Google Landmark Recognition and Retrieval Challenges

Maxence Dutreix, Nathan Hatch, Raghav Kuppan,
Pranav Shenoy Kasargod Pattanashetty, Anirudha Sundaresan

April 25, 2018

1 Project Summary

Identifying landmarks in photographs from around the world is an open research problem hindered by the availability of a worldwide dataset and the lack of annotations in the existing datasets. The Google Landmark Recognition challenge, hosted by Kaggle, came about as a result of these obstacles to landmark recognition research. The Google Landmarks dataset [Noh et al., 2017] is one of the largest global landmark datasets to date. It contains a much larger number of classes than ImageNet [Deng et al., 2009], and its distribution of classes is highly imbalanced. This makes the dataset challenging to work with. The goal of this project is to develop sufficient research methodologies that can predict landmark labels directly from images, and help people better understand and organize photo collections.

The challenge consists of two components: Recognition and Retrieval. For Recognition (i.e. classification), we use a recently introduced Deep Neural Network architecture called ResNet to classify a given query image and come up with a confidence score for that particular prediction. We employ transfer learning by pre-training our classifier on ImageNet, and we use data augmentation techniques to ameliorate the skewness in the dataset across classes. This achieves a classification accuracy of 62%.

For Retrieval, we reproduce a baseline image retrieval architecture called DELF [Noh et al., 2017] and implement our own indexing system to handle the computational challenges of high-dimensional nearest-neighbor search. We also exploit the characteristics of this particular dataset (namely, the tendency of landmarks to be large, salient objects) by filtering results according to a saliency score. Our implementation achieves a mean average precision (mAP) of 0.083, which puts us in 51st place in the Kaggle challenge.

2 Background

Deep learning has considerably improved the state of the art in image classification. Using convolutional neural network (CNN) architectures like ResNet [He et al., 2016], it is possible to achieve a top-5 classification accuracy of 96.53% on ImageNet [Deng et al., 2009]. However, there are extensions and variations of these classification problems that have not been entirely solved:

1. While large, ImageNet contains only 1000 classes. We would like computer vision systems to be able to handle the much wider variety of classes that occur in the natural world.

2. Deep learning still has trouble when the training datasets are very small. For instance, we would like a computer vision system to be able to recognize a new species of bird given only one or two training examples.

3. A related but less well-studied problem in computer vision is *retrieval*. Given an image and a database, we would like to retrieve the images from the database which are most similar to the query image.

In order to stimulate progress on these problems, Noh et al. [2017] recently released the "Google Landmarks" dataset. The dataset contains ∼1M images of ∼15K landmarks around the globe, with a very skewed distribution: some landmarks have thousands of examples, but many have just one. Associated with this dataset are two current challenges on Kaggle. For the Landmark Recognition Challenge, the goal is to identify the landmark in each query image (i.e. classification). For the Landmark Retrieval Challenge, the goal is to indicate which database images contain the same landmark as the query image (i.e. retrieval).

For this project, we competed in both challenges. Section 3 gives details on our approach and results for the Recognition Challenge. Section 4 does the same for the Retrieval Challenge.

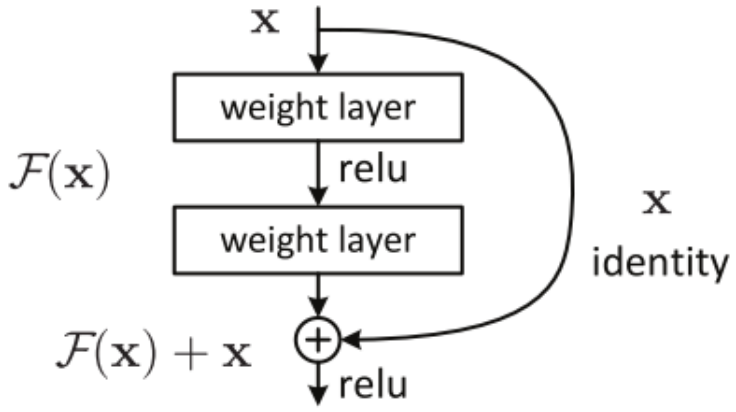
3 Recognition Challenge

Team Members : Raghav, Anirudha, Pranav

For Landmark Recognition, we adopted a Transfer Learning approach with Deep Residual Networks. Our dataset consists of 1,225,029 training images and 117,703 test images. A sizeable portion of the image dataset has links pointing to missing images.

