# Landmark Recognition Challenge with ResNet and DELF

Ilio Di Pietro, Mauro Martini
Politecnico di Torino
Corso Duca degli Abruzzi 24, Torino (TO)
s{266393, 260771}@studenti.polito.it

## Abstract

*In this project we propose a complete solution for the Google Landmark Recognition Challenge published by Kaggle in both 2018 and 2019 [2].*
*Landmark Recognition is a large-scale classification task which is raising the interest of machine learners from all over the world for its peculiar challenging aspects. A huge amount of landmarks have been captured and gathered in the Google Landmark Dataset. This extreme classification scenario is combined with the necessity to confirm the correctness of the obtained prediction, since the test set contains 'tricky' samples. To do so, a retrieval mechanism based on Deep Local Features is implemented. Our solution provides a full pipeline composed by three main stages. A pre-processing and filtering is the first necessary step to deal with constraints imposed by both time and computational resources available. The classification phase has been performed with a ResNet50 model exploiting transfer learning from ImageNet dataset. Finally, DELF module has been adapted to our specific application with a threshold-based decision system for an efficient verification of the predictions. Our solution is able to reach a Global Average Precision score of about 0.85, which is a good result that on the other hand should be analysed under a critical perspective to detect general weaknesses.*

## 1. Introduction

Large-scale image classification is one of the areas that strongly pushes machine learning and computer vision research towards competitive improvements. Famous challenges, such as ImageNet Large Scale Visual Recognition Challenge (ILSVRC) have brought machine learners to develop deep models which have dramatically increased the number of potential applications in recent years.
Landmark Recognition is a famous large-scale classification challenge that plenty of teams and experts have faced from 2018 since the first Google Landmark challenge has been published on Kaggle [2]. Research about landmark recognition needs to extend actual annotated dataset to allow machine learning to provide a set of useful application in the management of picture collections or for example to optimize a real-time landmark recognition on mobile devices with camera such as smartphones. Google Landmark dataset is a very precious collection of training examples, obtained along several years through manual annotation using GPS locations of landmarks and images comparison. One of the main goal of the challenge is therefore to extend it through predictions provided by classification models that must be verified with an innovative technique. To do so, an image retrieval model needs to be implemented, receiving the predictions as input and checking them. Deep Local Features model [1], for which the astonishing performances are described in the paper suggested by the organizers of the challenge, has been chosen for the verification purpose.

### 1.1. The Google Landmark Dataset

Google Landmark Dataset present a huge amount of images and categories. To have an idea, with respect to ImageNet dataset used in ILSVRC, in which there were more or less 1k classes, here we are dealing with a number of classes that is 15 times higher and a corresponding number of images equal to 1,2 Million.
Some key aspects of this dataset have to be taken in consideration:

- It contains well-centered and high quality images as well as images captured by tourists, which can contain rotated or partial landmarks or even no landmark, the so called *distractors*;
- For some of the classes, a very few training samples are provided (4-5);
- Test dataset is not annotated, no labels are provided.

More in particular, having strongly unbalanced classes means that for some of them there is not a suitable number of good training samples, rising consistently the difficulty in the learning process and classification. In addition,

images are not provided in an already organized directory, but within a csv file with a list of URL, in order to open and download the JPG file. The combination of these factors, makes the classification much harder and the pre-processing one of the crucial and most challenging section of the whole project.

((a)) *Sample database images [7]*


((b)) *Sample query images*

Figure 1: Google Landmark dataset, examples of database and test images

### 1.2. Project motivations and goals

This project aims to experience an hypothetical participation to the Google Landmark Challenge dealing with stringent constraints imposed by development time and computational effort required in this extreme classification task, being for us the first time we face something of this size. However, we decided to give it a chance since it contains very stimulating aspects which are possible to explore regardless of the available GPU power or memory. Indeed it brought us working on a very realistic scenario, where the quality and the amount of available data cover a crucial role for the results and for the organization of the development and tuning process itself. Moreover the DELF model for image retrieval resulted to be a fascinating research work to study and its application and adaptation in our pipeline represents of the main source of our efforts.
Even though solutions comparable with the Leader-Board of the challenge requires for strong hardware usage and greater expertise in the field, in first place we aimed to give a demonstration that such problems are accessible also to beginners. Secondly, we wanted to propose a full classification/retrieval pipeline employing a ResNet architecture and a customized threshold system for the predictions verification to optimize the DELF behaviour.

