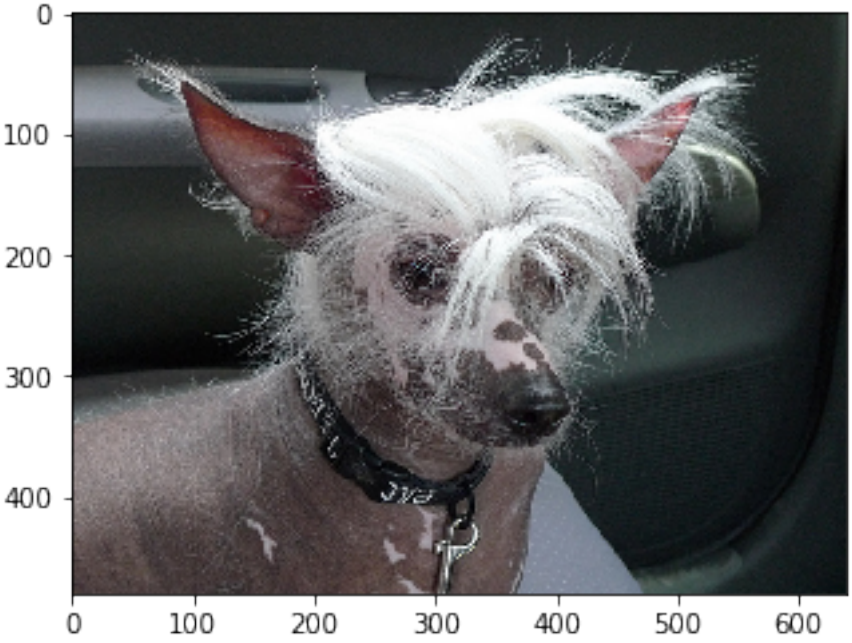
1.4.2 Exploratory Visualization

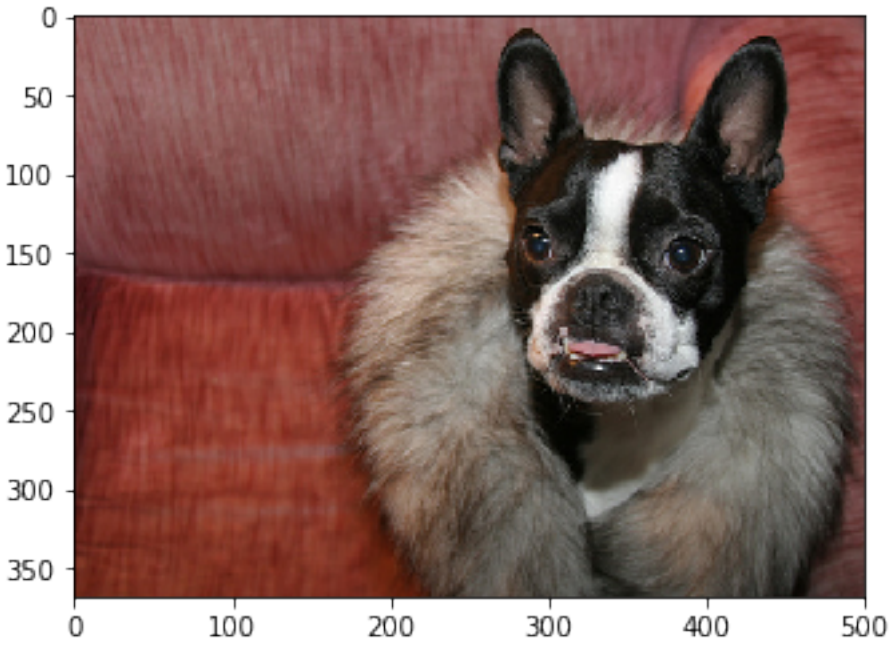
In this section, you will need to provide some form of visualization that summarizes or extracts a relevant characteristic or feature about the data. The visualization should adequately support the data being used. Discuss why this visualization was chosen and how it is relevant. Questions to ask yourself when writing this section: *- Have you visualized a relevant characteristic or feature about the dataset or input data? - Is the visualization thoroughly analyzed and discussed? - If a plot is provided, are the axes, title, and datum clearly defined?*

Below are some example images. You will notice that the images range in exact size. This suggests the images need to be transformed in order to be used for model building.

```
In [59]: import cv2
         import matplotlib.pyplot as plt
         %matplotlib inline
         for i in [glob("/data/dog_images/train/*/*")[0], glob("/data/dog_images/train/*/*")[300
             img = cv2.imread(i)
             img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
             plt.imshow(img)
             plt.show()
```


