Where's Spot?

Final Paper

Dog Facial Recognition Project

Data Analytics Experience
Priya Bhardwaj
Reza Ghaemi
Umang Mittal

Overview

Every day owners lose their dogs, or a dog is admitted to an animal shelter. Owner surrender, lost, stray, etc. The reasons are endless. In any case, the dog is just looking for its forever home. Whether that home is finding its way back to its family or waiting to be adopted.

This document consists of our data storage strategy, data sourcing, cleaning, ETL, our modeling analysis and the UI/visualization of our product. We will also cover final outcome and lessons learned.

Goals

1. Breed Recognition: use facial recognition to identify the dog's breed. This is useful for people coming into shelters looking for a specific breed/mix. Comparing the image of a dog to a purebred dataset will give a guess on the breed of a dog without spending the money on a DNA test.
2. Lost Dog Recognition: Dog owners can post photos of their lost dog, and people who find a stray pup can post a photo as well. Where's Spot will scan both images to see if there is a match between found/lost pups or pups entered at animal shelters. If there is a match, a location and contact information will be displayed.

Tools used for Data storage, modeling and UI/Visualization

I. Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

II. Google SDK

Google SDK is needed to interface with all the services of the google cloud platform. It is used to manage Virtual Machines, IAMs, and any services related to the platform without a UI. This will be used for the initial set up of our platform because it requires SSH private/public key setup and maintain other services.

III. Python

We are using python to build our model using Tensorflow and Keras. Also, to create a web UI we will be using Python interfacing with HTML and Node.js web UI which will provide us with an interface to the whole application. This allows us more flexibility to add more features.

IV. Jupyter NoteBook

For development purposes we are using Jupyter Notebook instead of native python. This is because Jupyter Notebook allows us to run code line by line in comparison to python script which requires you to run the entire code at once.

Cloud Services:

Google Cloud Platform

Compute Engine

This compute engine is an instance/cluster that will be the engine of all computations in our application.

Functions

This service will allow us to automate our ETL jobs based on events and triggers. This service will retrieve and pre-process our images based on our specifications and will aid with loading into our AWS storage.

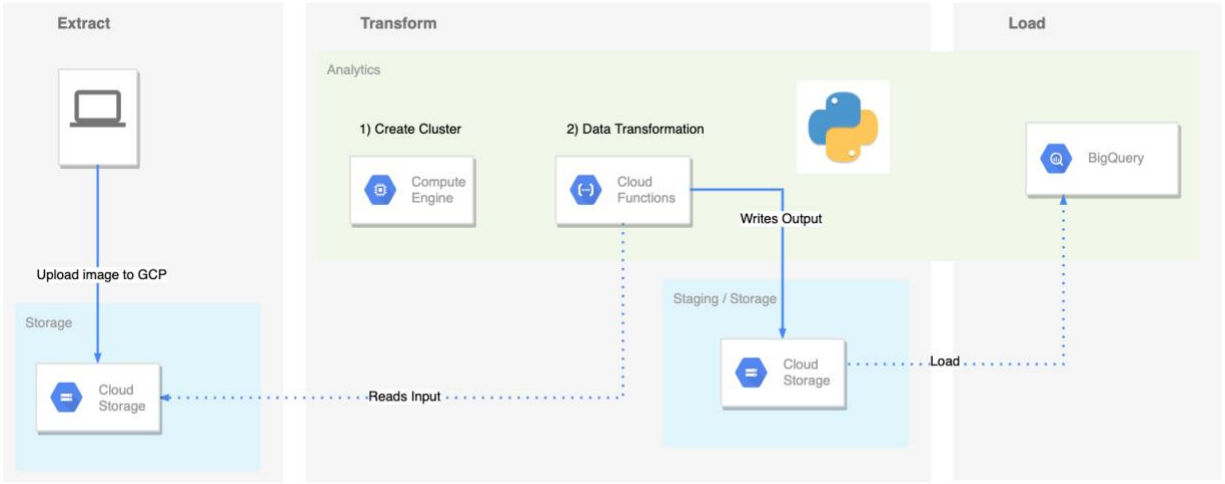
Google Cloud Storage

Google Cloud Storage is an online file storage or a “data lake” that allows world-wide storage and retrieval of any amount of data at any time.

