CS231N Project Final Report: Landmark Recognition Challenge

Yu Zeng
Stanford University
zengyu@stanford.edu

Chaonan Ye
Stanford University
yec0214@stanford.edu

Wanling Liu
Stanford University
liuwl@stanford.edu

Abstract

Landmark recognition is a popular and challenging task in image classification. A great obstacle to landmark recognition research is the lack of large labeled datasets. In order to make progress in this problem, Google proposes Landmark Recognition challenges. We participated in this year's challenge to build models that recognize the correct landmark (if any) in a dataset of challenging test images.

In this report, we explained our methods in Section 3 and introduced our dataset and features, including image pre-processing in Section 4.1. Then, we conducted several experiments on the dataset and investigated the performance of different single models commonly used for image classification. The results are shown in Section 5. After that, we made use of all findings to design the final model, which is an ensemble of the best models. The baseline models are mainly extended on ResNet [9] SE(Squeeze-and-Excitation)Net [10] and Deep Local Features (DELF) [13] with transfer learning, attention mechanism [21], and some fine-tuning techniques. Besides quantitative evaluations, we included qualitative evaluations of models in Section 5.4.

1. Introduction

Landmark recognition can predict landmark labels directly from image pixels, which can be helpful for people to comprehensively organize their photo collections and understand the landmarks during traveling. However, a great obstacle is the lack of large labeled datasets. In order to foster progress in this problem, Google proposed Landmark Recognition challenges and presented the largest worldwide dataset to date. We participated in Google Landmark Recognition Challenge 2019. Our project is to address a large-scale classification problem with abundant imbalanced classes. Our inputs for models are 3-channel color images. Specifically, model inputs are tensors with shape $[N, 3, H, W]$, where N denotes the batch size, $H = W$ are the scaled image size and 3 is the number of RGB channels. For each output image, we have a prediction with a $\{landmarklabel, confidence\}$ pair.

Since the dataset is tremendous and imbalanced, which contains 5 million images with ground-truth labels, spanning 200 thousand classes, we first conducted experiments on a toy set which contains around top 2,000 frequent classes (each has at least 183 images) out of 200 thousand total classes. Using the toy set, we compared the performance of different models, explored different fine-tuning techniques, and tried several ensemble strategies to gain inspirations of designing a efficient system. After that, we combined strategies that yielded better results to train our models for a larger dataset (62% of the entire dataset) with around top 18,000 frequent classes. Each class in the larger dataset contains at least 50 images. The reason for such sub-sampling is to help models to learn more efficiently for remained classes. Also, training on the full dataset is highly time-consuming. Finally, we ensembled models trained on the top 18k frequent classes [1] to make final predictions on the official test set `https://www.kaggle.com/c/landmark-recognition-2019/data`. Our result shows that even though models did not utilized the entire dataset for training, the predictions are still reasonable. In addition to achieving a competitive final result, we thoroughly analyzed our hyper-parameters and models in both quantitative and qualitative ways.

2. Related Work

Landmark recognition is an image classification problem which is a fundamental task in computer vision. There are several recent papers to address this problem ([18], [4]). Bag-of-features models ([5],[23]) have been studied extensively but these can only be carried out on small-scale datasets. Nowadays, more studies have focused on large-scale datasets. For example, [16] compared different scalable methods for building an image-feature vocabulary and introduced a novel quantization method based on random-

[1] we call models trained on the top 18k frequent classes "top 18k models" and models trained on the top 2k classes "top 2k models" for short

ized trees over 1 million images. However, the performance would not be scalable without efficient ways to include spatial information in the index as the corpus size grows.

Regardless of the size of the dataset, most methods addressing the problem of Landmark Recognition are based on deep learning techniques. For example, in both [19] and [14], the authors combined a landmark proposal technique with convolutional neural network features to match patches over extreme appearance and viewpoint variations. However, these studies are all based on pre-trained CNNs that are trained on an object-centric dataset, which is distinct from the landmark recognition task. Additionally, visual features selections, such as HoG [6], SIFT [12] and SURF [2], are common techniques to solve the landmark classification problem as well.

3. Methods

Generally speaking, we first conducted experiments using only around top 2,000 frequent classes with ResNet50 [9], ResNet101 [9], MobileNet-v2 [17] and Inception-v3 Network [20]. Besides the original ResNet models, we added SE [10] layers in the ResNet50 bottleneck structure. Furthermore, we applied transfer learning (used the pre-trained weights from ImageNet [7]), attention mechanism, stage fine-tuning techniques, DEep Local Feature (DELF) [13] and simple data augmentations (flipping and color jittering). Additionally, we changed loss function from generally used cross entropy loss to focal loss due to imbalanced classes. After experiments on the toy set (with 2,000 classes), we combined best strategies (including selections of hyper-parameters and baseline models, fine-tuning techniques, layer extensions) and trained the models in a larger set of data with around 18,000 classes. Finally, we made an ensemble of our best models. Rigorous discussion of methods is in the following parts.