0
100
200
300
400
500
600
0
100
200
300
400
500
600
700
800

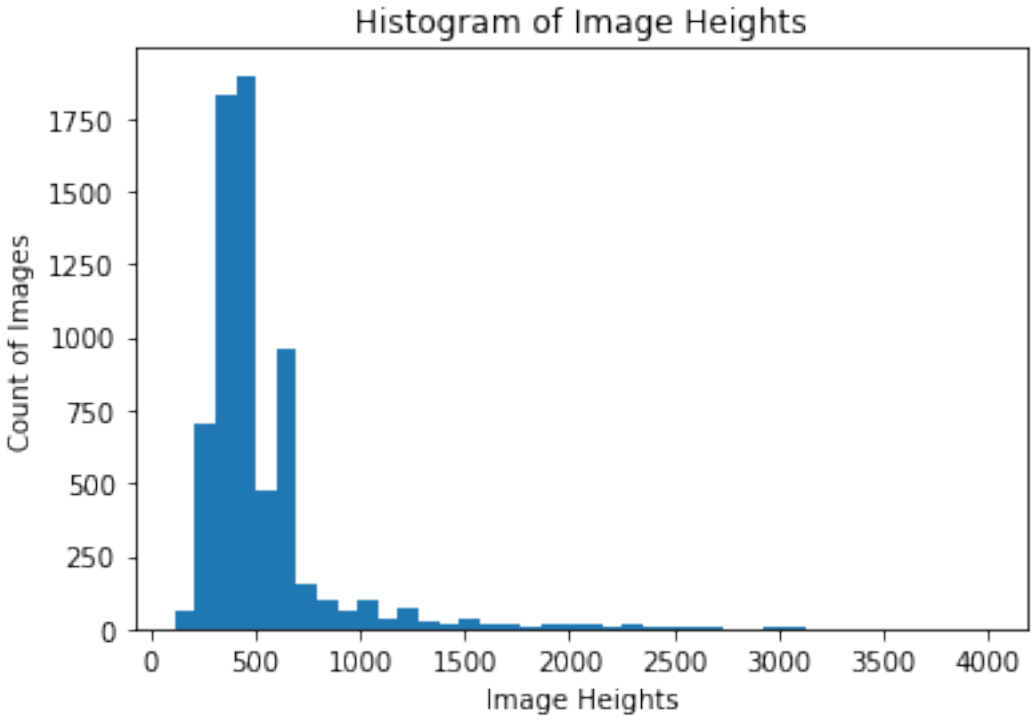

0
100
200
300
400
0
100
200
300
400
500
600

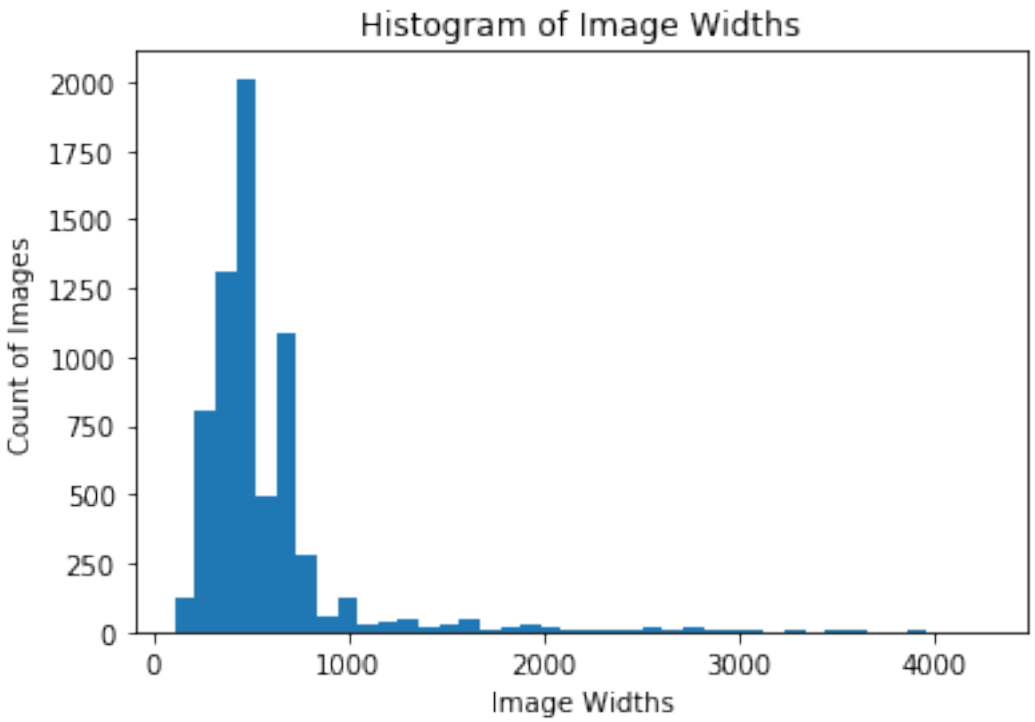


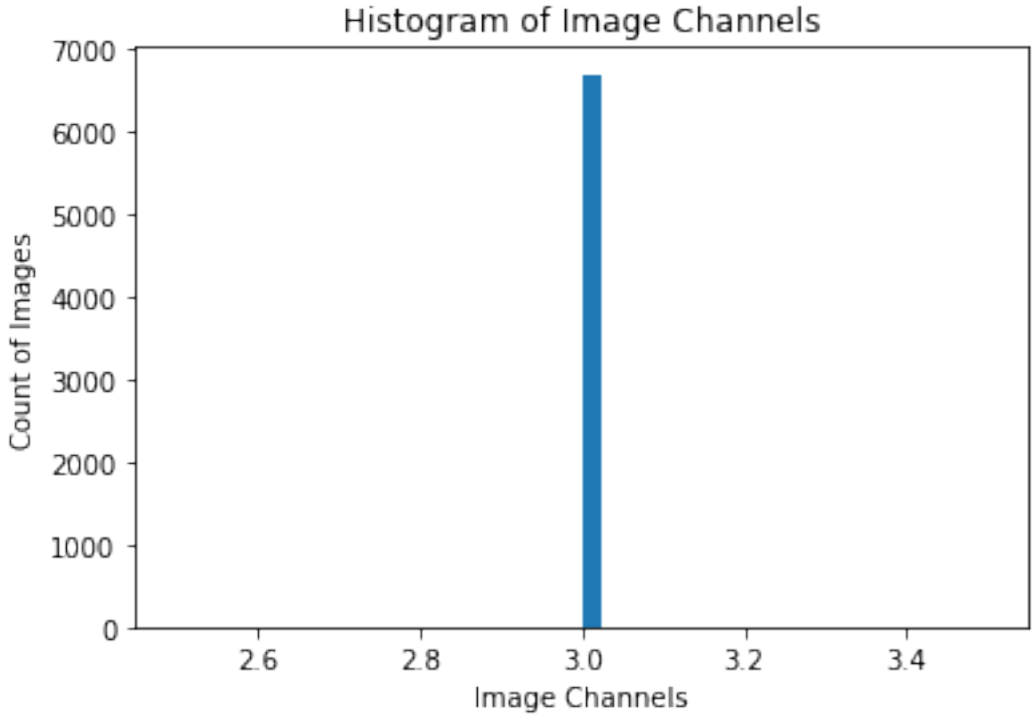
Below is a visualization of the shapes of the images that are included in the training set. We can see that the majority of them are relatively small, while some images are very large. This gives a clear indication that the images need to be resized when preprocessing the data. You can see that all images have 3 color channels, but vary widely on the other dimensions (height and width)

```
In [104]: height =[]
          width =[]
          channel =[]
          for i in glob("/data/dog_images/train/*/*"):
              img = cv2.imread(i)
              height.append(img.imag.shape[0])
              width.append(img.imag.shape[1])
              channel.append(img.imag.shape[2])

In [109]: import numpy as np
          plt.hist(height, bins=40)
          plt.title("Histogram of Image Heights")
          plt.xlabel("Image Heights")
          plt.ylabel("Count of Images")
          plt.show()

          plt.hist(width, bins=40)
          plt.title("Histogram of Image Widths")
          plt.xlabel("Image Widths")
          plt.ylabel("Count of Images")
          plt.show()
```

```
plt.hist(channel, bins=40)
plt.title("Histogram of Image Channels")
plt.xlabel("Image Channels")
plt.ylabel("Count of Images")
plt.show()
```




Histogram of Image Widths
Count of Images
Image Widths
0
250
500
750
1000
1250
1500
1750
2000
0
1000
2000
3000
4000


Histogram of Image Channels
Count of Images
Image Channels
0
1000
2000
3000
4000
5000
6000
7000
2.6
2.8
3.0
3.2
3.4

