ECE324: Final Report - Monumentum

Arsh Kadakia (1005228451), Sepehr Hosseini (1005082232), Haoran (Jayce) Wang (1004927163)

December 6, 2020

Word Count: 1988, Penalty: 0%

1 Introduction

The broad vision for our project Monumentum is to enrich tourism by providing tourists the background information they need to fully appreciate the history and significance of their destinations. To realize this vision, the supporting core technology and our main goal is to recognize a landmark or location given an image. Our approach is centered around image-based input as we believe taking photos with smartphones would be one of the most convenient ways to query information as it does not require any preliminary knowledge from the tourist regarding their location or landmark. Due to the sheer number of existing landmarks along with the variability of image-based input, the recognition process must be both scalable and generalizable. Therefore, we believe machine learning is an appropriate tool for the task.

2 Illustration / Figure

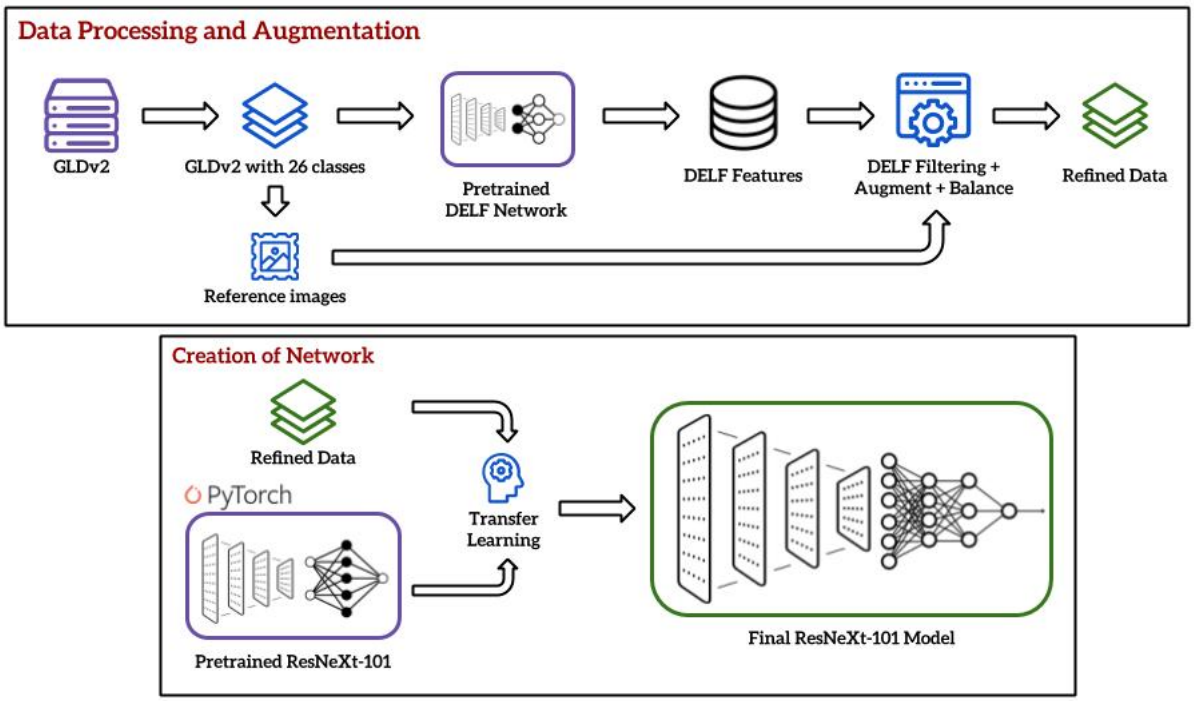


Figure 1: Illustration of data processing and training pipelines.

3 Background and Related Work

Many models were built to succeed on the GLDv2 dataset. A notable example is the 1st-place solution for the dataset [1]. An EfficientNet model was trained on a smaller version of the Landmark dataset. The final model then applied **transfer learning** using weights from the original model and retrained the fully-connected portion on the full dataset. The success of transfer learning on this dataset made it a promising approach for this project.