### 1.3. Running environment and Python libraries

Even though we started looking for a more powerful platform at an affordable price, we ended up using Google Drive COlab, where the 15GB of RAM and GPU dedicated memory have been the hardest limit to extend our solution onto a wider dataset. By writing partial results on csv files, we succeed in optimizing the possibilities of the environment. We mainly exploit three libraries in our whole model, together with more common ones as numpy for scientific computations:

- Pandas: to handle multi-field data structure, especially in the first pre-processing stage and in the final computation of performances;
- PyTorch: for the whole classification stage of the pipeline, from the dataset creation from directories to the training process of the net;
- Tensorflow: to use the imported code for DELF model.

### 1.4. Related works

The starting points for our project were the various studies carried out by the other participants of the challenge. In particular we focused on two works [3], [4] in which the proposed solutions are based on a classification part with a VGG16 pre-trained on ImageNet dataset due to its good features extraction properties. No final results in terms of GAP are provided at the end of their reports. We know instead that one of the team used 50GB of RAM, a good quality GPU, 8CPU and at least 500GB to work with 2k classes out of 15k [3].

## 2. Outline of proposed solution

### 2.1. Data pre-processing and class filtering

The huge amount of data present in the original dataset makes the pre-processing part necessary to define the quantity and the quality of the samples passed to the network.
Due to the limited hardware capacity of COlab, the number of classes has been filtered to a maximum feasible of 300.

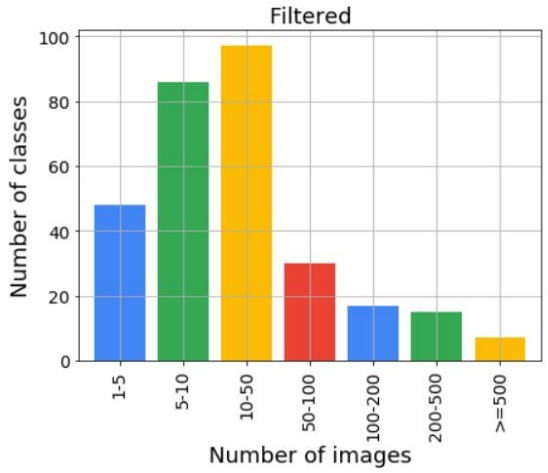


Figure 2: Distributions of number of classes versus number of images per class.

Landmarks between 1000 and 1300 were chosen. Distributions of data before and after filtering shows that, despite the reduction performed, they approximately kept the same distributions in terms of number of classes versus amount of images per class. The final distribution is shown in Figure[2]
A massive data cleaning was performed, taking only the not corrupted URLs and resizing the downloaded images at the resolution 128x128 to save memory and time. As next step, we have split our data in train, validation and test set. At the beginning a static percentage of training samples has been taken in consideration for the creation of the test set, but for classes with less than 10 images no samples was put in the test set. For this reason, in order to keep alive the difficulty of the challenge, a variable percentage has been adopted: if a class had less than 10 images 25% images was taken; 5% if the images were between 10 and 100 and 1% if they were more than 100. At the end of this step, the data distribution obtained is reported in Table [1].

| Training | Validation | Test |
|---|---|---|
| 12 245 | 2906 | 271 |

Table 1: Training, validation and test dataset size after class filtering and data clearing

Subsequently data were organized in directories to be able to use PyTorch dataset functionalities. Data Augmentation was eventually performed on the training set defining transformations of random crop and rotation ($\pm 20°$), applied only at training time by the dataloader without storing new images in memory.