Google Cloud BigQuery

BigQuery is a cloud based, highly-scalable, and cost-effective cloud data warehouse with an in-memory BI Engine machine learning built in.

ETL

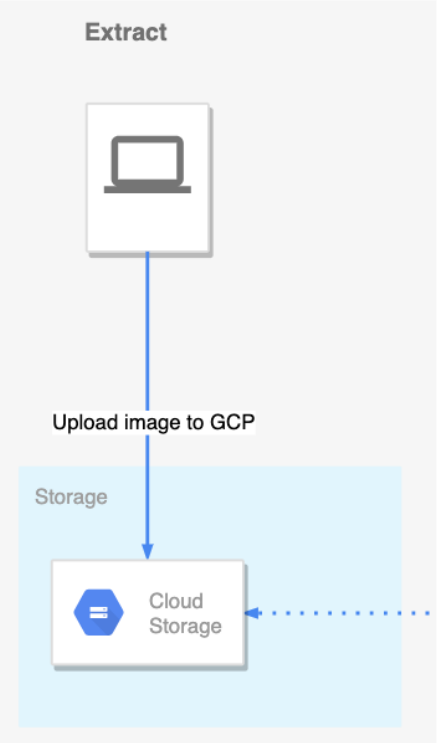


Our ETL process starts with data sourcing and more specifically the user uploading the image using a UI to our google cloud. Once the image is uploaded, that triggers our transformation layer using GCP services. The jobs in our transformation layer are triggered and any modifications are made to the image. Once the jobs are finished we take the images and output them to our storage and stage them to be fully ready to be added to our database for further OLAP analytics once the application scales to support such functionality.

Since we have ample dataset we may use GANs to make the dataset larger for a better model training. For this we will be using tensorflow since it provides better artificial neural networks.

Our Kaggle training dataset requires minimal transformation in the ETL process due to the high quality in the images. However, we will still document and build infrastructure to support transformation jobs of the images for images that require transformation

Data Sourcing



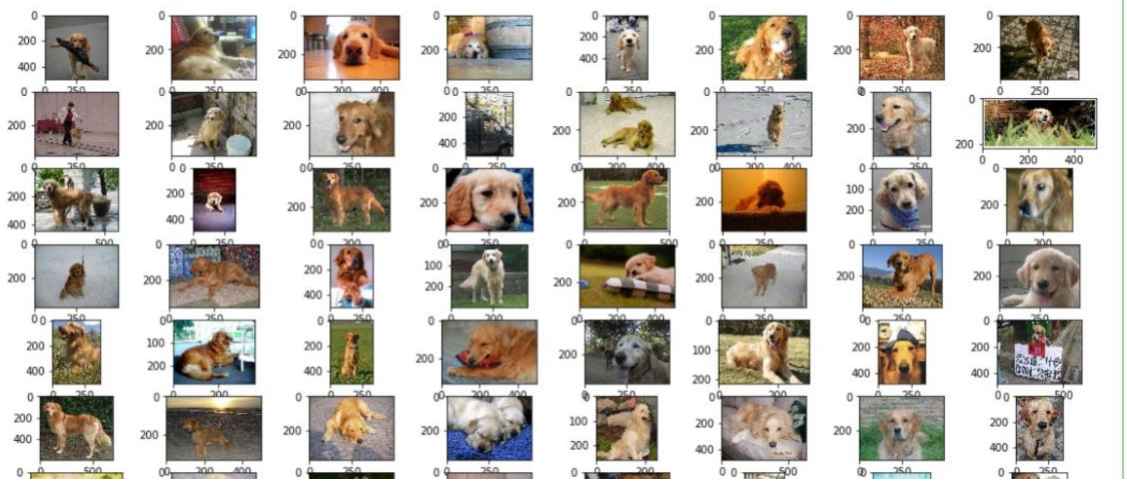
We are sourcing the majority of our training data from Kaggle. This dataset consists of 120 different purebred dog breeds and over 20k images. This will drive the majority of our training. This data will be stored in google cloud platform and will be accessed by GCP ETL jobs for transformation and load. This process also includes the user upload of the image that needs classification.