3.1. Loss Functions

For loss functions, we began with the most common cross entropy loss (CE):

$$CE = -\frac{1}{N}\Sigma_{i=1}^{N} log\hat{y}_{i,y_i}$$

where N is the number of examples, y_i is the ground-truth class for example i, $\hat{y}_{i,y_i}$ is the predicted probability for example i in class y_i, and it was calculated using softmax function ($\frac{e^{p_{i,y_i}}}{\Sigma_{j=1}^{M} e^{p_{i,j}}}$, where $p_{i,j}$ is the predicted probability for example i in class j and M is the number of classes). However, due to the fact of extremely imbalanced classes: some classes have over 10,000 images while some classes only have less than 10 images, the large classes contribute far more than the smaller ones does to the total loss. To address the imbalance, Tsung-Yi Lin et al proposed a loss function called focal loss [11] by down-weighting easy/confident examples so that contributions of larger classes to the total loss would be smaller than before. They added a modulating factor $(1-\hat{y})^{\gamma}$ to the cross entropy loss with tunable focusing parameter $\gamma \geq 0$:

$$FL = -\frac{1}{N}\Sigma_{i=1}^{N}(1-\hat{y}_{i,y_i})^{\gamma} log\hat{y}_{i,y_i}$$

We utilized this techniques in the experiments, expecting for improvements in the learning process.

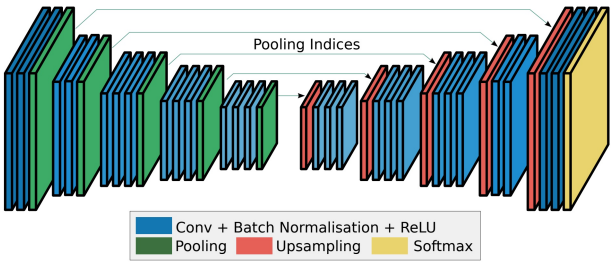


Figure 1. The cascading bottom-up and top-down structures used in SegNet. Image also adapted from SegNet. [3]

3.2. Attention Mechanism

The end-to-end soft attention structure aims to exploit fine-grained details in feature maps. The implementation usually resembles the encoder-decoder block [1] shown in Figure 1.

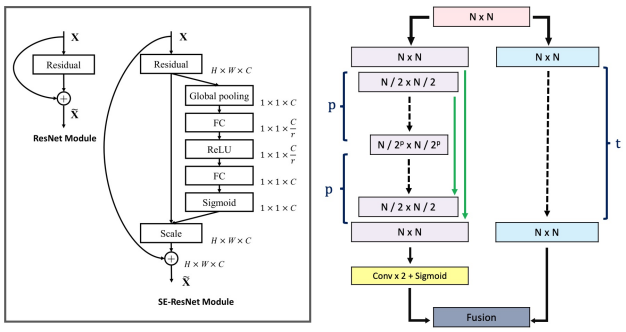
We adapted the SE-ResNet [10] to modify the pure residual unit in Attention Module [22] which tends to have add-on attention ability.

Our attention block is shown in Figure 2. Supposing the input size of the feature map is N $\times$ N, shown as the red block in the top, the input features are piped into two branches: (1) Purple branch. Each purple block is a SE-ResNet module depicted in left and then followed by a Max-pool layer with kernel size 3, stride 2 and padding 1 which halves the input size. After repeating p times, the size of intermediate layer is shrinked to size N / 2^p. To restore the reduced features, we interpolate the lower part bilinearly until achieving the original input size N. As to the yellow ending block, common convolutions and sigmoid layers are used to further adjust learning process. (2)Blue branch. Blue branch directly applies SE-ResNet block for t times.

The green long arrows are elementwise-sum. The last gray block represents fusion operation which could be formulated as

$$F_i(x) = (1 + P_i(x)) * B_i(x),$$

where i is the index of input images. F_i is the fusion result, P_i is the purple hourglass path while B_i is the blue SE-ResNet modules. The asterisk means elementwise-product.



(a) (b)

Figure 2. (a) The SE-ResNet block combines the residual path with squeeze-excitation mechanism. Image is from [10]. (b) Se-Attention module.

Note that the last layer of purple branch is sigmoid which squeezes the output into $[0, 1]$. Moreover, P_i is served as the feature selector to improve the learning capacity of original residual learing $F_i(x) = x + B_i(x)$.

The Se-Att (SE-ResNet-Attention) module is not restricted by the input size N since p is controllable, this allows us to plug it into our experimental frameworks flexibly.