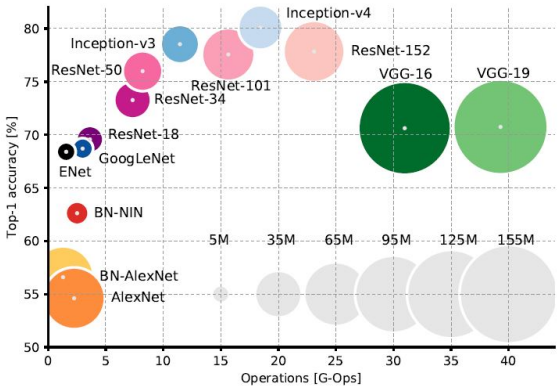


Figure 3: Top-1 accuracy versus number of operations in a standard forward pass, for ImageNet CNN models. Size of the blob proportional to number of parameters in the net [5].

### 2.2. Classification model and predictions

A trade-off between computational limits and accuracy is the key behind our choice for the classification model. As reported in section 1.4, most of other participants went for a VGG16 layout for the net, PyTorch offers a good set of predefined famous CNN models, hence we have chosen one of them as starting point. Authors of [5] have performed a comparison in terms of performance, memory usage, and number of operations for all the net architecture proposed in the ImageNet challenge along the years. In particular, for our necessities, two plots showing memory usage and accuracy versus number of operations performed during the forward pass, have been sources of precious information. Figure [3] shows the first one.

According to these useful trends, we decided to take in consideration a ResNet50 architecture, since it allows for a lower number of operations and a higher accuracy with respect to VGG16. Deep residual networks ensemble won the first price in ILSVRC 2015 classification competition with a 3.57 % error [6].
Their peculiar characteristic is that they are able to 'skip connection', or, in other words, layers outputs are feed-forwarded with an additional path jumping over some other layers. This aims to solve the problem of vanishing gradients. Indeed in CNN exploiting gradient-based learning methods and backpropagation, when multiple layers are used together with sigmoid activation functions, the gradient decreases exponentially, becoming very small for the front layers. ReLU activation function and ResNet architecture are good countermeasures for this issue.

Figure [4] shows the comparison between a plain CNN block and a typical residual connection building block. The output is computed according to $y = F(x, W_i) + x$, where $F = W_2\sigma(W_i x)$ with two layers, and sigma is the ReLU

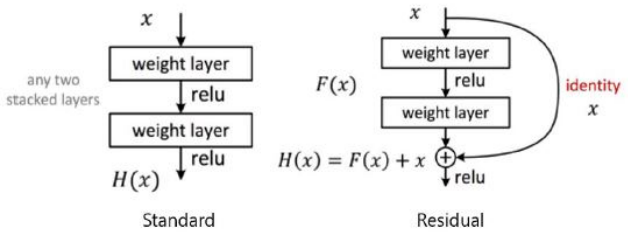


Figure 4: On the left: Plan and residual block comparison [6].

| Hyperparameter | Value |
|---|---|
| Learning Rate | 0,01 |
| Batch size | 128 |
| Epochs | 10 |
| Step size | 8 |
| Gamma | 0,1 |

Table 2: Optimal set of hyperparameters

| Image_id | Landmark_id | Probability |
|---|---|---|
| 1eee0f41da6ea346 | 1005 | 0,999 |
| 5f516881e61501c4 | 1054 | 0,478 |
| 912122e086d3e77d | 1031 | 0,608 |

Table 3: Example of predictions on the test set

activation function [6].

Our approach is based on fine-tuning of a pre-trained ResNet50 model, importing from PyTorch library weights for the convolutional layers already trained on ImageNet. The hyperparameters are tuned exploiting the annotated validation set derived from the original training dataset. Learning rate, batch size, epochs and step size are the ones considered in the optimization process. Final set of optimal hyperparameters has been reported in Table[2]. SGD has been chosen as optimizer, whilst a cross entropy criterion has been chosen to compute the loss. Transfer learning, in combination with 10 additional epochs of training on our Landmark dataset, has given great results in terms of accuracy and loss. As shown in Figure[5], after few epochs we have reached a plateau at 97% of accuracy on validation set (labeled) compared to a 99% on training.

Finally, the unlabeled test set has been passed to the model and a list of predictions in the structured format 'image_id, predicted landmark_id, probability' has been produced and written on a csv file. Some examples are reported in Table[3].