In order to remove "irrelevant" images in the dataset, the 1st-place solution also used DELF (Deep Local Features), a machine-learning pipeline that analyze and rank the images of the dataset based on how many

common features (inliers) they had with other images within the dataset [2]. This approach was interesting to us because it demonstrated how we could keep only images that had strong similarity to a set of reference images we chose (allowing us to customise the dataset to our expected distribution).

4 Data and Data Processing

The primary source of the data is taken from the GLDv2 Dataset consisting of over 5 million images and 203K classes, where each class represents a unique landmark consisting of several images of that landmark. The labels in this dataset correspond to a unique landmark ID ranging from 0 to 203093 (e.g. 70644 corresponds to the Sonnenstein Castle). The overall framework for collecting and handling this data is shown in Figure 2 below.

Figure 2: Framework used for Data Processing.

4.1 Dataset Scoping and Web Scraping

The first step in scoping was to take one subset of all the classes. The original dataset consisted of anywhere between 1 to 1000+ images per class, so we selected 50 classes, each ranging from 400 to 700 images. This ensured our dataset was relatively balanced while having enough images to learn from. From these 50 classes, we selected 26 to further lower the size of the dataset. We merge this with our webscraped images (2 images per class) using Selenium [3]. By this point we had 11.3K images across 26 classes.

4.2 Cleaning and Filtering with DELF

Due to the crowd sourced nature of the dataset, several classes consisted of images that inaccurately represented the class. To remove these images, we took advantage of a filtering mechanism *DELF (Deep Local Feature)*, pretrained on the GLDv2 Dataset with source code found on GitHub. This software takes in two images as input, and returns the number of inliers (matching features) between them. For our purposes, we compared all images of a given class with 3 "representative images" for that class. If the number of inliers between an image and all 3 representative images for that class is below a given threshold (chosen as 20), then we discard that image.

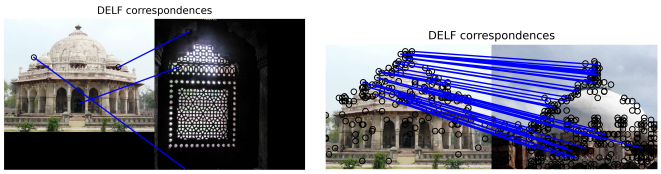


Figure 3: Example of inliers found between images.

4.3 Balancing and Finalizing Dataset

To ensure certain classes are not over/under represented, we perform balancing. Our methodology for balancing is that for any given class, when the total number of images is above a "target" then we discard any extra images, and when it is below then we augment (with Horizontal flip, Gaussian Noise, Cropping) to make up for the missing images. Our target is the median of images per class across all classes. Finally, the all images in the dataset are resized to 128x128 and normalized. The final dataset consists of 26 classes, 299 images per class, and 7.7K total images. The dataset is split into train/validation/test (80:10:10) and ready to be passed in.

5 Architecture

We trained many interesting models to see which one performed the best. The first model was the baseline LeNet model (described in Section 6).

The second model was the VGG16 model, with 13 convolutional layers and 3 fully-connected layers [4]. The fully-connected portion takes in a 25,088-dimensional convolutional output and its first & second layers consist of 4096 nodes, followed by a ReLU activation and Dropout layer with dropout probability of 0.5. The final linear layer ends in 26 nodes due to the 26 classes within the dataset with a softmax activation.

Figure 4: Definition and Illustration of VGG [4].

The third model was the ResNet-34 model, with 33 convolutional layers and 1 fully-connected layer with softmax activation [5]. This layer takes in a 512-dimensional convolutional output and provides a 26-dimensional output.

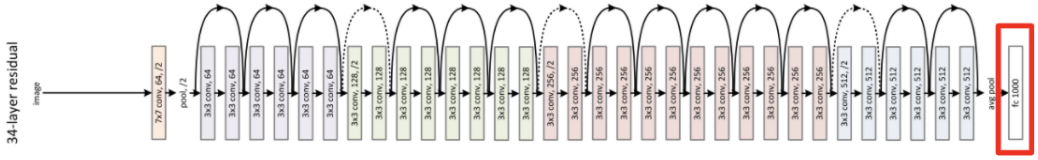


Figure 5: Illustration for ResNet-34 [5]. The dotted lines represent shortcuts with convolutional layers instead of regular "skip connections" that transform the input feature map to the required output channel size for summation. See Figure 7 for a more detailed explanation. Note that the /2 at the beginning of every colour change is referring to a stride of 2. This is a property shared between ResNet34 and ResNext101.

layer name	34-layer	50-layer	101-layer
conv1	$7 \times 7, 64$, stride 2		
	3×3 max pool, stride 2		
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
conv5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	average pool, 2048-d fc		

Figure 6: Model Definition for ResNet-34, ResNet50 and ResNet101 [5].