3.3. Stage Fine-tuning

To better apply transfer learning to the dataset, we further incorporated a three-stage fine-tuning ("stage fine-tuning" for short) process instead of fine-tuning all parameters. The original idea was proposed by Catherine McNabb, et al[8]. Our detailed method is shown as following (refers to Figure 3):

1. Utilize pre-trained ImageNet[7] weights of ResNet50 on layers except the last fully-connected (FC) layer to convert images into feature vectors. This step is marked by the blue rectangle in Figure 3.

2. Use the extracted flattened features from the previous step to update the last FC layer, which adapts class categories from 1k trained the ImageNet to 18k for the dataset. The green part in Figure 3 represents this stage.

3. After training FC for several epochs (3 in our current experiment), we add two more top layers. Specifically, parameters of conv4_x, conv5_x, and FC labeled by the red rectangle are tuned.

3.4. DEep Local Feature (DELF)

Since the training set images are each associated with a unique class but the formal test set may depict no landmark, we plan to use DELF [13] during the testing phase to solve this problem. It can extract local features based on trained

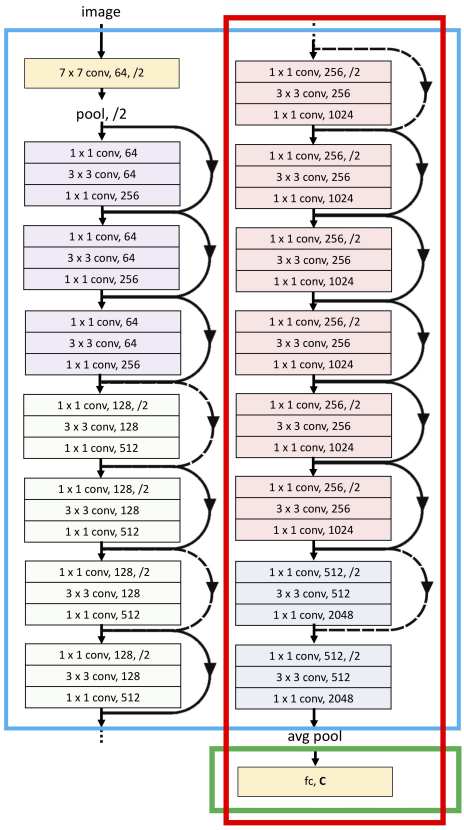


Figure 3. Three-stage finetune strategy. Stage 1: Blue; Stage 2: Green; Stage 3: Red

convolutional neural networks and matches them efficiently via an image retrieval system. We used it for matching local features of test images to training images known to have landmarks. If the matching points for the predicted class are less than one customized threshold, we claim that an image has no landmark.

It takes around 0.6 second per image which needs 35 days to complete all feature extraction given the massive amount of 5 billion data. Therefore, we resized the various image size to fixed size of 96×96. This pre-process helped us reduce the DELF time to 0.06 second for each image.

3.5. Ensemble

Ensemble is an appropriate approach to improve the model performance in practice. There are numerous ways to ensemble. We only tried mainly the following strategies:

1. Choose predictions with the highest confidence: for a specific image, choose the predicted class with the highest confidence among several models as the final prediction of this image and assign the confidence to this image to form an output pair $\{label, confidence\}$

2. Correct a top 18k model using a top 2k model: we

supposed that top 2k models would outperform for predicting the top 2k classes. Therefore, based on a model trained on the top 18k classes, replace its confidence with that of a top 2k model only when they have same predictions.

3. Increase confidence: based on one model, increase the confidence of its predictions of some images when other models have the same predictions by adding scaled confidences of other models.

4. Averaged or weighted confidence: based on the scores of the official test set, choose the model with better performance. If it has the same predictions as some other models, average their confidences or add their weighted confidence (according to the testing scores) together.

4. Dataset and Features

4.1. Dataset

The entire dataset is provided by Google (`https://www.kaggle.com/c/landmark-recognition-2019/data`). It contains 5 million images with ground-truth labels, depicting human-made and natural landmarks spanning 200 thousand classes. It was split into 3 sets of images: train, index, and test (we only used the training set and test set, the index dataset is for another challenge). The training set images are each associated with exactly one landmark. However, formal test images may be labeled by no landmark, one landmark, or more than one landmarks. Moreover, each category may contain quite diverse data: for example, images from a museum may contain outdoor images showing the building and indoor images depicting a statue located in the museum. Figure 17 in the 8 show some examples of the dataset.

4.2. Features

Our features are normalized 3-channel color images from the dataset, including some data augmentations. More specifically, model inputs are tensors of shape $[\mathrm{N}, 3, \mathrm{H}, \mathrm{W}]$, where N denotes the batch size, $\mathrm{H} = \mathrm{W} = 64|128|224|299$ are the scaled image size and 3 is the number of RGB channels. For each output image, we have a prediction with $\{$label, confidence$\}$ pairs. More details are explained following.

