Solution to Google Landmark Recognition 2019 Challenge

Iris Landerer, Thi Hoai Phuong Nguyen, Malte Sönnichsen

Computer Vision for Human-Computer Interaction Lab
Karlsruhe Institute of Technology
Karlsruhe, Germany
[iris.landerer, thi.nguyen3, malte.soennichsen]@student.kit.edu

Abstract

In the practical course at the Computer Vision for Human-Computer Interaction Lab at Karlsruhe Institute of Technology (KIT), we participated in the Google Landmark Recognition 2019 hosted by Kaggle. This challenge is about an image classification task with a huge dataset and number of classes. The main problems are the extreme imbalance of classes and the unrelated images within a class. In this paper, we use ResNet50 pre-trained on Places365-Standard dataset as a backbone. Furthermore, we clean our dataset to receive beneficial input for our Triplet Loss function. Finally, we got a Global Average Precision score of 1,348% and thus the 100th place at the challenge.

1. Introduction

Image classification technology has shown remarkable improvement over the past few years. As an example, the Imagenet classification challenge can be seen, where error rates decrease continuously every year. Researchers are more and more focusing on fine-grained and instance-level recognition problems to bring state of the art in computer vision to a new level. Instead of recognizing entities in general such as buildings, mountains and monuments, machine learning algorithms are trained to identify the Eiffel Tower, Mount Blanc, or the Statue of Liberty. However, the absence of a sufficiently large dataset with corresponding annotations in this area has implied a significant problem for researchers. For this reason, Google released its landmark recognition dataset last year, namely Google-Landmarks-v1 [1]. Google-Landmarks was the largest worldwide dataset of its kind available in 2018. Combined with the release of this dataset, Google additionally hosted two Kaggle challenges, Landmark Recognition 2018 and Landmark Retrieval 2018, to foster advancements in these research fields [1][2].

Figure 1: *Challenges of the Google-Landmarks-v2 dataset: covered landmarks (a and b), same buildings from inside (c) and outside (d), no landmarks at all (e and f)*

As stated above, recognition is concerned with recognizing specific instances of entities, e.g. distinguishing Golden Gate Bridge from just any other bridge. In contrast, retrieval deals with matching a particular object in an input image to all other instances of that object in a gallery of reference images. Google achieved its goal, as more than 500 teams from all over the world participated in these challenges. However, both instance recognition and image retrieval methods require even more massive datasets

with an increased number of images and more variety of landmarks, so that better results can be reached and more robust systems can be created. To support this goal, Google released Google-Landmarks-v2 this year [3]. It is an entirely new, even larger landmark recognition dataset that includes over 5 million images which are twice as many images as the previous dataset contained. Furthermore, the new dataset consists of more than 200 thousand different landmarks, i.e., an increase of seven times more landmarks than in Google-Landmarks-v1. Due to the difference in scale between the two datasets, this dataset is much more diverse and represents a significant challenge for state-of-the-art instance recognition algorithms. Based on this new dataset, Google introduced two new challenges on Kaggle Landmark Recognition 2019 and Landmark Retrieval 2019 [3][4].

In this paper, the dataset and its challenges are described in chapter 2. In section 3, we explain three different approaches belonging to the three best-ranked teams at Google Landmark Recognition 2019 challenge. Following, we present our various experiments and their corresponding results.

2. Dataset

In the following, we describe the dataset used for the challenge in detail, and also possible challenges of this dataset. Furthermore, the metric for evaluating the submission is explained.

2.1. Introduction of Google-Landmarks-v2

The Google-Landmarks-v2 dataset used for the Google Landmark Retrieval 2019 challenge and the Recognition 2019 challenge is the largest worldwide landmark recognition dataset currently available. The training set contains 4,132,914 images belonging to 203,094 classes, and the test dataset is of size 117,577. Only samples from the train set are labeled. Not only a large number of different classes, but also the distribution of these classes is a significant obstacle for the solution of the task. The classes are highly imbalanced.

