CS 82 Final Project: Dog Breed Detection System

Sharjil Khan

1. **Introduction & Motivation**

Classifying dog breeds correctly allows adoption agencies to look at a dog’s likely traits based on their breed. This allows them to explore the dog’s differentiating personality as well, so that they can provide a complete picture to possible adopters. Further, adoption agencies often work with volunteers who may not specialize in breed identification and having a program that can evaluate a dog’s picture to help determine a breed can allow the volunteers to help in the dog’s evaluation.

Detecting the breed of a dog from its visual appearance is not an easy task. The task is made even more difficult by the fact that most dogs are a mixture of different breeds. Having a tool to at least narrow down to a few breeds when trying to determine the breed of a dog would be of great help to many agencies that deal with dogs.

1. **Scope of the Project**

Given the time limitations to implement this project, I have narrowed down the scope of this problem to a set of simpler more manageable problems. I will first try to train models to differentiate between dog breeds that are more easily distinguishable. For example, breeds with different colors or a major size difference. I will gradually try to add more dogs and see how we can adapt the models as the problems become more difficult.

The final goal of this project is to differentiate between 8 dog breeds with an accuracy level which would be significantly better than blindingly assigning the dog to a specific breed. I want to be able to get to a point where given enough training data, I could conclude that the models would be able to differentiate between the dog breeds.

I believe Convolutional Neural Networks (CNN) would be the ideal model to use for this problem. Because CNNs are in general great for pattern recognition in images. However, given the reduced scope of the problem, I also intend to try out another simple color pixel counting method and other kinds of classifiers such as LDA or KNN to see if we could achieve similar results that a CNN model would.

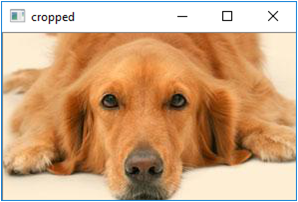
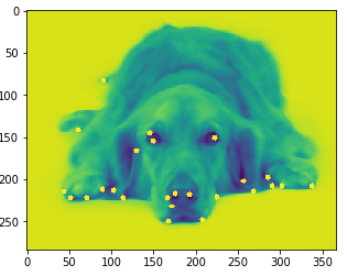
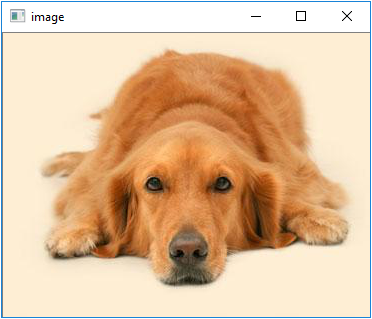
1. **Data Used**

The data used for the project comes from Stanford Dogs Dataset**:** <http://vision.stanford.edu/aditya86/ImageNetDogs/main.html>.

The data is structured into directories of dog images. The images are in sub folders based on their breeds. As a first step, I split the data into a training, validation and test set and use Scikit-learns data generator modules to feed the data into the models in batches. The pictures in the dataset. The pictures have different pixel counts that make it a challenge to compare them against each other. The dogs we want to detect are also part of a larger environment, which introduces noise in the data. I tried to use some libraries in python to try to address some of these issues and focus on the more important parts of the images to improve accuracy scores. These methods are described in detail in the following sections.

1. **Steps taken to Prepare the Data**

An initial exploration of the images, made it clear that the data is very noise. In a lot of the images, the dogs are occupying a small part of the overall image which contains a lot of other objects in the environment. In some of the pictures, there are even objects in front of the dog, partially covering the dogs. For example even in a simple picture like the one shown in Figure 1, more than 50% of the image is occupied by the environment which gives us no information about the dogs breed. I used an open source python module, OpenCV to process the image. The module uses a feature selection function to zoom in on more informative parts of the image. This cuts out the parts that reveal no information about the dog breeds and makes it much easier to learn about the specific dog breed traits with less noise in the training set. I found applying a simple open source filter like this greatly increased the accuracy results.



(Figure 1) In the left image, the dog is occupying around 50% of image area. The middle image shows how the key features selected by the feature selection function from OpenCV. The image is then cut using the co-ordinates from the feature selection function which results in the dog occupying about 75% of the image on the right.

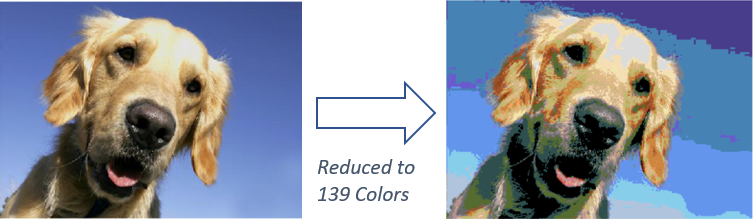
After applying the above mentioned filers, I split the images into a training, validation and test set to be used by Python’s data generator module that can feed the images in batches to the CNN model for training and testing. Apart from this approach, I also used a simple color count strategy to use simpler models such as KNN which is described in Section 5.1.

1. **Models and Approaches**
   1. **A simple Color Based classifier:**

In this simple method I reduce the image down to only 139 distinct colors by converting all color pixels to its closest match using its RGB value and the Euclidean distance between the color and one of the colors specified in the 139 color list.

As I go through the image, I just increment the corresponding columns for each color to create a row with color counts for each color in the image. My motivation of using this data to classify between dogs is to see how much better a CNN model would perform compared to a model that is simple looking at the colors of the dogs and not taking any shapes or patterns into account. Because if the classification accuracy using this method begins to resemble the accuracy using the CNN model, then we know we are not really using the patterns in the image to recognize the dogs.

I also note that reducing the images to just 139 colors still retains most of the distinguishing features of the image as shown in Figure 2. So I believe just counting the 139 major color should still allow us to make pretty accurate predictions.



(Figure 2) Reducing the colors to 139 basic colors retains most of the characteristics of the image allowing us to make our predictions with a reduced set of color predictors.

I will look into increasing and decreasing the number of base colors I use to characterize each image to see how it effects model accuracy. I also want to experiment with leaving out certain columns (colors) such as Blue, Red and Green that are not associated with the dog and hence it would help to reduce background noise from the environment.

In general I would definitely expect the CNN model to outperform this model easily, but this helps serve as a baseline for prediction accuracy that I would definitely want to beat using other methods.

**5.2 Convolutional Neural Network:**

**5.3 Transfer Learning:**

1. **Results**
2. **Conclusion**