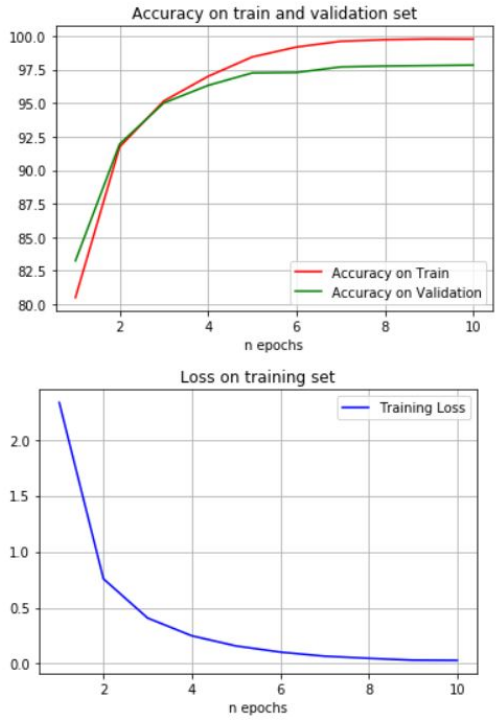


Figure 5: Accuracy and Loss over 10 epochs with optimal set of hyperparameters.

### 2.3. Retrieval model with DELF

Landmark Challenge contains a crucial problem to face: query images to classify are real images captured by tourists and they could be also *distractors*, i.e. shots which do not contain landmarks. CNN are not able to distinguish such test samples autonomously and they assign a label in any case. For this reason we need to filter those cases from the final score computation. A deep local features extractor (DELF), presented in the paper related to the Kaggle challenge, has been implemented for such a purpose, exploiting a retrieval mechanism. DELF pipeline is described in Figure[6] from the original paper [1]. For each processed query image, the predictions of the net must be checked. Images belonging to the predicted class are retrieved from the training dataset. When features are extracted, related information are stored in the form of *descriptors* and *locations*. Most significant features are selected by means of an attention-score mechanism and subsequently applying PCA algorithm to consider only the 40 principal dimensions. At this point, retrieval can start. It is performed selecting the K nearest neighbours of local descriptors from the query image with K = 10, among descriptors of database images previously organized on a KDtree. Finally, a system of aggregated index is exploited to connect each database image with its set of descriptors and locations to use for the

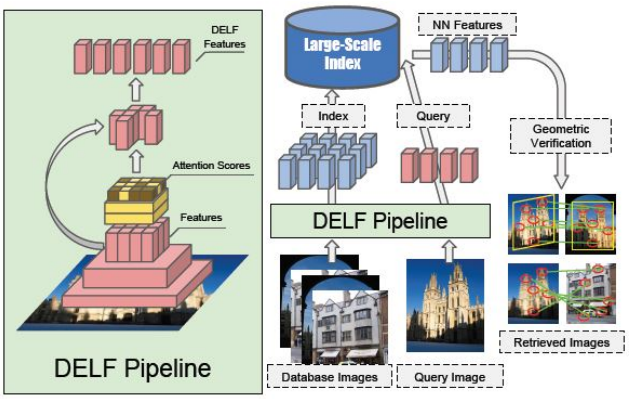


Figure 6: DELF pipeline scheme [1].

match. What practically compare query and database images is RANSAC [9], which is responsible of the geometric match of descriptors and locations. The results of the match is the number of inliers, which is finally employed to decide the correctness of the prediction [1].
Our contribute in such sense deals with the design of a suitable threshold system to detect a correct behaviour in both the classification and the retrieval phases, on the base of the number of inliers matched. A correct prediction is confirmed whenever 25 or more inliers are matched on a single image of the predicted class, or alternatively when their average value is greater than 8. If less than 5 inliers are found, the query is interpreted as a distractor with no landmark. For values between 5 and 8 the prediction of the net is considered wrong. This set of numerical thresholds have been tuned after a careful analysis of the number of inliers given by DELF in all the specific situations provided by the Landmark Dataset, as described in detail in section 3.

## 3. Final Evaluation

The full pipeline built for landmark recognition, composed of classification, retrieval and verification is represented schematically in Figure[8].