On the one hand, most landmarks are only contained a few times in the training set, since they are probably not so famous. On the other hand, there are a few classes which occur a thousand times in the training set. These classes have to be very popular such as the Eiffel Tower or the Statue of Liberty. The imbalance of the dataset can be seen in 2.

The retrieval challenge asks to find an image of the same landmark instance from an index set, whereas the task of the recognition track is to answer the corresponding label defined in the train set. In comparison to the Google-Landmarks-v1 dataset, which was released after an automated data cleaning step, the Google-Landmarks-v2 dataset only contains raw data. The v2 dataset was constructed by mining web landmark images without the cleaning step. Therefore, each category may contain quite diverse samples and presents specific challenges to the participating teams. For example, many images do not show the landmark directly, but are partly hidden by people who took a photo with themselves and the landmark. Also, bad weather can contribute to the fact that a landmark is barely recognizable. Examples of such covered images are shown in 1a and 1b. Another challenge is that images within a class can look completely different. An example of it is shown in 1c (outside) and 1d (inside). This phenomenon comes from the inside and outside look of one object. A third major problem comprises the fact that the dataset also includes images with no landmarks in it at all, as shown in 1e and 1f. That makes the task harder, as these non-landmark images distract from the actual landmark recognition.

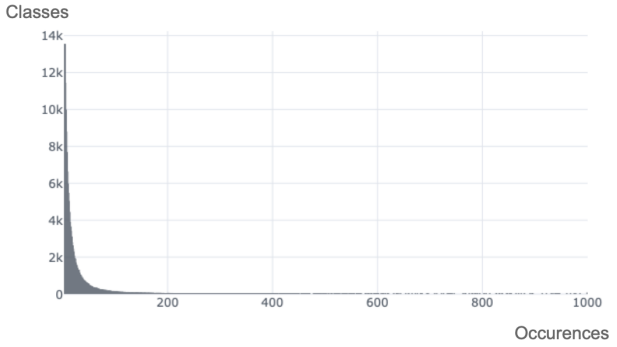

Figure 2: *Histogram showing the imbalance of the Google-Landmarks-v2 dataset. Some classes are very often contained in the dataset, but most of the classes occur just a few times.*

2.2. Evaluation metric

The evaluation metric for the Kaggle submission is the Global Average Precision (GAP), which measures both ranking performances as well as the ability to set a common threshold across different queries. It works by first predicting one landmark label and a corresponding confidence score for each query image. Each prediction (label/confidence pair) is treated as an individual data point in a list of predictions. The predictions are sorted in descending order by their confidence scores. Then, the Average Precision is calculated based on this list. The formula for the GAP is as follows:

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

where N is the total number of predictions returned by the system across all queries, M is the total number of queries with at least one landmark from the training set visible in it, $P(i)$ is the precision at rank i and $rel(i)$ denotes the relevance of prediction (1, if the i-th prediction is correct, and 0 otherwise).

3. Related Works

In the following we present the solutions of the three best ranked teams at Google Landmark Recognition Challenge 2019.

In [5] the winner team focuses on data cleaning and uses global, and local CNN approaches based on retrieval techniques for solving this challenge.

In the given Google-Landmarks-v2 dataset exists an extreme imbalance of classes. Therefore, all classes are removed which have no more than three samples. Furthermore, Gu and Li try to handle the inconsistency problem between the images within a class with matching pair approach. A matching pair is defined that two images have a cosine similarity above a threshold based on their descriptors extracted with ResNeXt101 (64x4d). Also, these descriptors used to calculate the cosine similarity of all images to each other. With a threshold of 0.5, the number of classes is decreased to 112,782 because lots of them have no matching pairs. At least, they set up new training data by selecting randomly at most 100 matching pairs for each class. On this cleaned data, they want to train a model which matches test images with train images to get a prediction instead of training a classifier. Image matching consists of two steps, global search (Step-1) and local search on global candidates (Step-2). The global search uses an ensemble approach for extracting image descriptors followed by l2-normalization. For this purpose, they use ResNet-101, ResNeXt-101 (644d), SE-ResNet-101, SE-ResNeXt-101 (324d) and SENet-154 as backbones. This global CNN model uses the sum of the contrastive loss and triplet loss as the loss function. In Step-1, all test images will be matches with all training images based on their extracted descriptors. Then the top-10 closest neighbors are held as global candidates for Step-2. Top-5 closest neighbors are used for prediction based on the highest accumulated class similarities. In Step-2, a pre-trained Detect-to-Retrieve (D2R) model is used as a local CNN model for matching local features. Each test image will be matched with its top-10 candidates from Step-1. Analog top-5 closest neighbors are used for prediction based on the highest accumulated inlier score produced from D2R model and representing the number of matching inliers. Lastly, re-ranking is applied for improving the performance by increasing the confidence score. From a list of predictions in descending order by the confidence score, the top-20000 predictions are chosen. They match the first ranked image with other images in top-20000 predictions using D2R models. Only images, that have an inlier score better than 24, are re-ranked below the first ranked image based on its inlier score. This re-ranking process is repeated for the top-100 images in the top-20000 predictions. With this whole pipeline, Gu and Li are awarded first place at the challenge with a GAP of 37.60%.