3.1 ResNets

The deep residual network, proposed by He et al. [2016], was arguably the most groundbreaking work in the computer vision and deep learning community in the last few years. The performance of image classification, object detection and face recognition, have been boosted due to its powerful representational ability. ResNet allows us to train very deep networks and still achieve compelling performance. A major limitation of deep networks is the vanishing gradient problem – as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient diminishingly small. As a result, performance saturates. The core idea of a ResNet is to work around the vanishing gradient problem by using the "identity shortcut connection" (a residual block, as shown in the figure). Thus, a ResNet is similar to a feed-forward LSTM network without gates. The networks that we have used for this challenge are the 18-layer and the 50-layer ResNets.



A Residual block

3.2 Data Loading & Augmentation

Prior to dataloading, we augment the data so that the training is more robust. Data augmentation includes:

1. Extracting a 224x224 random crop of the 256x256 original images
2. Random horizontal flipping

For training the network, we split the training data into training and validation classes in a 1:2 ratio. Some of the classes have a single image, in which case a slight distortion has been introduced to the same image and copied into the validation set of the same class. PyTorch takes in data according to a dataloader class that specifies the order in which the data is to be read. This is also used to define the batch size, which also depends on the system configuration.'

3.3 Transfer Learning

We append a new layer of 14,951 neurons to the ResNet model for the classes in our dataset. There are broadly two ways to train this network on our dataset:

1. Freeze all layers of the network except the last layer and train on training data. Only the last layer parameters (which was randomly initialized) will change.
2. Train the entire network with the dataset using the pretrained model for initialization. Here, the parameters of all the layers will change.

The first option usually works well if the dataset used to pretrain the model is very similar to the dataset being worked on. However, training the entire network usually works better although it requires more computational power. For this problem we used the ResNet18 and the ResNet50 architectures trained with the ImageNet database.

We started with a ResNet18 model with weights frozen upto the penultimate layer. Due to its poor performance, we upgraded to a ResNet50 model which led to an increase in accuracy of over 100%. Unfreezing the weights led to a further increase in accuracy to about 62% which was the highest we could achieve.

We used a Stochastic Gradient Descent with Momentum approach for optimization. The batch size was calculated based on the memory of the system being used for the optimization. A Cross Entropy Loss was used, which takes into account the closeness of a prediction rather than just classification error. The training was done for 12 epochs on an NVIDIA GTX 970.

Model	Train Accuracy	Test Accuracy
ResNet18	26%	25.3%
ResNet50 (frozen weights)	52%	48%
ResNet50 (initialized weights)	63%	61.5%

4 Retrieval Challenge

Team Members : Nathan, Maxence

The goal of image retrieval is to match images from a database with query images provided by a user. In the case of the Google Landmarks Retrieval Challenge, the criterion for a database match is that contains a landmark in common with the query image. What is challenging about the landmarks dataset is its sheer size. It is not easy to fit one million images in memory, let alone retrieve matches from among them rapidly enough to maintain user satisfaction. However, the fact that we are dealing with three-dimensional physical landmarks allows us to improve our results by imposing geometric structure on the problem.

Prior work. As a baseline for the Retrieval Challenge, we implemented Deep Local Features (DELF) [Noh et al., 2017]. This algorithm extracts features for the database images using a fine-tuned CNN, reduces their dimensionality using principal component analysis (PCA), and embeds the result in an indexing system for approximate nearest-neighbors search. Given a query image, database images can then be ranked based on how many of their features are neighbors of the query image's features. All of these steps contain tricks to make the giant dataset more tractable:

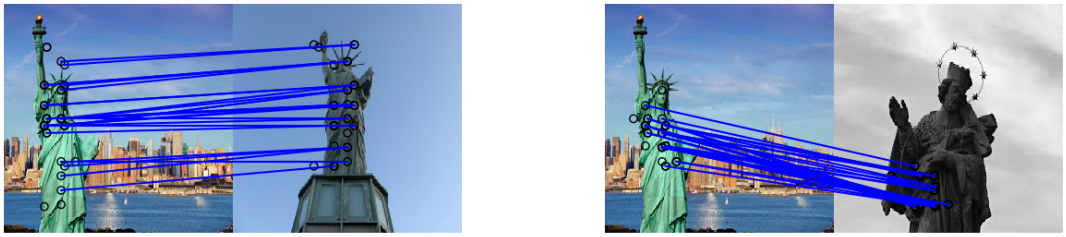
- The CNN produces thousands of features for each index image, most of which are irrelevant to the future task of retrieval. To address this, DELF includes an attention network which is trained to predict the *relevance* of each feature in a landmark classification task. A feature is retained only if its attention score exceeds a threshold value. After filtering, the index contains only ~ 200 million features.

- The CNN produces 1024-dimensional features. This is too many dimensions to work with in an indexing system, so they are reduced to 40 dimensions using PCA.