### 3.1. Quantitative results

An evaluation metric for such a complex total model should take in consideration multiple factors to measure the quality of the classification, but on the other hand must be considered that DELF model has a proper accuracy and limited efficiency too. The official metric used during the challenge to evaluate a submission is the Global Average Precision (GAP), which is computed as explained below[2]:

$$GAP = \frac{1}{M}\sum_{i=1}^{N} P(i)rel(i)$$

in which:

- M is the total number of queries with at least one visible landmark;
- N is the total number of predictions;
- $P(i)$ is the precision at rank i;
- $rel(i)$: 1 if the prediction is correct, 0 otherwise.

Hence, while computing GAP, predictions are considered with a ranking based on their probability at prediction time. Precision is therefore computed as the ratio between true positives and total positives. True positive occur when the prediction made by the classifier is correct and DELF is able to verify it putting $rel(i) = 1$ . On the other hand, false positive occur when the model incorrectly label a sample, and DELF is able to verify it putting $rel(i) = 0$.

$$P(i) = \frac{tp(i)}{tp(i) + fp(i)}$$

Thanks to DELF we are able to detect the distractors, that should be filtered from the predictions in the computation of the GAP, considering only M at denominator. In addition, this metric allows us to consider the predictions with a lower probability weighted with a precision factor dependent on all the previous analysed cases; whenever the net is not sure of its classification it gives a low confidence score, hence if the label has been assigned not correctly and the DELF has not been able to detect the mismatch, this singular case will have a lowered influence on the GAP value. In such a way, we can take in account also the effect of errors coming from the retrieval stage.

Our solution is able to reach as final output of both classification and retrieval stage a GAP of 85.59%.

However, this result needs to be interpreted critically. First of all it is referred on a very limited number of classes over the entire Landmark dataset. Secondly, in the qualitative results section it will appear clearer that neither GAP is 100% reliable and robust as a metric for such complex scenario. This is due to the fact that in order to check predictions with no error a ground truth is needed, whilst in our case DELF has a proper efficiency as well. Furthermore, the dataset contains poor quality sample images which act as disturbs on the accuracy evaluation of the model.

### 3.2. Qualitative results

A meticulous detection of errors in the DELF output results has been performed by looking at the retrieved database images associated to each query and the features matched. It is important to remember that the two main goals of our solution are:

Figure 7: Visualization of feature correspondences: on the left good matches between query and database images; in the center wrong classification detected by DELF; on the right DELF mismatches due to ambiguous geometry or different part of landmark captured.

- Obtain predictions as correct as possible from the classification model;
- Retrieve and check efficiently the predictions.

From a first look at images correspondences it is possible to state that the net is predicting correctly the majority of the landmarks, and the retrieval system is able to find the associated database images. As shown in Figure[7], we can expect to see several cases for what deals with the number of inliers matched by RANSAC. In particular, we have found out that DELF works efficiently on images where landmarks are well centered in the picture, whilst a wrong DELF behaviour occurs in a couple of specific scenarios. First, for specific landmark involving variable or ambiguous geometric characteristics, such as landscape at the seaside (landmark 1046), RANSAC is not able to match features and it assigns a wrong prediction even though the net classified it fine. On the other hand, the net most probably makes error on prediction when landmarks have similar shapes and features, and a good match can occur at verification with DELF. This problem is practically solved thanks to the detailed threshold system using the number of inliers in most of the cases. Last but not least, as shown in some example in Figure[7] , when pictures contains different parts of the same landmark both the classifier and the geometric verification have problems, and a customized pre-processing on such landmarks is the only way to approach the situation. We applied data augmentation using rotation and crop only in a general fashion on the whole training dataset.

## 4. Conclusions

With this project we have developed a complete solution for the Google Landmark Challenge with limited resources and we have been able to obtain good value in terms of classification reducing the original dataset. Next natural step should be try it out on a more powerful GPU or platform increasing the number of classes. For sure it would rise the difficulty in learning and classification and a more detailed data augmentation focused on landmark with few training samples would be taken as countermeasure, before considering the employment of a more expensive classifier. We have enabled an optimized usage of the DELF model through a proper tuning of the thresholds for the number of inliers matched by RANSAC. Moreover, as future work we propose to use a customized model to extract features in the DELF in order to improve the efficiency in the retrieval stage. In this new scenario it would be possible to further train the features extractor applying a landmark-specific data augmentation. Last but not least, an interesting study could be related to a new formulation of GAP metric, which possibly takes in account more in depth the multiple stages of the full-pipeline, considering errors coming from both the classifier and the retrieval model.