In [6] Chen *et al*. introduce their solution with which they reached the second place in the Landmark Recognition Challenge 2019 with a final GAP score of 35.98%. Their approach starts by classifying images with the k-nearest neighbor's algorithm. Global features of the images are extracted using a ResNet152 backbone with Npairs loss. After extracting descriptors, all test images are matched with all the images from the train set. Each test image is labeled by taking the most frequent label of the top-5 matched images.

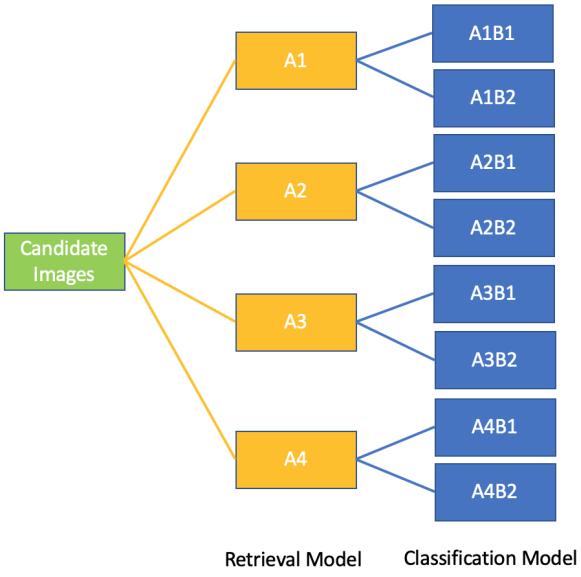

Figure 3: *Overview about different parts of the re-ranking of the second place solution:*
A1: number of top-5 matched image labels is not more than 2 and the minimum score of the predicted label is more than 0.9. A2: number of top-5 matched image labels is not more than 2 and maximum score of the predicted label is more than 0.85. A3: the image does not satisfy A1, A2 or A4. A4: the top-5 matched images labels are all different.
B1: the predicted label between the retrieval model and classification model are the same. B2: otherwise.

One problem of the dataset mentioned in 2.1 is the presence of a massive number of images containing no landmarks at all. For filtering these non-landmark images, a single object detector model based on a Faster R-CNN is trained on the Open Images V4 dataset [7]. This dataset

stands out as the largest existing dataset with object location annotations containing 14.6 million bounding boxes for 600 object classes on 1.74 million images. For distinguishing the non-landmark images, the 600 object classes are divided into three parts, namely landmark part, uncertain part, and non-landmark part. The landmark part includes the following five classes: building, tower, castle, sculpture, and skyscraper. On the contrary, the uncertain part comprises the seven classes: house, tree, palm tree, watercraft, aircraft, swimming pool, and fountain. All other classes are considered in the non-landmark part. An image will be categorized as a landmark image if at least one object in the image belongs to the landmark part. If there is one object in the image belonging to the non-landmark part, it will be a non-landmark image. As of the last step, re-ranking is used. For this reason, the landmark images are first ranked according to the top-5 matched images labels and scores. For that, they use a retrieval model which was also developed by Chen *et al*. during the parallel running Landmark Retrieval Challenge 2019. Then, the landmark images are divided into four parts and further each part into two subparts as can be seen in 3. Re-ranking is applied corresponding to these parts using a classification model based on ResNet152 and trained with about three million images from the train set only considering landmark images.