The final model was the ResNext101 model, with 100 convolutional layers and 1 fully-connected layer with softmax activation [6]. This layer takes in a 2048-dimensional convolutional output and returns a 26-dimensional output. A notable refinement our group made was adding an extra fully-connected layer of 2048 nodes with Dropout (probability 0.5) before the final layer to increase accuracy and reduce overfitting. This version of ResNext is called "Modified ResNext" in comparison to the "Original ResNext".

In the convolutional portion of ResNet-34 and ResNext101, there are 4 "superblocks", each with varying number of 2-layer blocks. ResNext101 follows a similar definition to ResNet101 in Figure 6. Three notable changes are the inclusion of feature paths (seen in Figure 7), the different input channel size of 32 to the 1st

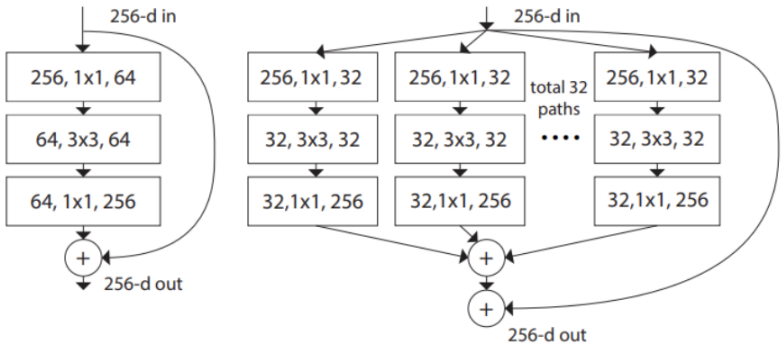


Figure 7: Illustration of a "block" in ResNet101 & ResNext101 [6]. In ResNet-34, the input passes through a set of convolutional layers and is then summed with the output from that set. In ResNext101, the input is sent through 32 different paths and is summed with the output from all paths.

superblock and how input channels to every superblock are *decreased* by a factor of 2.

Finally, note that the convolutional portion for each model (except for the baseline) are *frozen* with pre-trained weights on the ImageNet dataset. Only the fully-connected portion are trained for those networks, as the approach of **transfer learning** is utilised for the problem.

6 Baseline Model

The chosen baseline model is the architecture known as LeNet. LeNet is a small CNN with two convolutional layers followed by three fully-connected layers. Both convolutional layers use a kernel size of 5 by 5 and stride 1 with output channels 6 and 16 for the first and second convolutional layers respectively. Furthermore, both convolutional layers are followed by an average pooling layer with kernel size 2 by 2 and stride 2. The first two fully-connected layers have 120 and 84 neurons while the last one has as the same number of neurons as output classes (26 in our case).

