The Combination of Convolutional Neural Network and Feature Matching in Fine-grained Localisation

Yuanqi Lin Haoyuan Yu

***Abstract*—Geolocation is an important problem in many AI applications. However, GPS may not be accurate enough to provide fine-grained geolocation in a small indoor or outdoor environment. In this case, recognizing the location from image information becomes particularly important. In this project, we will propose two ways to find the geolocation from an image, a pretrained CNN model, and feature matching through SIFT and FLANN.**

***Index Terms*—CNN, SIFT, Fine-Grained Location Recognition**

I. Introduction

Geolocation is highly important nowadays in some fields of Artificial Intelligence applications. For example, autonomous vehicle navigation uses the network address and GPS coordinates to find the user's current location. However, depending on the required accuracy level, different geolocation techniques need to be used. Specifically, the image data becomes vital when finding the fine-grained geolocation in some small indoor and outdoor environments because GPS coordinates and network addresses are not accurate enough to provide the fine-grained position data.

This project utilises a set of training images with known locations and a set of testing images taken in random places in the same museum to recognize the exact location of each testing image.

This report will firstly introduce the possible methods: Convolutional Neural Network (CNN) and the feature matching methods, then discuss their performance. With the support of experimentation, a result solution will be given producing the closest coordinates to the true ones for each testing image.

II. Methods

A. Convolutional Neural Network (CNN)

First, we try to build up a Convolutional Neural Network to automatically study features for each training image and then predict the coordinate of each test image.

Figure 1 shows that the distribution of coordinates is sparse, and there are 7500 different images with 1499 different coordinates (labels) in the training set, so it is difficult to classify this kind of training set. Therefore, we choose to do regression that the CNN model could generate coordinates for each test image.

Figure 2 shows the structure of the CNN model that we build up. We utilise the ResNet50 [1] model pre-trained on ImageNet without tuning the parameter because tuning the ResNet50 parameters consume a lot of memory, and we want to use the weight of ImageNet. In this model, all convolutional layer uses activation function *relu* and the final output layer use activation function *linear*.

In order to figure out the actual performance of the CNN model, 20% of the training set are split into validation sets. Also, the loss function *Mean Absolute Error* are utilised because the same loss function is used when evaluating the testing set. However, due to the limitation of memory size, the input image needs to be resized to 200×200, which would influence the performance of the CNN model because some information may be lost.

Several experiments have been set up to discover neural network structures and fine-tuned hyper-parameters with lowest *Mean Absolute Error (loss)*. The detail of the experiments will be discussed in Section III-A.

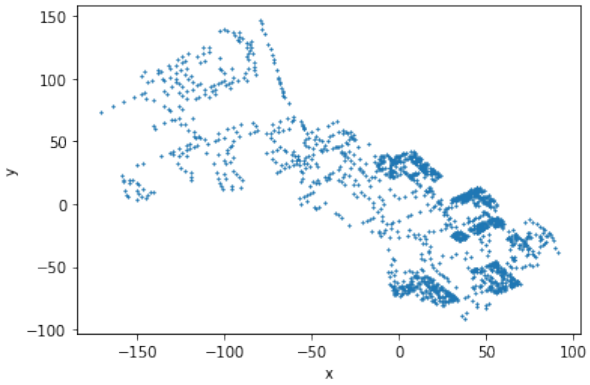


Fig. 1: Training Set Coordinates Distribution

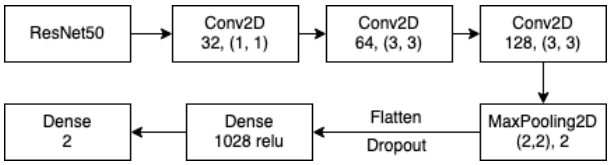


Fig. 2: Structure of CNN

B. Feature Detection: SIFT

Scale-Invariant Feature Transform (SIFT) is one of the most popular feature detection algorithms. It can be used to locate and describe local features from images. There is a

range of other popular algorithms, Speeded Up Robust Feature (SURF), was not experimented in this project due to copyright restrictions, but it should be ideal since it is a faster version of SIFT. Oriented Brief (ORB) cannot detect rotations and scaling, while sensitive to noise. The detected features in SIFT, however, are more accurately positioned in both space and scale.