4.3. Pre-processing

We split the official training data into 3,719,622 training images(90%), 247,975 validation images (6%) and 165,316 temporary testing (4%) images during the experimental phase. The official test data was only used for final model submissions since the official test data has no labels released and 65% is private. We normalized the data via per-channel mean and std to re-range them from $[-1, 1]$ and resized them to the appropriate input image size according to different models. Additionally, we horizontally flipped the training data with a 0.5 probability and randomly deployed resize cropping, color jittering, rotation, scaling, shear or translation for the training data. Some of data augmentation methods might help the model to be invariant to specific conditions (e.g. by rotating images, the model may hopefully be invariant to different orientations of objects).

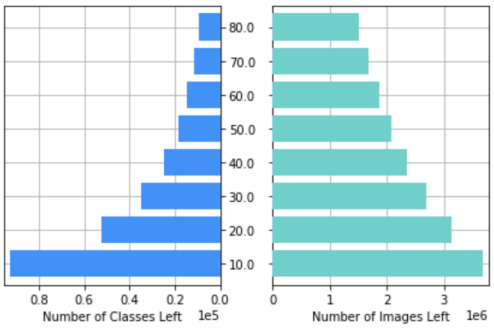


Figure 4. Removing classes which have less than n (shown as y axis) images

Besides, considering the extremely imbalanced classes, we removed classes that contain less than 50 images based on the class distribution (shown in Figure 4). Also, due to the amount of data and time limitation, we subsampled top 2,000 most frequent classes as our toy dataset and conducted model selection experiments on the toy dataset. Meanwhile, we removed all classes which have less than 50 images and left around 18,000 classes as the second dataset. The top 2 best-performed models for top 2,000 classes were applied to train the 18,000-class dataset.

5. Experiments and Results

Due to large-scale data and the long time of training process, we decided to first conduct experiments on a smaller amount of dataset (top 2k classes) and then carried out best models for the larger dataset (top 18k classes).

5.1. Evaluation metrics

We mainly used Global Average Precision (GAP) on the validation set as our evaluation metric. The GAP [15] is the official metric to quantitatively evaluate the results. Specifically, for each output image, we have N predictions with (label, confidence) pairs,

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

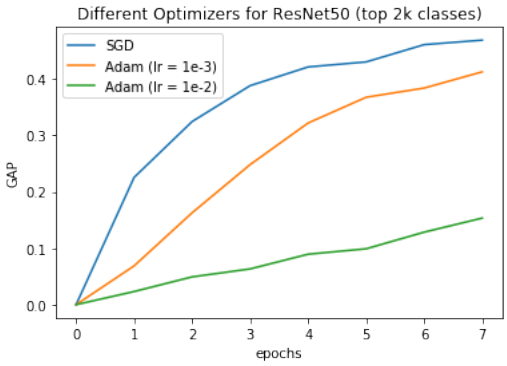


Figure 5. The performance of using SGD and Adam optimizer on ResNet50 for top 2k classes

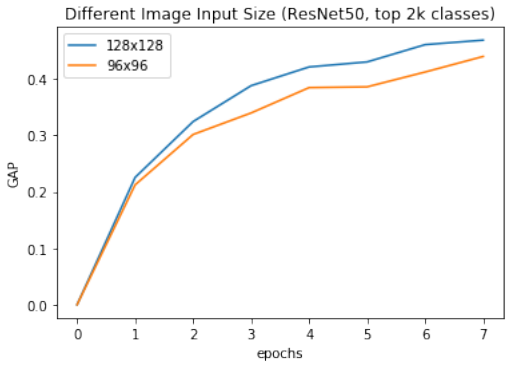


Figure 6. The performance of different input image sizes(128x128 and 96x96) on ResNet50 for top 2k classes on validation set

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 (note that some queries may not depict landmarks); $P(i)$ is the precision at rank i; $\mathrm{rel}(i)$ denotes the relevance of prediction i:

$$\mathrm{rel}(i) = \begin{cases} 1, & \text{if prediction } i \text{ is correct.} \\ 0, & \text{otherwise.} \end{cases}$$

Additionally, we also used accuracy and loss to evaluate our results.

5.2. Top 2k classes

For the top 2k classes, we tried ResNet50, ResNet101 as baseline models. Besides, we extended SE [10] layers in the ResNet50 bottleneck structure and trained models with different loss functions. Additionally, we tested the performance of Inception-v3 [20] and MobileNet[17].

5.2.1 Hyper-parameter Selection

We ran the hyper-parameter selection experiment mainly on the toy dataset. We tried different optimizers, learning rates, image sizes and loss functions on ResNets. Then, we utilized the outperformed one (i.e. SDG with learning rate 0.01, 128×128 image size, 256 batch size, cross entropy loss) for almost all other models on both the 2k dataset and the 18k dataset.