Awarded as third place, Ozaki and Yokoo describe in [8] their solution approach. Emphasizing the importance of data cleaning, an automated data cleaning system is built using spatial verification. For the first cleaning step k nearest neighbor is applied on the Google-Landmarks-v1 dataset getting for each image its neighbors. Secondly, spatial verification is performed on up to the 100 nearest neighbors of each image representation to reduce computational cost. The last step is responsible for checking if the count of verified images in the second step is greater than a certain threshold. In this case, the corresponding image is added to the cleaned dataset. Besides data cleaning, an ensemble of different descriptors is used to achieve better results. For this ensemble, six models are used containing FishNet-150, ResNet-101, and SE-ResNeXt-101 trained with cosine-softmax based losses. All models are pre-trained on ImageNet and Google-Landmarks-v1. Each model outputs a descriptor with 512 dimensions. Concatenating these descriptors results in a final descriptor with a dimension of 3072. For the recognition task itself, the following approach is applied: The first step is to find the top-k neighborhoods in the train set by the euclidean search. Secondly, the label of the given query is estimated by the majority vote of soft-voting based on the sum of cosine similarity between the neighboring training image and the query. The sum of cosine similarity is used as a confidence score. Finally, the influence of distractors is suppressed by a heuristic approach. Using this approach in combination with the spatial verification and the ensemble results in a GAP of 35.54%. For achieving this result, spatial verification and the ensemble are fundamentally crucial because, without them, the approach performed a lot worse having a GAP of only 20,79%.

4. Model

4.1. ResNet50

In our work, we use ResNet50 pre-trained on Places365-Standard dataset as backbone [9]. The Places365-Standard dataset contains 365 scene categories with at most 5000 images each. With approximately 1.8 million images in total, this dataset is smaller than the Google-Landmarks-v2, but still sufficient as a good representation. Besides, ResNet50 stands for Residual Network with 50 layers. This architecture supports residual learning, i.e. the networks tries to learn the subtraction of features learned in each layer instead of learning some features directly by using residual connections [10]. In the following, we use this pre-trained ResNet50 for almost all the experiments below.

4.2. EfficientNet

Mingxing Tan et. al. have observed that balancing network depth, width and resolution are important for efficiency [11]. Based on this insight, they designed a new neural network using neural architecture search and scaled it up, while maintaining the above mentioned balance to create a new family of efficient models. The last experiment additionally requires the backbone to have fewer parameters, while maintaining the same performance, in order to train as many experts as possible, which is why we used EfficientNet instead of ResNet50. Besides, we have not used EfficientNet in the previous experiments, because there is no pre-trained model on Places365 and the model is quite new.

5. Experiments

In this section, we explain our solutions for the Google Landmark Recognition 2019 challenge. First, we begin with the setup of the experiment, which is applied for all experiments in the following. Then for the experiments themselves, we try different loss functions and add some data pre-processing steps.

5.1. Experiments setup

First, we explain the pipeline used for all experiments. The main idea is matching test images with training images based on their feature descriptors to make a prediction. For training, we use ResNet50 mentioned in 4.1 to extract feature descriptors of training images. All images are resized

to 224x244 and are trained in batches with a size of 64. We also try different pre-processing steps and losses that lead to various experiments explained below. For the learning rate, we start with 1e-4 and use ADAM as an optimizer to improve performance. For the evaluation, ResNet50 is used again for extracting feature descriptors of test images. For each test image, we match it with all descriptors of training images to get a training image that has the shortest distance to the test image. From this found training image, we take its class label to categorize the test image and calculate a distance based confidence score.

5.2. Baselines

In this section, the baselines, which are later taken for comparison with the experiments we conducted, are introduced.