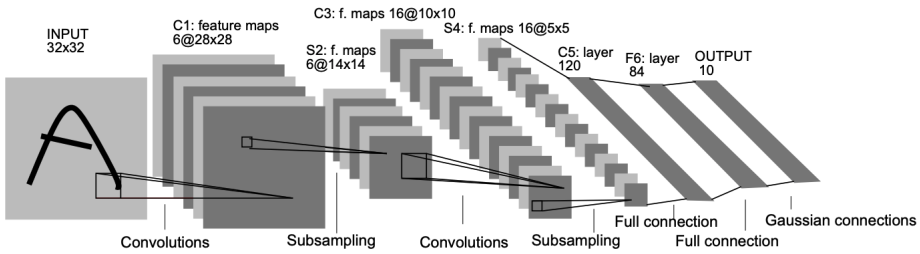
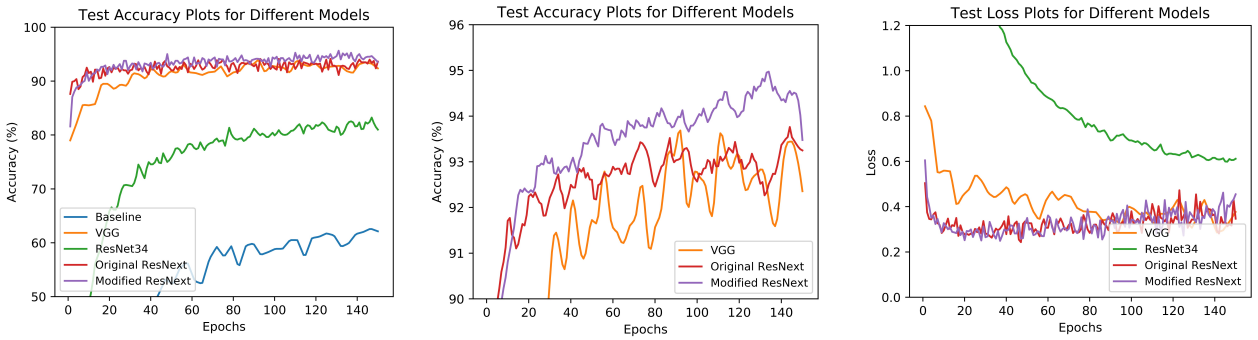


Figure 8: Illustration of LeNet architecture. Note the input dimensions shown are 32x32x1 with 10-dimensional output while ours were 128x128x3 with 26-dimensional output. [7]

7 Quantitative Results

For our models, **top-1** test accuracy was the primary evaluation metric. This metric is defined as the percentage of predictions in which the true class matches the most probable class. This metric was chosen as our goal was to see how well our models could correctly generalize to unseen images. The final test results are as follows: The best model is the Modified ResNext, with 2 fully-connected layers with Dropout before the final layer, with a maximum test accuracy of 95.7%.

A confusion matrix was also generated (see Figure 10) to understand the vulnerabilities of the model. Further comments will be made regarding them in the Section 8.



(a) Model plots of test accuracy. **(b)** Magnified Figure 9a for top 3 models. **(c)** Test loss versus epoch for all models.

Figure 9: Plots of Top-1 test accuracy and loss for different models. Note that accuracy in Figure 9b is not exact, and is slightly smoothed to allow for visual ease.

Table 1: Statistics and best hyperparameters for all models, ranked from best to worst maximum test accuracy.

Models	Maximum Test Accuracy	Conv. Layers	Trained FC Parameters (M)	Total Parameters (M)	Optimizer	Learning Rate	Batch Size	Epochs
Modified ResNext	95.7%	101	4.25	91.1	Adam	0.0005	12	150
Original ResNext	94.8%	101	0.0533	86.8	Adam	0.001	16	150
VGG16	94.0%	13	119	134	Adam	0.001	16	150
ResNet-34	83.3%	33	0.013	21.3	SGD	0.001	16	150
Baseline	60.4%	2	1.61	1.61	SGD	0.01	16	150

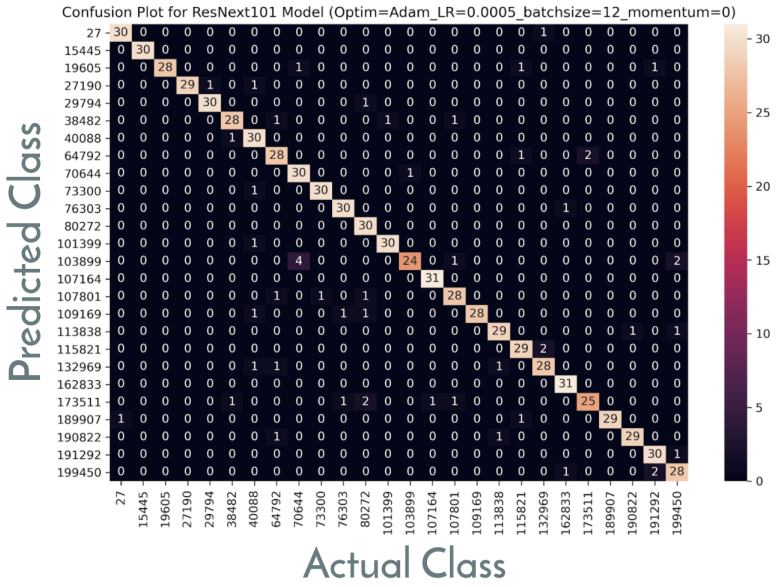


Figure 10: Confusion matrix for ResNext101.

