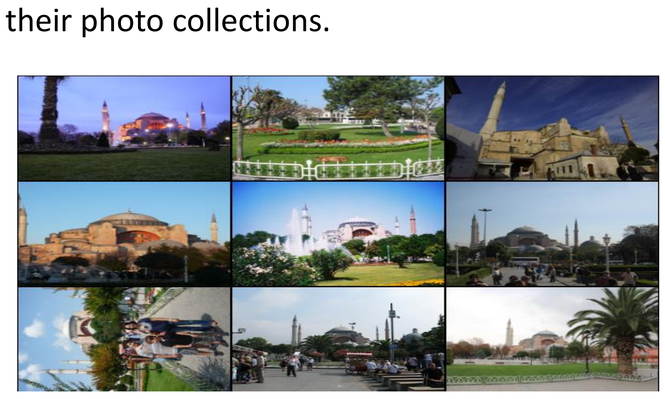
Google Landmark Recognition Challenge

Computer Vision Lab
Talha Bilal
talhabilal821@gmail.com

Problem Description

This competition challenges to build models that recognize the correct landmark in a dataset of challenging test images. This technology can predict landmark labels directly from image pixels, to help people better understand and organize their photo collections.



Obtaining the Image Data and Preprocessing

Data was in the form of CSV files with image URLs and landmark IDs. Our sample of 200 classes contained class labels (Landmark IDs) from '1000' to '1200'. The download of images of different resolutions takes time (major time is spent in opening the url link). In the interest of time, we decided to work on images of resolution 96x96 rather than the full resolution for our project. Final results were produced on a data sample of 200 classes with resized images.

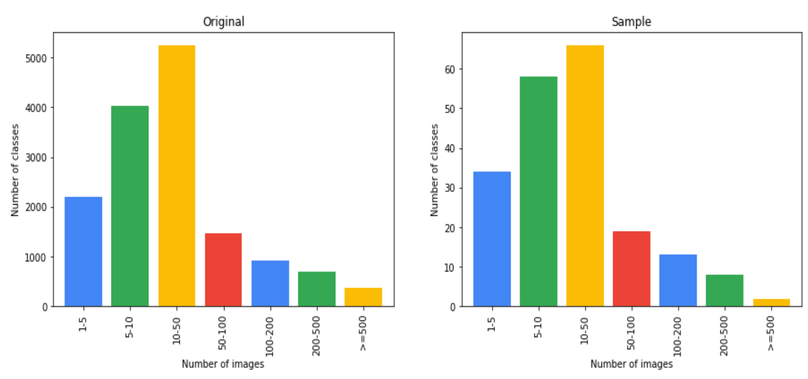


Figure 1: we have used a subset(sample) of original dataset. In sample dataset each class has same proportion of images as in original dataset.

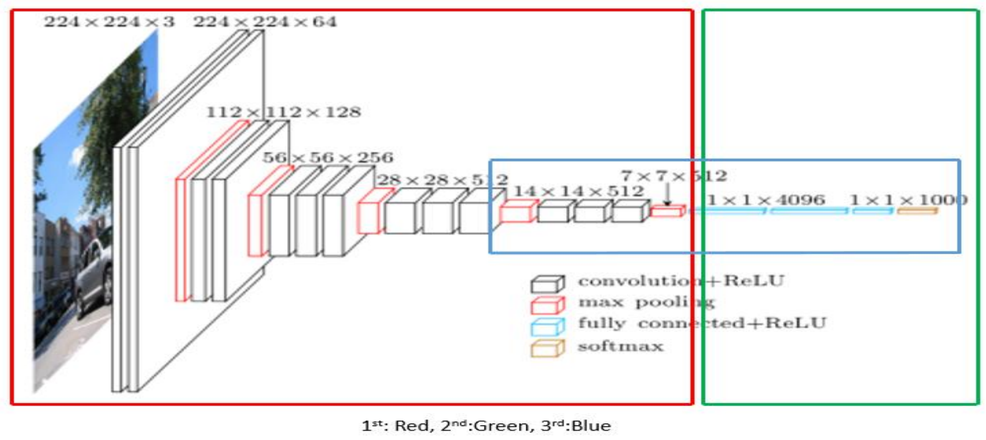
Data pre-processing for this project can be broadly classified into following steps

a) **Train-validation-holdout data:** We took 1% images from each class to make the holdout dataset. Next was the train validation split on the remaining 99% data. 20% images from each class were labelled as validation set and the remaining 80% were used for training.
b) **Fetch the image files:** Next step was image download in appropriate folders. Three separate folders were created, one for each train, validation and holdout set. Resized images (96x96) were downloaded to respective folders.
c) **Make the directory structure:** Each class is a subfolder inside Train/Validation folder. This subfolder contains all the image files belonging to that class.
d) **Data Cleaning:** Many of the downloaded links were broken so we have to manually check each folder for those broken links

Model Training

Different CNN architectures can be found in the Keras library for example VGG16, VGG19, Inception, etc. While VGG19 and Inception are more heavier architectures (number of parameters to train), training VGG16 was doable in the time frame that we had. So our process looked like this:

1) Convert images (96x96) to vectors using Image Net weights on bottleneck layers of VGG16.
2) Initialize the weights on top three layers. For this, a model having just three dense layers (2 ReLU activation, 1 softmax) was trained.
3) We chose to freeze only bottom 16 layers as opposed to 19 so the network can learn better for the images outside the Image Net images.



1st: Red, 2nd:Green, 3rd:Blue

Batch Size	50
Number of Epochs	32
Optimizer	Adam(lr(0.001), Beta1(0.9), Beta2(0.99))

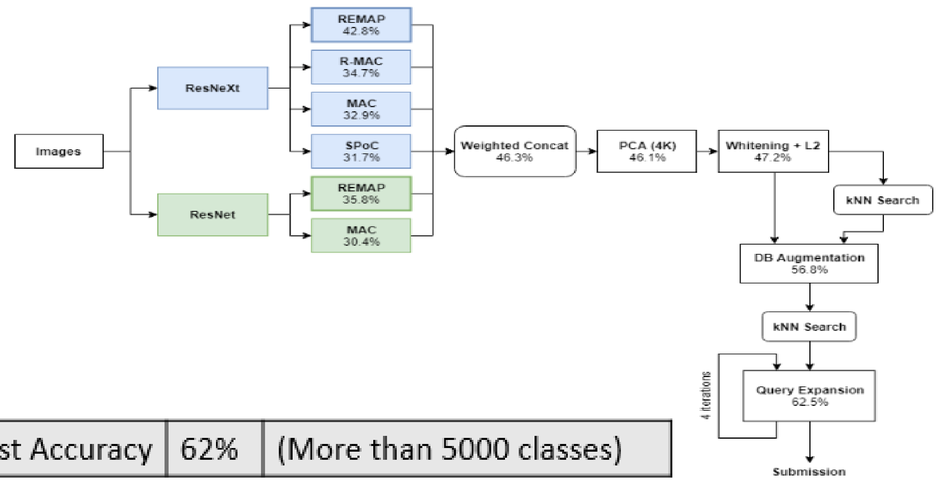
Results

Training Accuracy	76.22%	200 classes
Validation Accuracy	74.07%	200 classes
Test Accuracy	81%	200 classes

Problem with the Solution:

It is important to note that this prediction step ignores the fact of not having any landmark to classify.

1st Rank Solution



Test Accuracy	62%	(More than 5000 classes)

Future Work

In the future we are planning to focus on data cleaning. For example their were many classes with less than 3 samples. We can remove those classes and that will improve our results. We can solve our problem of having a landmark or not by using DeLF.