Also we have used dataset from Kaggle competition for dog breed classification which can be found on the below link. The data in this set has already been separated into test and training sets. There is a limited number of images for each breed. Our goal is for the model to use the user uploaded images with marked breeds to increase the training accuracy.

https://www.kaggle.com/c/dog-breed-identification

Data Sample

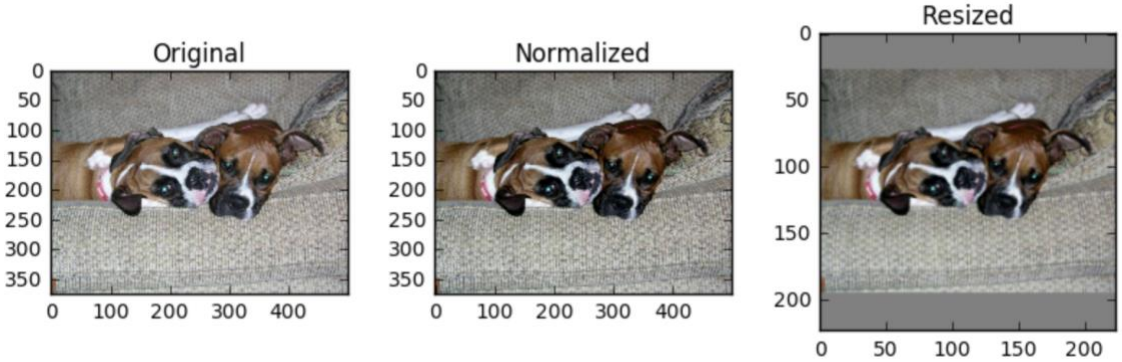
```
In [94]: import matplotlib.pyplot as plt
         import matplotlib.image as mpimg
         from matplotlib import rcParams

         %matplotlib inline

         # display images
         fig, ax = plt.subplots(8,8)
         fig.set_size_inches(18.5, 10.5)

         for i,a in enumerate(ax.flat):
             a.imshow(mpimg.imread(file_name[i]))
         fig.show()
         # ax[1].imshow(img_B);
```