8 Qualitative Results

Two notable vulnerabilities that emerge are the 4 incorrect predictions of class ID 103899 (the Hurricane Ridge mountain) instead of the actual class ID 70644 (the Sonnenstein castle) and 6 incorrect predictions of class ID 173511 (the Purana Qila fort) [8].

These vulnerabilities were determined by considering the most incorrectly predicted classes. Out of around 31 predicted test examples for each class, it can be seen that class ID 103899 and 173511 are the two most incorrectly predicted classes. Upon examination of their incorrect classifications, the following Figure 11 emerges.

(a) Incorrect classification of class ID 103899 (left) for actual class ID 70644 (right).

(b) Incorrect classification of class ID 173511 (left) for class ID 80272 (The Marmashen Monastery) and 107801 (The Red Fort).

Figure 11: Visual illustration of notable incorrect classifications for the model.

In Figure 11a, the common feature between both images is the trees. Therefore, it can be concluded that the model struggles at classifying correctly test images for different classes when there are common features (like trees) between them.

In Figure 11b, the model incorrectly predicts class ID 173511 for other classes. A possible reason is how all 3 images (each corresponding to a different landmark) share the common feature of circular brown stone towers. Due to their visual similarity in terms of structure, the model incorrectly predicts these classes.

9 Discussion and Learnings

Considering 26 output classes, we believe that our final ResNeXt-101 model performed very well considering the achieved test accuracy. This proves that our model was able to generalize very well on data it has never seen before. Our results are further validated as randomly guessing predictions on our problem would only yield an accuracy of 1/26, or 3.84%.

From academia, we had preconceived notions that our ResNet-34 model would perform better than VGG16 as ResNet-34 did not only have more convolutional layers, but the advent of residual connections seemed to be a breakthrough in boosting performance of modern CNNs [5] [9]. To our surprise, our VGG16 model performed much better than ResNet-34 and almost on-par with our ResNeXt models. On closer analysis, we attributed the unanticipated success of VGG16 to its large fully-connected layers (Figure 4), which contributed to most of VGG16's parameters. This is in contrast to ResNet-34's single fully-connected layer (Figure 5), which in our context was only 26 neurons. Ultimately, this finding made sense as we only trained the fully-connected layers for both networks, thus the large number of variable parameters in VGG16 allowed it to learn much more complex relationships that a single classifier layer in ResNet-34 simply did not have the capacity to.

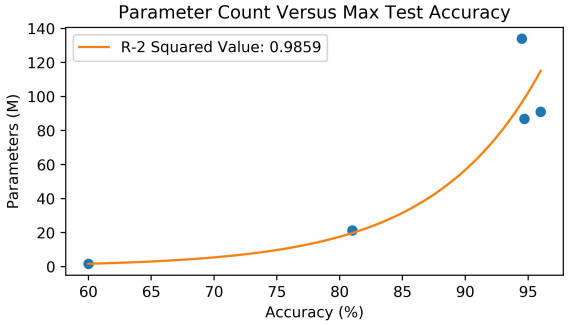


Figure 12: Plot of max test accuracy versus total parameter count for models in Table 1 with an exponential fit

By further analyzing the relationship between parameter count and test accuracy of the models we experimented with (data from Table 1), we found an interesting exponential relationship (as seen in Figure 12). While adding more parameters did help to increase generalization performance, there are clear diminishing returns as an improvement of 20% test accuracy from the baseline took a 20x increase in parameter count while a 35% increase required approximately 90x. Understanding this relationship will be useful in analyzing the trade-off between computational requirements and desired performance of neural networks when required in future projects.