The only adjustable parameter in SIFT is the Hessian minimum threshold, it determines how significant the output point must be to be considered a point of interest. The higher value means a smaller number of the points detected, and the points are more accurate. By contrast, a lower threshold means more features will be observed. To get as many points as possible, the minimum value was chosen to produce the maximum amount of key points, details will be discussed in Section III-B.

The main issue with SIFT is the time complexity. In order to compare each testing image with each training image, the original idea was to perform SIFT on both pictures every time. Due to the high time complexity, it could take over 580 seconds to do a complete round of matching between a testing image and the whole training set, which is not ideal. Regarding this problem, we decided to perform SIFT on all the testing and training images before matching, and save the describers in a python dictionary, which can be accessed directly in later steps.

C. Feature Matching: FLANN

Fast Library for Approximate Nearest Neighbours (FLANN) is a collection of a range of algorithms designed for closest neighbour search, performs especially great with large datasets and high-dimensional features [2]. After obtaining the describers from the last step, FLANN is used to find the matches according to those key points. In order to keep the suitable matches and remove the matches created by noise, Lowe's Ratio test was used. After getting the two best matches (K=2 for K nearest neighbours) for each keypoint in the testing image, the two points are expected to be different enough to be considered a helpful point, otherwise, it would be noise to be removed. The lower the ratio, the less and more accurate matches will be found. The threshold was set to 50% finally after experiments.

FLANN, by its definition, is a solution to large datasets. Instead of making a one-to-one comparison between each testing image and training, it provides methods to add all the describers into a single matcher and then train at once. Using the pre-trained matcher and the descriptors of the testing images, the performance and the speed of the model is optimized. Compared with the original method of individual comparing, the speed of the whole process can be reduced from days to 4 minutes.

D. Combination: CNN and Feature Matching

After applying CNN and the feature matching methods (SIFT & FLANN) separately to the image datasets, the performance of each technique was compared.

CNN can provide a result for each testing image, but the overall accuracy is lower than feature matching. For feature matching, after extracting features and training the matcher, we labelled the testing image's coordinate the same as the best-match image's coordinates. It is both accurate and computationally efficient, but some testing images may fail to find a match in the training set.

The solution here is to combine the two methods to get an optimal result. CNN will perform as a baseline strategy, for those who fail to compute a match in the FLANN matcher, instead of giving it an empty or random coordinate, the CNN results were used.

III. Experiment

A. Structures of CNN

Several experiments have been done in order to figure out a convolutional neural network model with the lowest loss. In all experiments below, *Mean Absolute Error* is utilised as loss, batch size 128, number of epoch 20 and learning rate 0.001.

First, we tried different pre-trained CNN models such as ResNet50, ResNet101, VGG16 and VGG19 weight on ImageNet without training the parameters. When testing the performance of different pre-trained CNN models, a Multi-Layer Perceptron with 1024 hidden nodes and activation function *relu* was appended to the output layer of the pre-trained model.

Table I demonstrates the loss for the different pre-trained models on both the training and validation set. ResNet50 have the lowest loss on both collections of images. However, all these models are overfitting the training set. We considered that a simple Multi-Layer Perceptron is not suitable for the image regression problem.

Pretrained Model	Loss (training set)	Loss (validation set)
ResNet50	**2.87**	**17.35**
ResNet101	3.65	18.06
VGG16	4.25	19.89
VGG19	4.38	21.09

TABLE I: Loss for Different Pretrained Model

In order to solve the overfitting problem, we added some convolutonal layers to the output of the ResNet50 model and a dropout layer before output the labels. We attempted different combination of convolutional layers and max-pooling layer with dropout 0.3.

Table II shows the number of convolutional layers and corresponding loss. The hyperparameters for convolutional layers is same as the hyperparameters defined in Figure 2.