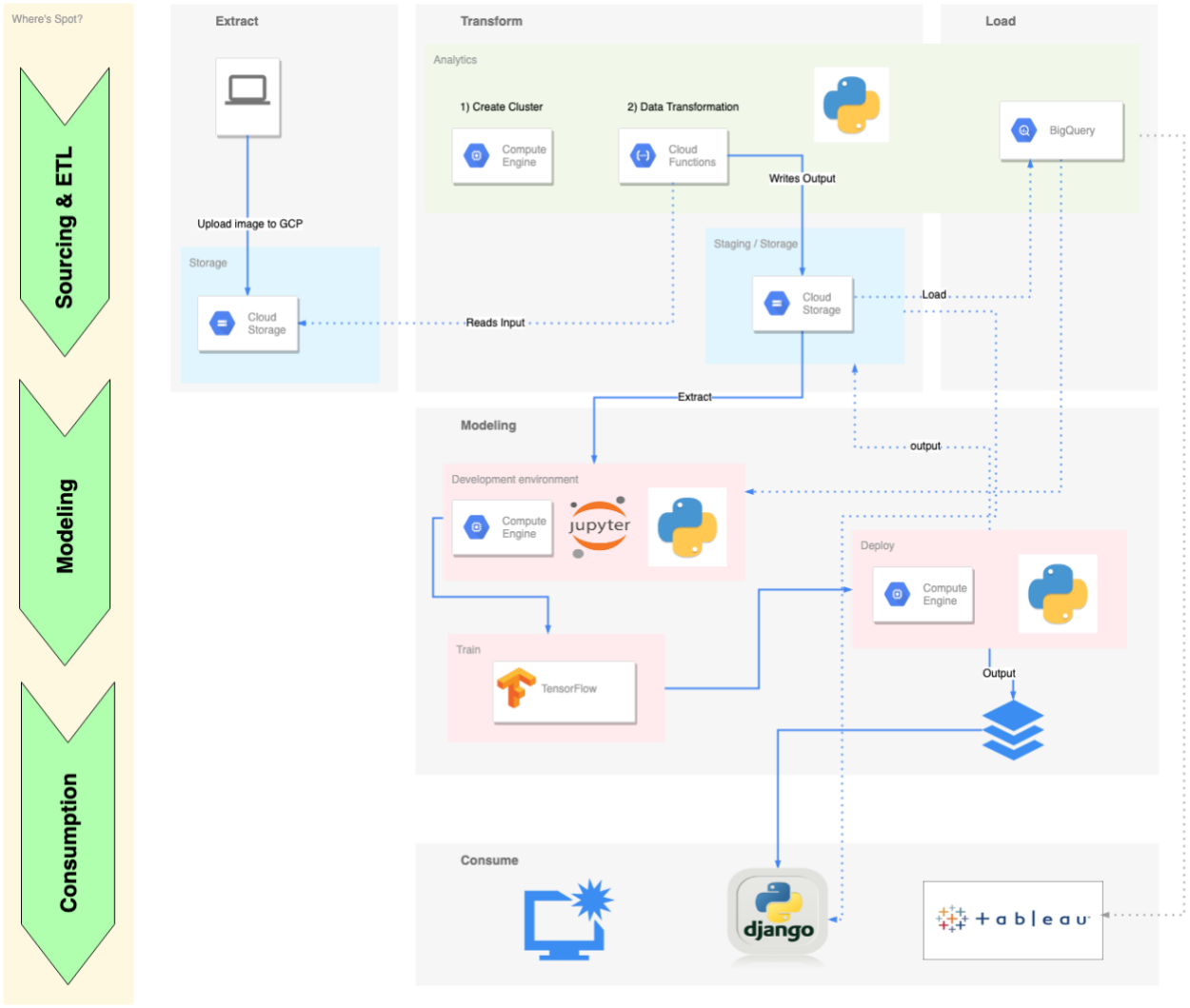
Data Cleansing

ETL transformation jobs:

- Look for inconsistencies in the image
  - If the image of the dog has a busy background or if the dog is not the sole focus of the image. These images can cause false positive predictions and therefore will get filtered out in our training.
- Classify if it is a dog and not another animal
  - We could get images of cats and other animals and need to make sure our process only picks up dog images
- Resize and normalize the image
  - This will allow a more consistent classification with higher accuracy