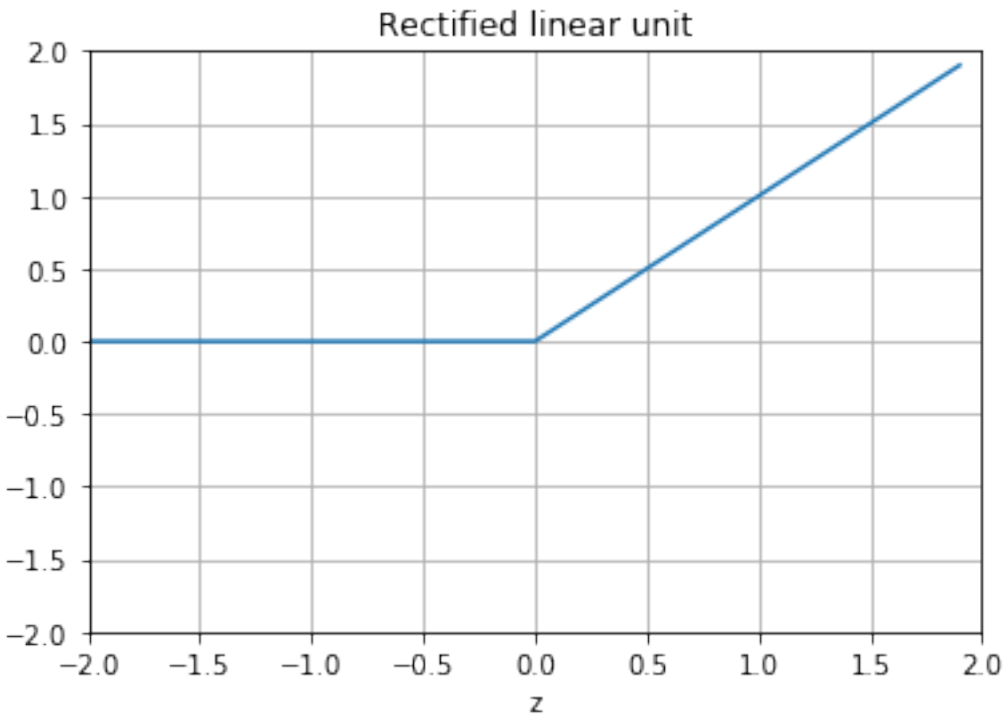
1.4.3 Algorithms and Techniques

In this section, you will need to discuss the algorithms and techniques you intend to use for solving the problem. You should justify the use of each one based on the characteristics of the problem and the problem domain. Questions to ask yourself when writing this section: - *Are the algorithms you will use, including any default variables/parameters in the project clearly defined? - Are the techniques to be used thoroughly discussed and justified? - Is it made clear how the input data or datasets will be handled by the algorithms and techniques chosen?*

For this project I will be using convolutional neuro networks to solve this problem. This problem of multiclass image categorization is commonly addressed with deep convolutional nets. I used these techniques in two major ways: for a model from scratch and finally through transfer learning. The deep learning framework used for this project is entirely pytorch.

Model from Scratch A convolutional net gets its name because it includes stacks of convolutional layers. These layers essentially uses a kernal or filter of certain size to scan an input matrix representing an image and creates a convolutional layer. These layers are stacked on top of one another to help discover more complicated features. (https://michhar.github.io/convolutional-in-layers-and-sequences/) A convolutional net typically includes a few other layers including: A pooling layer is used reduce the number of parameters in a model. An activation layer is used to identify whether a neuron is activated and is essential for introducing non-linearity in the model, which is essential for identifying unique features. ReLU (rectified linear unit) is the standard activation function used. This function looks like this:

```
In [112]: # from: https://github.com/colah/nnftd/blob/master/fig/chap3/relu.py
          z = np.arange(-2, 2, .1)
          zero = np.zeros(len(z))
          y = np.max([zero, z], axis=0)

          fig = plt.figure()
          ax = fig.add_subplot(111)
          ax.plot(z, y)
          ax.set_ylim([-2.0, 2.0])
          ax.set_xlim([-2.0, 2.0])
          ax.grid(True)
          ax.set_xlabel('z')
          ax.set_title('Rectified linear unit')

          plt.show()
```



Another layer I used heavily is the batch normalization layer. This is used to normalize the input layer and is generally applied to improve speed, performance and make models more stable (https://en.wikipedia.org/wiki/Batch_normalization). To help prevent overfitting I also employed dropout, which essentially drops out parts of a channel when training a model. This helps the model generalize better by artificially adding noise to the training data.

When building my model from scratch I chunked together the following configuration of these layers:

convolutional > relu > batch normalization > pooling

I stacked 5 of these chunks together and then put the dropout layer at the end and connected that to a linear layer that produces the classification results by ending in 133 features, one for each dog breed.

Transfer Learning Model The transfer learning model was constructed differently. Transfer learning uses an existing model that has been pre-trained by others. The main benefit of using transfer learning is to save yourself the costs (in time, data, and money) of building a robust model from scratch.

The basic method uses is to: 1. Import a pretrained model 2. Freeze all the layers (layers weights will not be updated) 3. Unfreeze some final layer(s) to have its weights updated when training. In my case I just replaced the final layer with new linear unit with 133 output features to match the number of dog breeds.

Model optimization technique In order to train the models (from scratch and transfer) I used stochastic gradient descent to minimize the cost functions and in the