SGD vs. Adam optimizer We chose appropriate learning rates respectively for SGD and Adam optimizer and as we can see from Figure 5 that the GAP value (similar as prediction accuracy) of SGD (learning rate $1e-2$) increases faster than the Adam optimizer (learning rate $1e-3$ or $1e-2$). Therefore, we mainly used SGD as our optimizer.

Image size We tested the influence of different image sizes on the GAP value. As shown in Figure 6, the performance of larger image size can contribute to higher GAP.

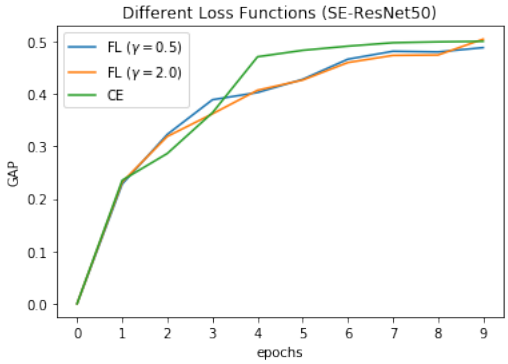


Figure 7. The performance of different loss functions on SE-ResNet50 for top 2k classes on validation set. We tested Focal Loss with gamma value of 0.5 and 2.0.

However, there is a trade-off between the GAP value and the training time: the larger the image size is, the longer the training time would be. For 96x96 image size, training each epoch costs around 1 hour, while for the 128x128 image size, it costs around 1.5 hours to train.

Cross entropy loss vs. focal loss We tested different loss functions on SE-ResNet50. However, as we can see from Figure 7 that the cross entropy loss performs similarly with different focal losses (no matter what gamma value we chose). Therefore, we decided to stick on the cross entropy loss due to fewer hyper-parameters and fewer calculations.

Learning rate decay We tried reducing the learning rate on top 2k dataset after 20 epochs from $1e-2$ to $1e-3$ for both MobileNet and Resnet50. The Fig. 8 shows that the performance was boosted after learning rate decay.

5.2.2 Model Selection

We conducted experiments on the top 2k dataset and then chose models that performed better to train on a larger top 18k dataset. In addition to ResNets, we also experimented

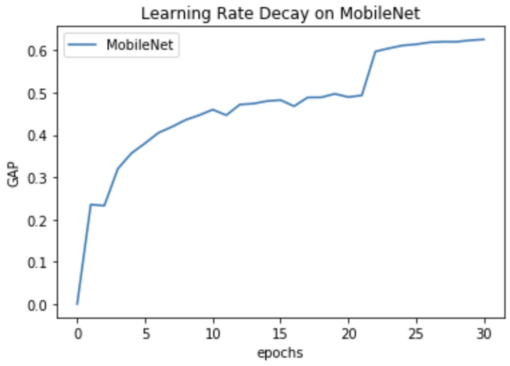


Figure 8. The performance of using learning decay on MobileNet for top 2k classes on validation set

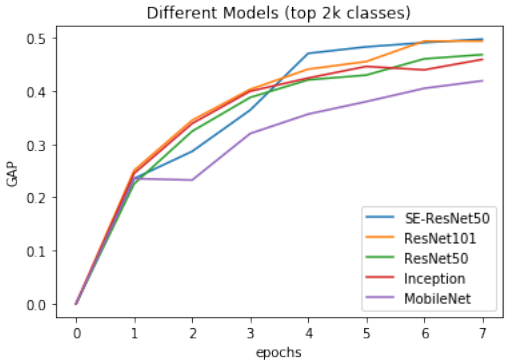


Figure 9. The performance of different models for top 2k classes on validation set

on Inception-v3 [20] and MobileNet[17], hoping to find a more suitable model to run on a larger dataset.

ResNet vs. SE-ResNet After the hyper-parameter selection, we trained SE-ResNet50, ResNet50, ResNet101 with the same hyper-parameters (i.e. SDG with learning rate 0.01, 128×128 image size, 256 batch size, cross entropy loss). As we can see in Figure 9 that ResNet101 and SE-ResNet50 performed similarly and a little better than the ResNet50.

MobileNet We tried MobileNet [17] due to its high efficiency and compactness. The input size of pretrained MobileNet is not as flexible as ResNet. Therefore, we picked $3 \times 224 \times 224$ as the input size and trained the model for 30 epochs. The optimizer is SGD with learning rate 0.01, momentum 0.9. However, the training time for one epoch is similar to ResNet50 (with $3 \times 128 \times 128$ input size and same hyper-parameters) due to its larger input size.