Data Storage Strategy Architectural Diagram

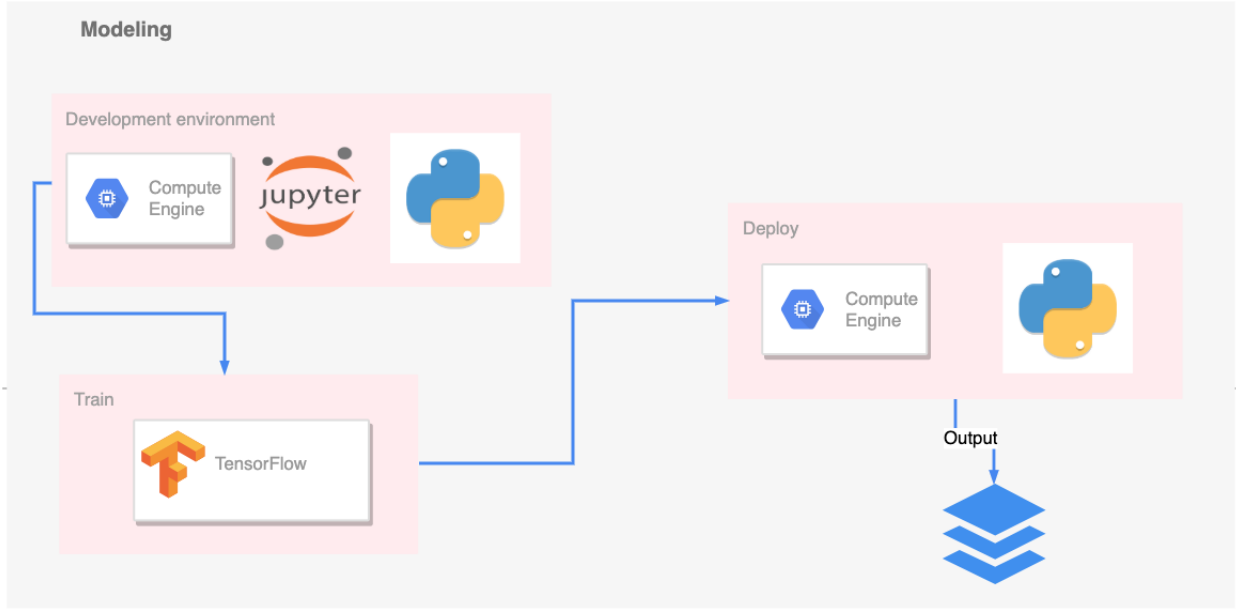


Our data storage strategy is embedded in our ETL workflow. Our ETL process leverages our Google Cloud Platform storage as a data lake. Once the data is sourced either via a user input or from the training dataset, we land and store the data in the data lake which enables and triggers our transformation layer to read the data and apply our ETL rules engine to transform the images. The second step to the storage of our data is taking the transformed images after our ETL logic and land them in another directory in the "data lake" which essentially acts as our staging environment. The data being stored in the staging environment is ready to be consumed by the modeling layer.

In our application architecture we have also included the pipeline to send usable data to Google BigQuery which will enable reporting and other visualizations on the source data. The data that is being landed in BigQuery has another set of transformation rules applied to be able to be consumed by the data warehouse to allow Tableau reporting and other reporting requirements that may be needed. Reporting is very important as it allows us to answer critical operation questions that may be essential for the product.