5.2.1 Random Baseline

Our first idea was to test the dataset's degree of difficulty with a simple random baseline. As outlined in 2.2, every prediction consists of a pair of the landmark label and its confidence score. For this baseline, we take a random label for the landmark and predict a confidence score based on a uniform distribution. Applied on the test set, we unfortunately achieved a GAP score of 0%.

5.2.2 Baseline on 1000 classes

Since our random baseline failed, we searched for another approach. One of the biggest challenges of this recognition task is the large dataset with its approximately 200.000 classes. For this reason, we attempt to limit the number of classes to the 1000 most frequent classes. As a model, we take the pre-trained ResNet50 as described in 4.1. Nonetheless, the GAP score is still 0%, which shows that this dataset is very challenging.

5.3. Experiment 1: Adaptive softmax

For our first experiment, we attempted an approach using adaptive softmax because it is much quicker than normal softmax. Adaptive softmax was initially introduced in the field of natural language processing for efficiently training a neural network based on language models over vast vocabularies. Since we have to handle a massive number of classes for the recognition task, we applied adaptive softmax to help to solve our task. In the following, a summary of adaptive softmax is given. Adaptive softmax is based on hierarchical softmax, which is a previous attempt to speed up softmax. The hierarchical softmax parts the softmax into two stages. First, the cluster *C(w)*, to which every word *w* is uniquely assigned, is predicted. Then softmax is applied to all the words for the specific cluster.

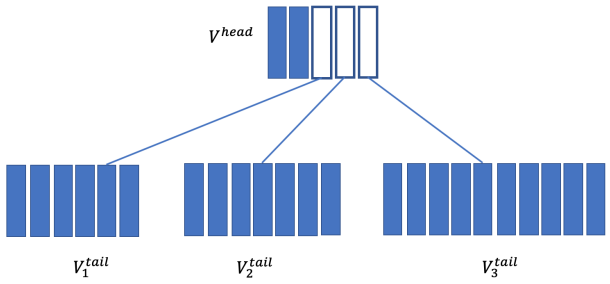

Figure 4: *Adaptive softmax with four clusters, namely one head and three tails. The head contains a small number of the most frequent words, while the tails contain larger numbers of less frequent words.*

However, as described by Grave *et al*., the hierarchical softmax is slower compared to adaptive softmax, as it does not work well on GPUs [12]. Their adaptive softmax approach is optimized for GPUs. It is based on Zipfs law that most of the words in a corpus are covered by only a small subset of the vocabulary. The adaptive softmax exploits this information by associating words in the vocabulary to clusters regarding the frequency of the words. For the approach, the dictionary has to be partitioned into *n + 1* clusters: $V = V^{head} \cup V_1^{tail} \cup ...V_n^{tail}, V_i \cap V_j \neq \emptyset$. A smaller number of the most frequent words in the vocabulary is assigned to V^{head}. The much larger set of rarely occurring words is distributed over the individual subsets of V^{tail}. An overview of this structure is given in 4. After partition, the first step is to compute a softmax over all the words in the head, plus *n* extra words corresponding to the tail clusters. Only if the word is not contained in the head, additional softmax has to be applied on the next tail clusters.
We used adaptive softmax to encode our labels. With the encoded labels, we used the ResNet50 model as described in 4.1 with cross-entropy loss. After a long time of training, this approach finally results in a GAP score of 1.239%.

5.4. Experiment 2: Triplet loss

One of the biggest challenges is to deal with a massive number of classes. Besides, we want to learn the similarity between images. Therefore we choose triplet loss as the loss function for our second experiment.

For training with triplet loss function, we feed the network a triple of images which present an anchor image, a positive and a negative sample. The positive sample represents an image which has the same label as the anchor. Moreover, another way around the negative sample must have a different label as the anchor. Then, triplet loss learns to distinguish a positive sample from a negative sample by

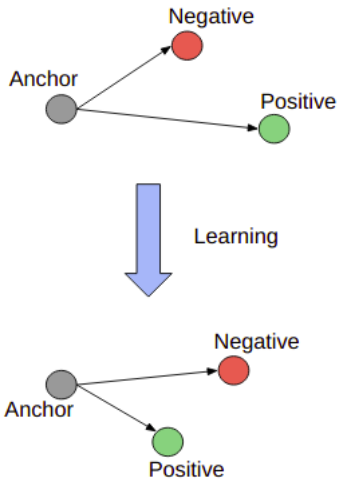

Figure 5: *Triplet Loss function learns to maximize the distance between anchor and negative sample, and to minimize the distance from anchor to positive sample.*

maximizing the distance between the anchor and the positive sample and minimizing the distance between the anchor and the negative sample illustrated in Figure 5.