Inception-v3 Inception-v3 [20] was carried out on top 2k dataset due to its modest computation cost compared to simpler, more monolithic architectures. We selected SGD optimizer with learning rate 0.01, momentum 0.9, weight decay $1e-4$. However, the pre-trained model only accepted

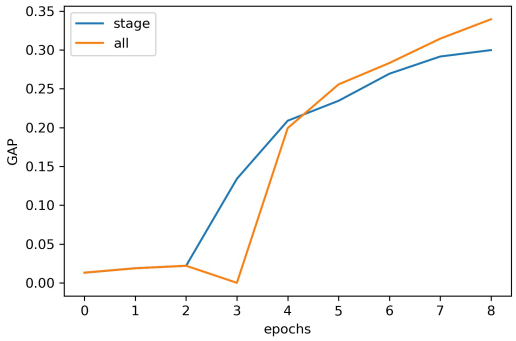


Figure 10. The result of finetuning all parameters and stage strategy

input size as $3 \times 299 \times 299$. Therefore, we decreased the batch size to 32 and only run 8 epochs to avoid costly time expense(3.5 hours per epoch).

Results Based on the performance of all models (shown as Figure 9), we can see that ResNet101 and SE-ResNet50 beat other models. In addition, considering their larger model capacity and less time cost which might be more suitable for the larger dataset, we mainly chose ResNet101 and SE-ResNet50 as baseline models for the top 18k dataset.

5.3. Top 18k classes

Based on the results from experiments of top 2k models, we mainly used SGD with learning rate 0.01, momentum 0.9 (for improving the gradient descent), and weight decay 1e-4 (for preventing overfitting), input image size 128×128 , batch size 256 (according to GPUs memory capacity).

5.3.1 Stage fine-tuning

We compare the 3-stage finetune and fully finetune strategy in top-18k to determine the transfer method. The result is shown in Figure 10: the fully tuned version has better GAP than 3-stage finetune. The reason could be the super inter-class difference requires more parameters to learn well. Thus, we decide to tweak all layers for transfer learning.

5.3.2 Single models

ResNets We firstly trained ResNet50, ResNet101 and SE-ResNet50 for 8 epochs, and we got the results shown in Figure 11. We can see from the figure that ResNet50 performed a little worse than the other two models which are consistent with our experiments in top 2k classes. Therefore, we decided to train ResNet101 and SE-ResNet50 for more epochs (20 epochs, see Figure 12) and submit our results for official testing at `https://www.kaggle.com/`

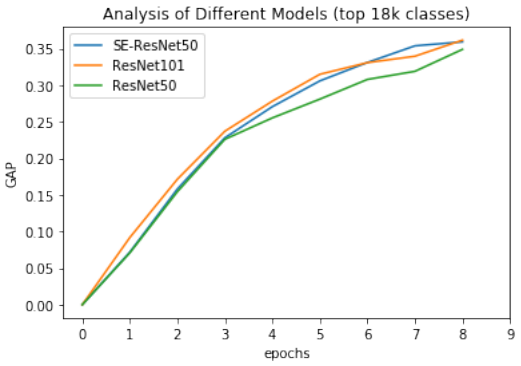


Figure 11. The performance of different models for top 18k classes on validation set

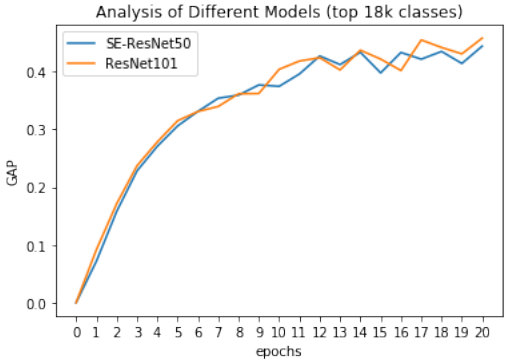


Figure 12. GAP of ResNet101 and SE-ResNet50 for top 18k classes in 20 epochs on validation set

`c/landmark-recognition-2019/leaderboard` (refer to Table 2). After submitting those predictions to the official website, we got the best scores of ResNet101, SE-ResNet50: 0.02616, 0.03317 respectively.

SE-ResNext with attention mechanism We attached attention module as described in 3.2 to each of the SE-ResNeXt blocks excluding the last one. The whole architecture is: Conv1 $\rightarrow$ Maxpool $\rightarrow$ [SE-ResNeXt + **Attention**] $\times$ 3 $\rightarrow$ SE-ResNeXt $\times$ 3 $\rightarrow$ Average pool $\rightarrow$ FC. The comparable result is shown in Figure 13. We can find the attention module do improve the classification accuracy and the speed of converge. We got best test score 0.02288 for this model from the official website.

