Using Convolutional Neural Network to Predict Dog Breed

Rachel Tumulak

# Project Overview

Image classification is the process of identifying through computer vision the visual content of an image.

There are a lot of applications image classification can be used for such as face recognition, self-driving

cars, automatic vacuum cleaner and a lot more. Common to these applications include identifying an

object. An example is identifying whether a human face is in a picture or not to recognize face of a

human. For a self-driving car to be able to drive automatically, it needs to identify stoplights, signs, and

humans to drive safely. Vacuum cleaners need to identify if an object is a dirt and needs to be cleaned,

or a wall to change its direction.

There are a wide range of species in the world, human with a curious mind sometimes just wants to

know what specie that is. In each specie there are a lot more classifications. An example would be

animals. There are a lot of animals in the world that some even look similar. Tigers, lions, leopards have

common features and they belong in the Cat family. There are also domestic cats which are separated in

terms of its breed. For a human to detect what breed a cat would be is challenging due to many factors.

A dog breed can be identified through behavior, body type, face, and ear shape, vocal, color, fur,

markings and patterns, body size and mannerism. In an image one can only focus on what can be seen

namely body type, face and ear shape, color, fur, markings and patterns, and body size.

Machine learning, as a subset of artificial intelligence, provides system the ability to learn and improve

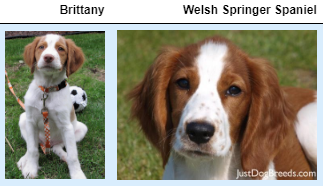
automatically through experience. Image classification is one the topics in machine learning and a

common deep learning algorithm that is being used to analyze visual imagery is Convolutional Neural

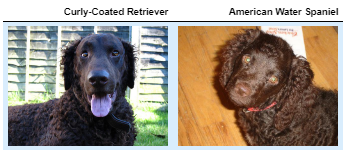
Network or CNN.

# Problem Statement

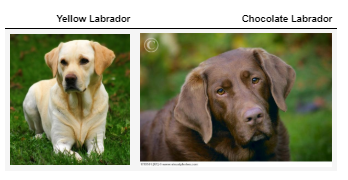
This project aims to build a machine learning pipeline that will identify an estimate of a canine’s breed. The most identifiable characteristic in a canine is its coat color, pattern, and length. Consider that even a human would have trouble distinguishing between a Brittany and a Welsh Springer Spaniel



It is not difficult to find other dog breed pairs with minimal inter-class variation (for instance, Curly-Coated Retrievers and American Water Spaniels).

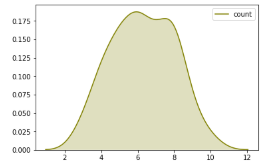


Likewise, recall that labradors come in yellow, chocolate, and black. Your vision-based algorithm will have to conquer this high intra-class variation to determine how to classify all of these different shades as the same breed.



Datasets and Inputs

* unbalanced dataset of images representing 133 different dog breeds



*Figure A. Distribution of cat breeds*

* there are 8351 total dog images
  + 1/10 of the total images are to be used for testing
  + 1/10 of the total images are to be used for validation
  + 8/10 of the total images are to be used for training
* There are 13233 total human images
* For training the images will be transformed:
  + Random rotation
  + Random resize
* All datasets are resized to fit the model

The dataset was made available by Udacity Machine Learning Engineer Nanodegree Program

Solution Statement

A proposed solution to this problem is to undergo two approaches of Convolutional Neural Network.

1. **Traditional**. I will then construct a Convolutional Neural Network from scratch that will identify cat breeds.
2. **Transfer Learning.** I will use an existing CNN model that has been trained and use its knowledge for training the newer model.

First, I will use VGG16 pretrained model to detect if a dog exists in a picture. All images will be transformed into randomly rotated and resized for training. Second, I will build a convolutional neural network from scratch to predict the breed of the dog. Third, to improve the model, I will use an existing CNN model that has been trained, ResNet18, and use its knowledge for training the newer model.

According to the observations of researchers, in convolutional neural network, the deeper the better. However, even if the models tend to become more capable after some depth the performance degrades. When the network goes too deep, calculating the gradients from a loss function shrinks to zero after several application of the chain rule. Which results to no learning performed. ResNets solves this problem by allowing the gradients to directly flow through the skip connections backwards from the later layers to initial filters.

# Implementation

## Metrics

To evaluate the model, I will use F1 score to determine model performance. F1 score is the weighted average of the precision and recall. Where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal.

### Formula A. F1 Score formula

## Data Processing

For data processing, Validation Dataset and Testing Dataset are resized to (244, 244). Randomly selected from Training Dataset are resized and cropped to 244, and rotated to 15 deg. All datasets were normalized to be accepted by the models.