- Given a query image, the indexing system must search for the nearest neighbors of each feature of that image. In order to make 40-dimensional nearest-neighbors search tractable, DELF uses a combination of product quantization (PQ) [Jegou et al., 2011] and KDTrees.

Perhaps because of all of these shortcuts, the results contain many spurious matches. To counteract this problem, DELF exploits the structure of the dataset by using a *geometric verification* step. For each image feature, DELF retains the 2-D coordinates of that

Figure 1: Retrieval results from our implementation of DELF. Query image is on the left of each example. Circles represent features that had a match using nearest-neighbors search. Blue lines represent the subset of those feature pairs that still had a match after fitting to an affine transformation. **Left:** A good match. **Right:** A spurious match.



feature in the original image frame. Then, given a proposed matching between the index and query features, DELF attempts to find an affine transformation on the feature coordinates that achieves this matching. Any feature matches whose transformation error exceeds a threshold are considered outliers and ignored when scoring the index image. See Figure 1 for a visualization of this matching process.

Our modifications. The authors of DELF provided pre-trained networks and associated code for feature extraction and attention filtering. They also released the code for their geometric verification step. However, they did not release their indexing code. Hence, we had to come up with our own way of building an index of ~ 200 million features and performing nearest-neighbor search within it.

We decided to use KDTrees, which are a good data structure for efficient nearest-neighbors search.[1] The time complexity of this search is exponential in the dimension of the data, which in our case was 40. This was too big, so we modified the PCA step to further reduce the feature dimensions to 10.[2][3] With this modification, generating results for a query image takes less than one second. This means we can generate a complete submission file for the Kaggle challenge (100,000 query images) in about 24 hours.

Results. The Kaggle challenge is scored based on mean average precision (mAP). Our results are summarized in the table below. "PCA-10" refers to the approach described above. "PCA-10 (tuned)" refers to our second submission, wherein we tuned the hyperparameters of the nearest neighbor search (maximum distance, and number of neighbors to find) to better match the lower dimensionality of the search space.

[1] This part was mostly **Nathan's** work.

[2] Another reason for reducing PCA dimensions was to decrease the database size. In order to fit 200 million features into 32GB of RAM, each feature needs to be less than 160 bytes, even without taking into account the rest of the pipeline.

[3] DELF circumvented this problem by using product quantization (PQ). However, given the time constraints of this class project, we were not able to understand PQ well enough to implement it ourselves.

Method	mAP
Random	0.000
PCA-10	0.068
PCA-10 (tuned)	0.083

For comparison, the current leaders for the Retrieval Challenge have a mAP of 0.556, and our submission is in 51st place.

Future work. Based on qualitative analysis of our retrieval results, we have a few ideas for potential modifications.[4] Despite geometric verification, our results are cluttered by a large number of spurious matches. For example, in Figure 1 (right), there are a large number of matches in a relatively small region of the query image. With this insight, we decided to implement an additional threshold during verification based on *match region saliency*. We quantify saliency as the area of the convex hull of the match region, and we reject images whose saliency score falls below a threshold value. Calculating the convex hull is relatively cheap computationally, which preserves the tractability of our algorithm. Furthermore, saliency is intuitively a reasonable metric for this dataset, since landmarks are by definition salient objects. We have not had time to generate a full submission file with this approach, but preliminary results on the Statue of Liberty query image in Figure 1 showed an improvement in average precision from ~ 0.2 to ~ 0.5.

5 Conclusion and Future Work

The Recognition Challenge could have been tackled in many different ways. Using a transfer learning approach with Deep Residual Networks, we were able to achieve an accuracy of around 62%. Data augmentation was reasonably effective although storing the extra images required a lot more memory. Given the size of the dataset and the limited processing power available, this was a challenging constraint. We learned through several crashes and memory leakages the difficulties in playing with large datasets. Hyperparameter tuning, as discussed in class (and as it turned out in practice), is more "art" than science. The way we did it was to plug parameters at random and observe a few epochs of training. Ideally, we would have liked to have done a grid search, or even cross validation.

One future possibility is to use an R-CNN (Region based CNN), an architecture that has been very successful in object detection applications and could have helped alleviate the skewness of the dataset. This might provide better results than the ResNet.

The Retrieval Challenge is also an open problem. Our approach used a CNN to extract features, PCA to reduce their dimension, a KDTree for efficient nearest-neighbors search, and geometric verification to filter out spurious matches. As with many machine learning applications, large datasets are both a blessing and a curse. Hopefully the eventual winners will publish their approach for the benefit of the research community.

[4]This part was mostly **Maxence's** work.

References

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33 (1):117–128, 2011.

Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3456–3465, 2017.