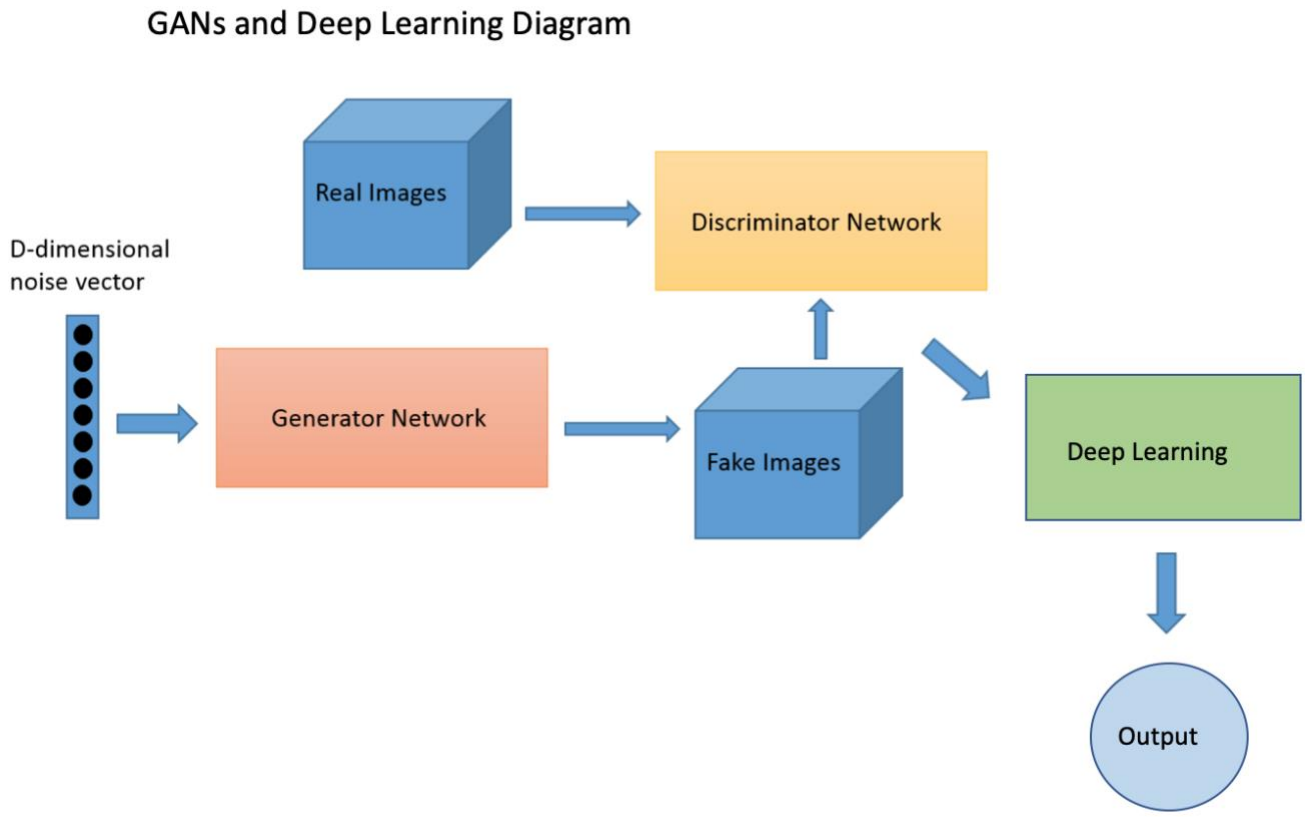
In our AI engine, we pull the data directly from our staging environment as it is most suitable for consumption by the models we have created. Since we have a smaller dataset we used GANs to make the dataset larger for a better model training (more about GANs later in this document). These GANs generated images will also be landed in our staging environment and allow the model to extract data.

Modeling and Analytics:



For building the actual model to predict the breed of a new record (in this case, Dog Image), we will use Jupyter notebook for development purpose since we will have to go through multiple train and test scales to fix the parameters of the model, after which we are going to migrate all the code written on Jupyter to a native python script. A native python script will be more compatible with the web UI.