## References

[1] Noh, Hyeonwoo, et al. *Large-scale image retrieval with attentive deep local features.* Proceedings of the IEEE International Conference on Computer Vision. 2017.

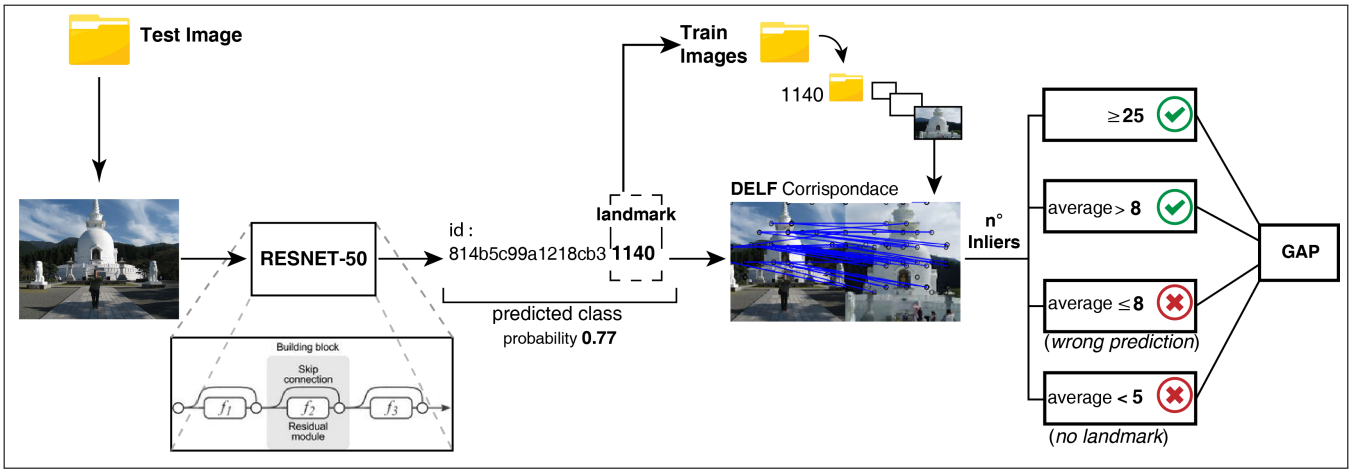


Figure 8: Overall scheme of the presented solution: classification with ResNet50 and retrieval with DELF. Number of inliers are used to check predictions with a threshold system.


[2] Kaggle website, *Google Landmark Recognition Challenge*, official page. https://www.kaggle.com/c/landmark-recognition-challenge/overview. 2018.

[3] Catherine McNabb, Anuraag Mohile, Avani Sharma, Evan David, Anisha Garg, *Google Landmark Recognition using Transfer Learning*. https://towardsdatascience.com/google-landmark-recognition-using-transfer-learning-dde35cc760e1, 2018.

[4] Kevin Widholm, *Landmark Recognition Challenge - Development and Experiences*. https://www.novatec-gmbh.de/en/blog/landmark-recognition/. 2018.

[5] CANZIANI, Alfredo; PASZKE, Adam; CULURCIELLO, Eugenio. *An analysis of deep neural network models for practical applications*. arXiv preprint arXiv:1605.07678, 2016;

[6] HE, Kaiming, et al. *Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. p. 770-778.

[7] André Araujo and Tobias Weyand, Software Engineers, Google Research *Google-Landmarks: A New Dataset and Challenge for Landmark Recognition*. https://ai.googleblog.com/2018/03/google-landmarks-new-dataset-and.html, 2018.

[8] M. Fischler and R. Bolles. *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*. Communications of the ACM, 24(6):381–395, 1981. 5.

[9] F. Perronnin, Y. Liu, and J.-M. Renders, *A Family of Contextual Measures of Similarity between Distributions with Application to Image Retrieval*, Proc. CVPR'09