The triplet loss function is described as:

$$L(A, P, N) = \max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha, 0) \quad (2)$$

where A is an anchor image, P is a positive sample of the same class as A, N is a negative sample of a different class from A, α is a hyperparameter for the margin between positive and negative pairs and f is an embedding. Based on the online hard example mining method [13], we choose for an given anchor a P that has the maximal distance to the anchor. We then take an N which has minimal distance to P.

In Figure 6, the loss curve converges after a long training only at our defined margin, which is equal to 1 for this experiment. That means our model can not learn anything to discriminate between P and N samples. We also try to decrease the learning rate slightly, but the loss curve does not sink.

In our opinion, the P and N samples are still too hard for training. Therefore, we try to clean the data in our next experiments.

5.5. Experiment 3: Triplet loss on matching pairs

In experiment number three, we expand our second experiment with data cleaning. Inspired by the winning team with matching pairs approach mentioned in 3. We also create matching pairs for our model and store at most 100 matching pairs for each class. The matching pairs are defined with a cosine similarity of 0.85 in our case, which is 0.35 higher than from the winning team. The reason is that

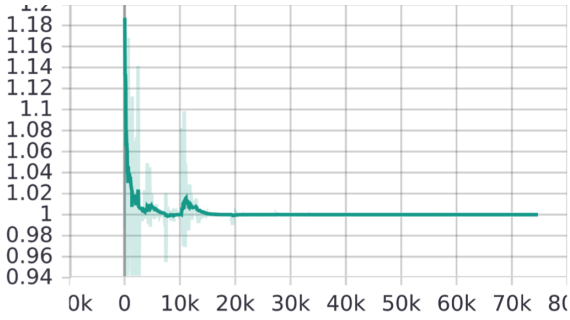

Figure 6: *Loss curve of experiment with triplet loss converges at the defined margin (1 in this case).*

Figure 7: *A matching pair of one class, which is defined by a cosine similarity of 0.85 based on its feature descriptors.*

we do not have enough capacity to train several models as the winning team does. Thus, we need more similar images to ease the learning process. An example of matching pairs can be seen in Figure 7. We then use each matching pair as an anchor and its positive sample. The negative sample is chosen from the remaining matching pairs within the mini-batch, which has minimal distance to the positive sample. The loss curve goes below the margin (0.2 in this experiment) shown in Figure 8. It indicates that the matching pairs help our network to separate positive from negative samples more easily. Finally, we get a GAP score of 1,248%.

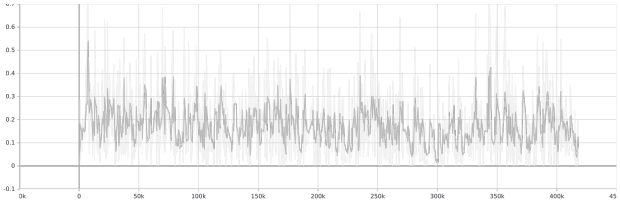

Figure 8: *Loss curve of experiment with triplet loss on matching pairs goes below the defined margin (0,2 in this case).*

5.6. Experiment 4: Triplet loss and center loss on matching pairs

In this experiment, we want to improve performance by using center loss. The principle of center loss is minimizing

intra-class variations for a better separation of features [14]. The formula of center loss is described as:

$$L_C = \frac{1}{2}\sum_{i=1}^{m} ||x_i - c_{y_i}||_2^2 \tag{3}$$

where x_i is a deep feature, c_{y_i} is the y_ith class center of deep features and m is the size of a mini-batch.

In Figure 9, the points represent deep features of different classes marked with different colors. The white points are class centers in each case. During training, these class centers are updated iteratively. Hence, the features of different classes are denser and easier to separate.