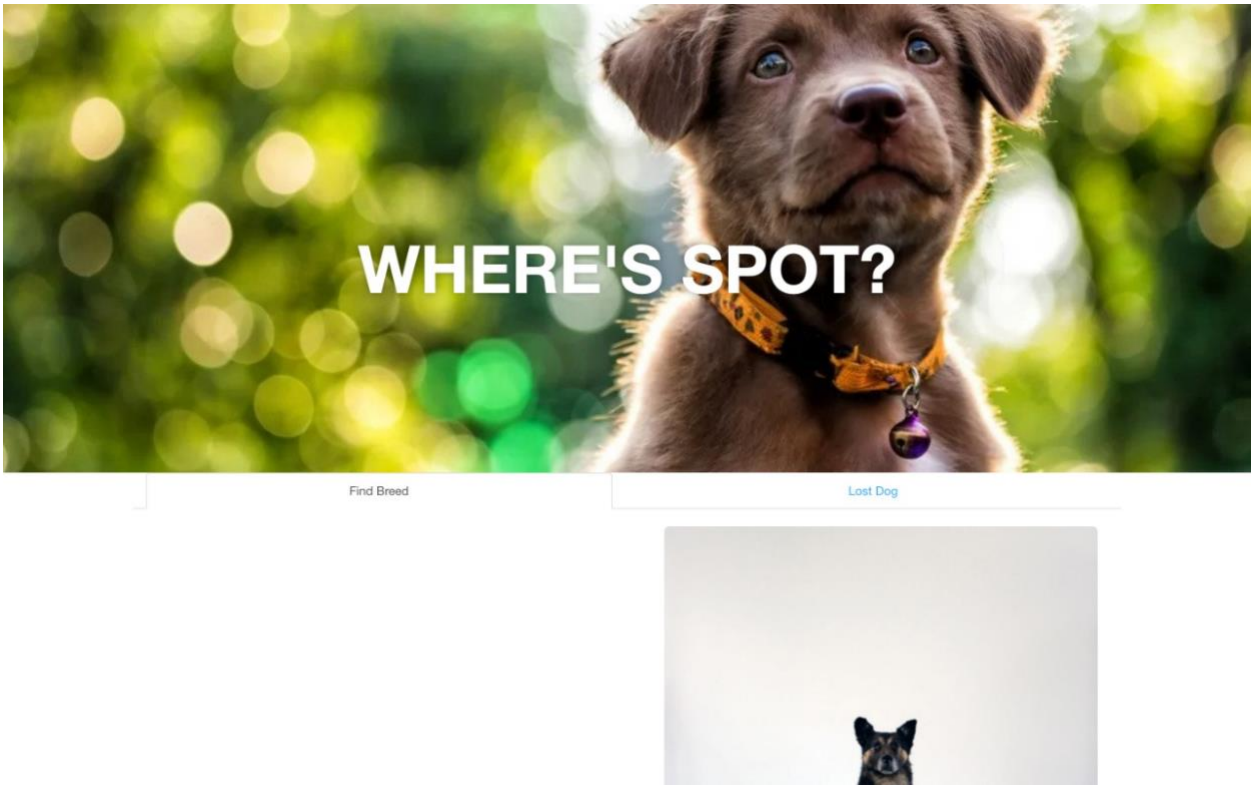
The model build will leverage Keras which is a high-level API for TensorFlow. This will allow us to reduce the size of the python script drastically since using TensorFlow natively is lengthy. Keras provides us enough flexibility and ease of use. After we have generated the model this will enable the presentation UI layer.



The modeling process can be explained in 3 steps. First step is the discriminator network of GANs which is nothing more than a classification learning model that takes real images and builds a predictive model. This discriminator becomes the life line of GANs as it minimizes the errors that the generator network creates. The generator network is what make GANs special, it uses the model created in the discriminator network to generate 'Fake' images by applying random noise via the D-dimensional noise vector. It is important to note that these fake images are created only within the breed of the dog and will be added to the master list of images we have landed in our staging environment before modeling. Our actual model then pulls all images in the staging environment (fake and real) and applies CNN deep learning algorithm via tensorflow. This strategy gave us the highest accuracy and precision in our validation.

We found that the optimal number of hidden layers in our CNN model is 6. After several iterations we came to this conclusion that 6 layers allows us to train our model with positive results without overfitting our dataset.

UI / Visualization Layer:



Once the model is deployed and the output is either captured via raw files or files pushed to our ETL process, we will enable the UI to read off of the outputs. We have two different approaches for our visualization and UI strategy. First approach is creating a web UI via Django to be the portal and UI of the application and the other is using Tableau to read off BigQuery for other analytics to aid with data management and operations. Due to the nature of our project, we have chosen a web UI over Tableau to be the UI of the application.

Our web portal for the application is being built off of Django as its free and open source and makes the web development process very easy which enabled us to full focus on the business implication of the application and fine tuning our model. Django is an ideal tool for us, where web design is the need, to bring out the real concept and prospects of our idea.

In the final output of Where's Spot, Tableau was unable to be integrated in the given time. This feature will be a future addition.

Lessons Learned:

This project allowed us to work on several new tasks:

Pre-processing for images
TensorFlow
GANs
Django
Image Recognition

Over all, the model did not work entirely as expected. After GANs, we were expecting a higher accuracy on the test and train data. In addition, we were unable to add the test data to our GANs model. This was because the dataset was not labeled, and it would have been difficult for us to label it. Another big lesson was that we had originally planned to use AWS and a multi-cloud strategy. Due to configuration issues, cost and ease of use, we solely used Google Cloud.