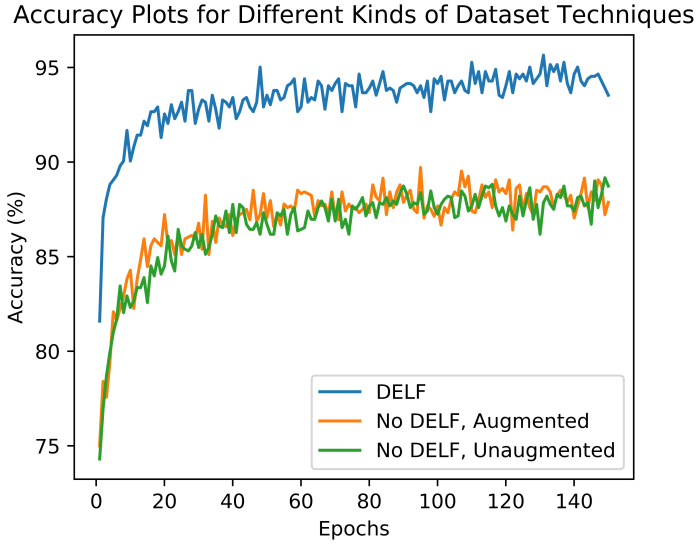


Figure 13: Effect of DELF and Augmentation on Overall Model Performance

Finally, we also attribute our success to our implementation of the DELF pipeline to filter out irrelevant images from our dataset, which taught us the importance of spending time with our data to ensure it is of optimal quality for our applications. Without DELF, we could visually see for certain classes that some validation/test photos were completely uncorrelated with any training photos (Figure 2). This made sense given the worse validation accuracy (Figure 13) without DELF as those types of photos were simply too anomalous for our network to reliable classify.

10 Ethical Framework

The primary stakeholders involved with respect to Monumentum are individuals seeking to gain insight towards certain landmarks (typically tourists), tour guides or other services that help inform individuals curious of certain landmarks, and the environment.

10.1 Tourists, Tours, and Other Services

With the significant amount of data being available online, simply knowing the name of a landmark gives access to a lot more information. Although many individuals would still appreciate tours and services for learning about landmarks, Monumentum could potentially undermine their influence and necessity. On the other hand, Monumentum respects the **autonomous** decisions of people. People are free to make their own decisions and learn about landmarks in any way they see best fit, so Monumentum is not directly inflicting harm against these services, but simply providing tourists with an extra option and the means of **beneficence**.

10.2 The Environment

Currently Monumentum is relatively low scale, only totalling to around 7.7K images and does not require high amounts of energy consumption. There is a clear trade off between **beneficence** (purely designed for the purpose of helping others), and the resources and energy consumed as the dataset and scale of the project begins to increase. If the number of landmarks being trained on were to increase, which could very likely be the case with so many different landmarks across the world, it is essential to look for ways to lower energy consumption, potentially by researching ways to lower the parameter count.

References

[1] S. Jeon, *1st place solution to google landmark retrieval 2020*, 2020. arXiv: 2009.05132 [cs.CV].

[2] Waelkh, *Landmark2020-delf model amp; submission code*, Jul. 2020. [Online]. Available: https://www.kaggle.com/waelkh/landmark2020-delf-model-submission-code.

[3] A. Suresh, *Web scraping images from google*, Mar. 2020. [Online]. Available: https://medium.com/@wwwanandsuresh/web-scraping-images-from-google-9084545808a2.

[4] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2015. arXiv: 1409.1556 [cs.CV].

[5] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[6] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, *Aggregated residual transformations for deep neural networks*, 2017. arXiv: 1611.05431 [cs.CV].

[7] *Neural networks¶*. [Online]. Available: https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html.

[8] T. Weyand, A. Araujo, B. Cao, and J. Sim, *Google landmarks dataset v2 – a large-scale benchmark for instance-level recognition and retrieval*, 2020. arXiv: 2004.01804 [cs.CV].

[9] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, *Inception-v4, inception-resnet and the impact of residual connections on learning*, 2016. arXiv: 1602.07261 [cs.CV].

Note this page is not included in word count

Haoran Wang
Permission to post video: Yes.
Permission to post final report: Yes.
Permission to post source code: Yes

Arsh Kadakia
Permission to post video: Yes.
Permission to post final report: Yes.
Permission to post source code: Yes

Sepehr Hosseini
Permission to post video: Yes.
Permission to post final report: Yes.
Permission to post source code: Yes