5.3.3 Ensemble

Finally, We tried ensemble strategies described in Section 3.5 and the results are shown in Table 1. It can be noticed that only weighted confidence approach works well: the ensemble of ResNet101 and SE-ResNet50 improves the higher SE-ResNet50's GAP by 1.4%) and all other strategies increased the lower GAP but decreased the higher GAP.

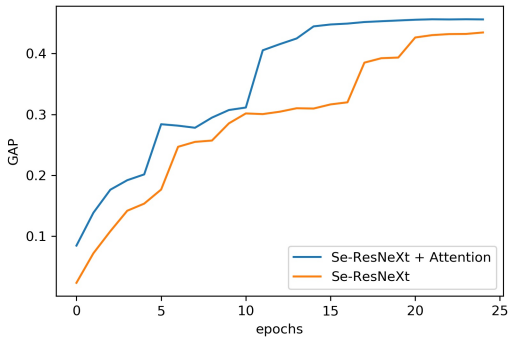


Figure 13. GAP of SE-ResNeXt50 with / without attention module on validation set

		Method	Ensemble
ResNet101-2k	ResNet50-18k		
0.00608	0.01967	1	0.00608
		2	0.01800
		3	0.01993
SeAtt50-18k	ResNet50-18k		
0.02288	0.01967	4	0.01993
Se-ResNet50-18k	ResNet101-18k		
0.03317	0.02616	5	**0.03364**

Table 1. Experimental ensemble results

Therefore, we used the best result we got from this weighted confidence strategy and filtered out images with no landmark using DELF. We used the result given by this model for our final submission and achieved top 24% in this competition (our final results are shown in Table 2).

Models	ResNet101	SE-ResNet50	Ensemble	Ensemble + DELF
Public Scores	0.02616	0.03317	0.03364	0.04756

Table 2. Final ensemble results

5.4. Qualitative evaluation

5.4.1 Saliency Map

We used saliency map to visualize our results. For example, as we can see from Figure 14, our SE-ResNet50 classified this image correctly while ResNet50 misclassified this image. Since the saliency map computes gradients of (unnormalized) class scores with respect to image pixels and takes absolute value and max over RGB channels, it can tell us what pixels matter in the image. We can get some intuition about why those two models perform differently from the saliency map: SE-ResNet50 focused more on the landmark

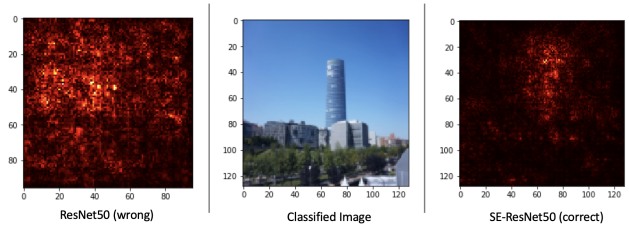


Figure 14. Saliency map example: SE-ResNet50 (correct classification) vs. ResNet50 (misclassification)

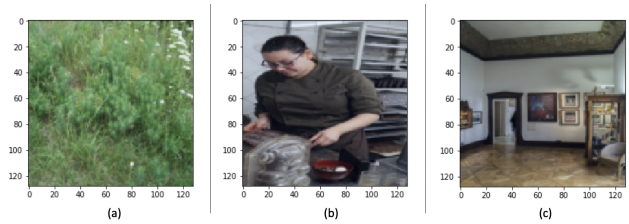


Figure 15. Examples of hard cases

(the tall blue building) itself, while the ResNet50 focused on almost the entire image. It means the SE-ResNet50 model may be better at choosing important pixels than ResNet50, so it would be more likely to classify the image correctly.

5.4.2 Examples of classification results

In this part, we share some examples of images in the training set and classification results of SE-ResNet50 on the validation set to discuss the model. In Figure 15, we show some hard cases for the model to determine the landmarks.

Since the training dataset is not clean, we have some pictures like grass (Figure 15 (a)) and people (Figure 15 (b)) which may not contain effective information of landmarks. This indicates problems since those pictures contain more general information which could appear in any place even without specific landmarks, but the model still tries to classify them into specific landmarks, which could cause future misclassification. Another hard case is that the dataset contains some indoor scenes for some landmarks (e.g. Figure 15 (c)). Sometimes only from the outside appearance of some landmark, can we tell which landmark it is. Thus, in those situations, it can be very hard for the model to classify the pictures taken from the indoor scenes.

In Figure 16, we show some correct classifications of the model. We can see from those images that the model has a certain degree of robustness since it can still correctly classify the landmark in the picture when the scene was taken from some special viewpoints (e.g. Figure 16 (a)). Also, it can make good predictions when there are some occlusions in front of the main landmarks (e.g. Figure 16 (b)). It can be

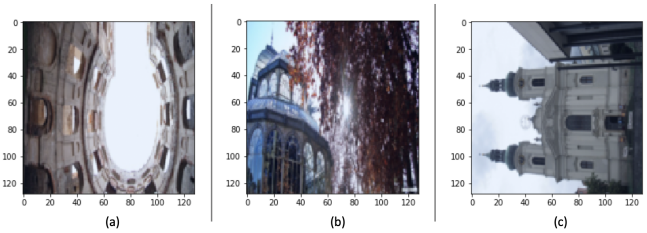


Figure 16. Examples of correct classification

invariant to the rotation of the landmark as well (e.g. Figure 16 (c)).