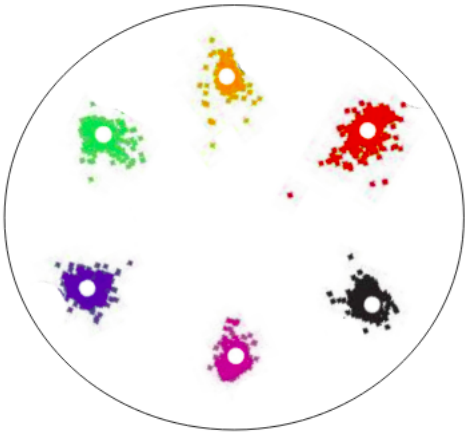

Figure 9: *Center loss function minimizes intra-class variation by updating the center of each class colored in white. Classes are marked in different colors. The points represent deep features colored according to its classes.*

Our loss function is now the sum of center loss and triplet loss. We train our ResNet50 on matching pairs again and get a GAP score of 0,308%. This result is worse than the one without center loss at first glance. However, it performs quite good after a short training time according to the end of our practical course.

5.7. Experiment 5: Mixture of experts and distance weighted triplet sampling

Based on the insights of the previous experiments, we designed a model, which tackles the problem of quite different inner class semantics. For this task we train multiple experts, each one assigned to its own domain. In detail, our model consists of k backbones as different experts without weight sharing and one backbone for gating, in order to weight the output of each expert.

Our loss function is triplet loss for the weighted output of the experts and cross entropy for the gating model. Due to

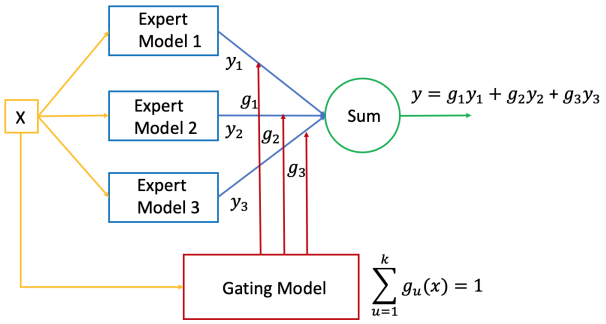

Figure 10: *Mixture of experts model, consisting of 3 experts, whose outputs are selected by the gating network [15].*

the fact that we do not have any information about the different semantics of the images, we use unsupervised clustering methods to generate the labels for the gating model. For this step, we use FINCH [16] because it provides more general clusters than K-means and is computationally tractable in contrast to spectral clustering. We iteratively embed all training images before each epoch with the gating model, cluster this embedding with FINCH, and use the assigned clusters in the following training of the entire model to train the gating model. The experts are trained with triplet loss, but in contrast to the previous experiments we use distance weighted triplet sampling[17] instead of hard example mining and matching pairs.

6. Results

Table 1 shows the results of our various experiments for the Google Landmark Recognition Challenge of 2019. The GAP metric is used for evaluation. Starting with our baselines, both of them only achieve a GAP of 0%. Our experiment concerning adaptive softmax scored second best from all our experiments with a result of 1.239%. However, this result was achieved after five days of training. Besides adaptive softmax, we also tried a second approach using triplet loss. But using only triplet loss we did not achieve a result any better than a GAP of 0% as our baslines. For this reason, we combined triplet loss with matching pairs which achieved 1.248% as GAP score. We also tried a combination with matching pairs, triplet loss and center loss, in order to improve our GAP score. Unfortunately, our expectation were not fulfilled and the combination of these three factors performed worse than only matching pairs and triplet loss. Thus, matching pairs and triplet loss is our best experiment being awarded as 100th place out of 176.