## Face Detector

For the implementation of the Cat Detector, I made use of OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images. OpenCV provides many pre-trained face detectors, stored as XML files on github.

## Dog Detector

For the implementation of the Dog Detector, a pretrained VGG16 Model, along with weights that have been trained on ImageNet, a very large, very popular dataset used for image classification and other vision tasks, was used. ImageNet contains over 10 million URLs, each linking to an image containing an object from one of 1000 categories. Given an image, this pre-trained VGG16 model returns a prediction (derived from the 1000 possible categories in ImageNet) for the object that is contained in the image. To detect if a cat exists, the output falls under the range of 151-268, inclusive, to include all categories from 'Chihuahua' to 'Mexican hairless'.

## CNN from Scratch

I created a CNN from scratch, and I attained a test accuracy of at least 10%. Setting aside the fact that the classes are slightly imbalanced, a random guess will provide a correct answer roughly 1 in 67 times. I built the model having three sequential layers and applied a 2d convolutional layer with a kernel of 5, stride of 1, and where bias is true, a 2d max pooling over an input signal composed of several input planes, and a rectified activation unit. To compute for loss and optimization I used a Cross Entropy Loss and SGD optimizer.

## Transfer Learning

I used the same dataset for the CNN Scratch. For the CNN using transfer learning, I applied finetuning, where instead of random initialization, I initialized the network with a pretrained network, like the one that is trained on ImageNet. I loaded the pretrained model restnet18 and reset the fully connected layer having several filters of 512 input features and 133 output features. I used the same loss function and optimizer for the CNN using transfer learning which is Cross Entropy Loss Function and SGD Optimizer.

## Post Processing

I defined a function that takes an image path as input and returns the cat breed that is predicted by your mode. The algorithm accepts a file path and first determines if there exists a cat in a picture. If there is a cat, predict what breed it is, if there is not cat, provide a resembling cat breed.

# Model Evaluation and Results

## Face Detector

To check the accuracy of the model, I tested it with 100 images of dogs and 100 images of humans. I attained an 98% accuracy using a pretrained OpenCV to detect face in an image of a humans. 17% of the face was detected on dog files.

## Dog Detector

To check the accuracy of the model, I tested it with 100 images of dogs and 100 images of humans. I attained an 100% accuracy using a pretrained VGG16 to detect dog in an image of a dog. No dogs were detected on human files.

## CNN from Scratch

I trained the model with 10 epochs having a training and validation loss as displayed in the figure. The valid loss increased some time after the least valid loss. I loaded the model with the least validation loss.

### valid loss train loss

### Figure B. Graph of CNN from Scratch Train and Valid Loss

For testing the model from scratch, it produced an F1-Score: 12% and test accuracy: 13% (112/836).

### 

### Figure C. CNN from Scratch Trivial Confusion Matrix

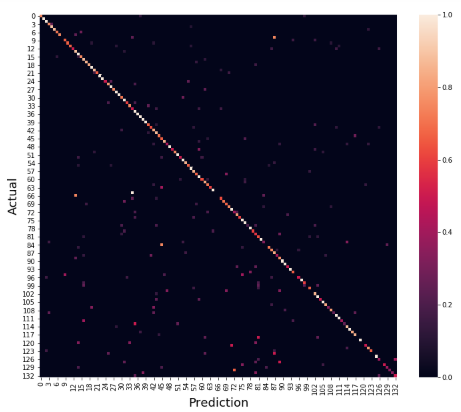
Based on the trivial matrix shown in Figure, we can see that the diagonal values are 0 or black which means that the model has not identified those labels and has a value of 1 recall for the Komodor breed. This can be due to the number of images Komodor has.

## CNN Using Transfer Learning

### valid loss train loss

### Figure D. Graph of CNN using Transfer Learning Train and Valid Loss

The training and valid loss for CNN using transfer learning decreases as the epochs increases. This means that the training is stable and is improving overtime.



### Figure E. CNN using Transfer Learning Trivial Confusion Matrix

The result for CNN using transfer learning is 73% (617/836) for test accuracy and 75% for f1-score. We can see from the figure that the diagonal line is colored and there are some colors that are white which denotes that the f1-score for that class is almost 1 which means that it is a good classifier.

# Conclusion and Recommendation

Clearly using transfer learning has improved the model dramatically. This also saves time for training since it is a pretrained model and a lot of data has been used on this model. For improvement of the system I would recommend the following:

1. Increasing the number of epochs
2. Testing out ConvNet as fixed feature extractor
3. Balancing the number of images for the dataset