6. Discussion

As for ensemble strategies, we found that choosing predictions with the highest confidence among several models is usually a bad idea. That is because suppose we use a top 2k model and a top 18k model, the top 2k model with very low testing score in Kaggle usually predicts high confidence since the uneven data distribution encourage the classifier to learn the majority. Suppose the private testing data is nearly identically distributed as the training data, the top 2000 are still the most frequent classes, so the 2k models tend to predict confidently.

7. Conclusion and Future Work

In conclusion, we conducted several experiments to address large-scale landmark recognition problem. We generated two datasets: one with top 2k frequent classes and the other with top 18k frequent classes, since we have a very large dataset and trying to train classes with too few images may be inefficient (since those classes would be harder to train with only a few examples). Then, we did hyper-parameter selection and model selection on the top 2k dataset and deployed the top 2 out-performed models: ResNet101 and SE-ResNet50, on the top 18k dataset. Besides, we tried the attention mechanism and stage fine-tuning for the top 18ks dataset. Finally, we made an ensemble of best models with DELF to generate the final model. Moreover, our model beats 76% teams in this competition.

Even though we removed all classes which have less than 50 images, the training dataset was still massive and noisy. For example, some plants images are labeled as landmarks in the training set. In the future, we would like to clean the training data to reduce the noise first, and then train the model. We believe that it would help our models to learn more efficiently. Also, adding more data augmentations may help the model to be invariant to more conditions (e.g. the different brightness of lights), which would lead to a more robust model.

Id: 020b9a2e84420614 ; landmark_label: 137910

Id: 020a2fde87a79dc8; landmark_label: 153152

Id: 0201d18189f08dbb; landmark_label: 81825

Figure 17. Example of training images: each image has an Id and an associated label.

8. Appendices

The Figure 17 shows examples of the traning dataset.

9. Contributions and Acknowledgements

All of us wrote or modified code for our models, and trained and fine-tuned models. Y.Z mainly made contribution to stage fine-tuning, DELF and attention mechanism, C.Y was mainly in charge of MobileNet, Inception Network and ResNets, W.L mainly contributed to ensemble, ResNets and qualitative evaluations. All three wrote the reports. Overall, we made contributions to the project evenly.

The GitHub link for the project is `https://github.com/zengyu714/landmark-recognition`.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[4] Z. Chen, A. Jacobson, N. Sünderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, and M. Milford. Deep learning features at scale for visual place recognition. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230. IEEE, 2017.

[5] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] A. Garg. Google landmark recognition using transfer learning. `https://towardsdatascience.com/google-landmark-recognition-using-\transfer-learning-dde35cc760e1/`. Dec 14, 2018.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[10] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[13] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han. Large-scale image retrieval with attentive deep local features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3456–3465, 2017.

[14] P. Panphattarasap and A. Calway. Visual place recognition using landmark distribution descriptors. In *Asian Conference on Computer Vision*, pages 487–502. Springer, 2016.

[15] F. Perronnin, Y. Liu, and J.-M. Renders. A family of contextual measures of similarity between distributions with application to image retrieval. *2009 IEEE Conference on Com-*

puter Vision and Pattern Recognition, pages 2358–2365, 2009.

[16] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[18] P. Shukla, B. Rautela, and A. Mittal. A computer vision framework for automatic description of indian monuments. In *2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 116–122. IEEE, 2017.

[19] N. Sünderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. *Proceedings of Robotics: Science and Systems XII*, 2015.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[21] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2017.

[22] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. *CoRR*, abs/1704.06904, 2017.

[23] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*, 73(2):213–238, 2007.

Code Libraries

1. PyTorch. `https://pytorch.org/`

2. ResNet. `https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py`

3. Dataset loader. `https://pytorch.org/tutorials/beginner/data_loading_tutorial.html`

4. SENet. `https://github.com/moskomule/senet.pytorch`

5. Attention module. `https://github.com/tengshaofeng/ResidualAttentionNetwork-pytorch/blob/master/Residual-Attention-Network/model/attention_module.py`

6. DELF

   - `https://github.com/anishagg/Google-Landmark-Recognition/blob/master/Scripts/DeLF.ipynb`
   - `https://tfhub.dev/google/delf/1`

7. Pretrained models. `https://github.com/Cadene/pretrained-models.pytorch`