7. Outlook

According to our analysis, we develop some ideas to improve our approach. However, we have neither imple-

Method	GAP	Rank
Baseline: Random	0	130-176
Baseline: 1000 Classes	0	130-176
Adaptive Softmax	0.01239	100
Triplet Loss	0	130-176
Matching Pairs + Triplet Loss	0.01248	100
Matching Pairs + Triplet Loss + Center Loss	0.00398	113

Table 1: Performance of our different experiments on the private leader board from the Google Landmark Recognition Challenge 2019. The Global Average Precision (GAP) is used for evaluation. Our best result, which is the combination of matching pairs with triplet loss, is awarded for 100th place.

mented nor tested these ideas due to a lack of time and too late insights. In addition to our following proposed ideas, there are possibly more straightforward ways to improve our results, such as larger backbones and hyperparameter tuning, which we skip.

7.1. Iterative matching pairs creation

5.5 has shown that using easier positive examples for triplet loss greatly improves the training, especially in the beginning. The positive examples are too easy to improve the model later on during training, because due to the fact that a model, which has never seen the dataset, has created these examples. For that reason, we think using the trained model after some epochs to create new matching pairs and iteratively repeat this step can improve the overall performance in the end.

7.2. Re-Ranking

Besides the proposed ideas, we should have used re-ranking as post-processing for each experiment. The literature and especially the related works has shown that re-ranking improves the final results of almost every approach.

7.3. Probabilistic models

Due to the fact that the query images contain images that show no landmarks at all, probabilistic modelling is needed to detect outliers and to assign them to a low confidence score. Such a probabilistic modelling can be easily achieved by enabling dropout during inference. Through multiple feed forwards of the same input, the standard deviation from the outputs can be calculated and then used for the confidence score. However this method is computationally expensive.

8. Conclusion

In this paper, we presented our solutions for the practical course at KIT regarding the Google Landmark Recognition 2019 challenge. We had to deal with a vast uncleaned dataset, a massive number of classes, and limited computational power and memory. We tried several experiments based on pre-trained ResNet50 model and a last experiment with EfficientNet as backbone. It stands out that data preprocessing has a critical role because of the imbalance and inconsistency in the dataset. Finally, we get a GAP score of 1,248%, which leads us to the 100th place of 176 groups in total. In our opinion, we can receive a better score if we have more time to try other approaches such as re-ranking and better modeling of uncertainty for confidence prediction.

References

[1] "Google Landmark Recognition Challenge 2018," https://www.kaggle.com/c/landmark-recognition-challenge, accessed: 2019-08-19.

[2] "Google Landmark Retrieval Challenge 2018," https://www.kaggle.com/c/landmark-retrieval-challenge, accessed: 2019-08-19.

[3] "Google Landmark Recognition Challenge 2019," https://www.kaggle.com/c/landmark-recognition-2019, accessed: 2019-08-19.

[4] "Google Landmark Retrieval Challenge 2019," https://www.kaggle.com/c/landmark-retrieval-2019, accessed: 2019-08-19.

[5] Y. Gu and C. Li, "Team JL Solution to Google Landmark Recognition 2019," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[6] K. Ozaki and S. Yokoo, "Large-scale Landmark Retrieval/Recognition under a Noisy and Diverse Dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[7] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv preprint arXiv:1811.00982*, 2018.

[8] K. Chen, C. Cui, Y. Du, X. Meng, and H. Ren, "2nd Place and 2nd Place Solution to Kaggle Landmark Recognition andRetrieval Competition 2019," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[9] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[11] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *arXiv e-prints*, p. arXiv:1905.11946, May 2019.

[12] E. Grave, A. Joulin, M. Cissé, H. Jégou *et al.*, "Efficient softmax approximation for gpus," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1302–1310.

[13] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[14] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European conference on computer vision*. Springer, 2016, pp. 499–515.

[15] A. S Bock and I. Fine, "Anatomical and functional plasticity in early blind individuals and the mixture of experts architecture," *Frontiers in human neuroscience*, vol. 8, p. 971, 12 2014.

[16] M. Saquib Sarfraz, V. Sharma, and R. Stiefelhagen, "Efficient Parameter-free Clustering Using First Neighbor Relations," *arXiv e-prints*, p. arXiv:1902.11266, Feb 2019.

[17] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Sampling Matters in Deep Embedding Learning," *arXiv e-prints*, p. arXiv:1706.07567, Jun 2017.