Num of Conv2D	Loss (training set)	Loss (validation set)
1	8.96	15.13
2	6.86	13.69
3	**6.31**	**12.37**

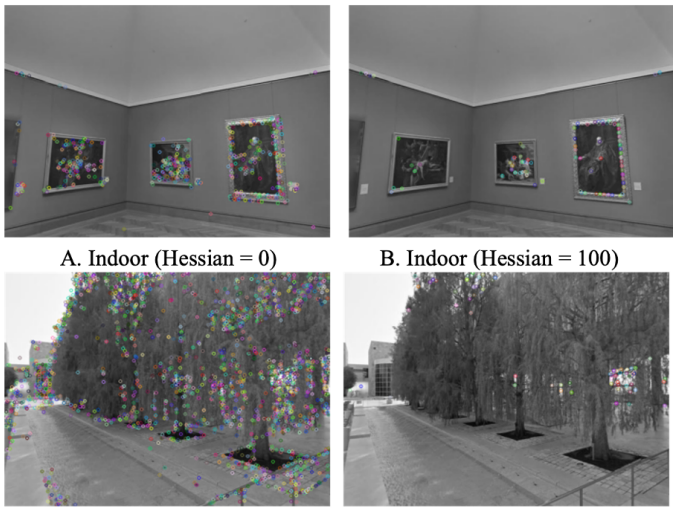
TABLE II: Loss for Different Pretrained Model

Table II illustrates that when three convolution layers are utilised, the loss on both training and validation set become lower.

Therefore, the convolutional neural network defined in Figure 2 are used. The *Mean Absolute Error* of the CNN model on 50% of the testing set is 11.93 and this model work as the baseline model in this project.

B. Threshold determination in SIFT

During experimentation, we found that SIFT is especially good at detecting indoor objects such as desks, chairs, and paintings. As shown in Figure 3, not only were the shape of the frames captured, but also the details of the paintings. For the outdoor objects, trees for example, the detected key points of the leaves are a bit messy, these noises can affect the matching in later steps. After adjusting the Hessian minimum threshold, the noise was effectively reduced, but the valuable points were removed at the same time, and the performance of the indoor ones was affected as well. It is worth mentioning that, for the outdoor images, not only do the complex objects such as trees matter, but also other simpler objects such as the handrail and the buildings. Keeping the threshold low helps preserve potential matches on those objects.



A. Indoor (Hessian = 0) B. Indoor (Hessian = 100)

C. Outdoor (Hessian = 0) D. Outdoor (Hessian = 100)

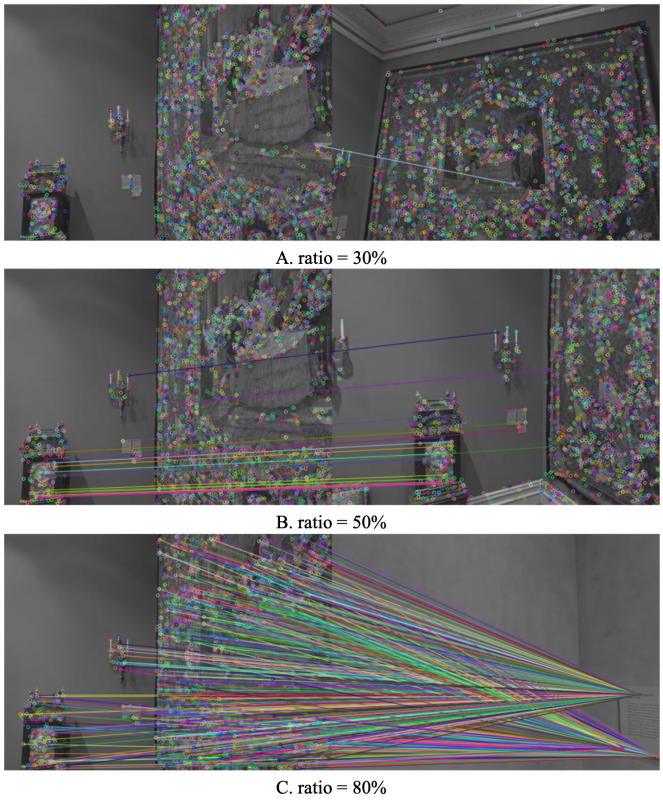
Fig. 3: The contrast between indoor and outdoor images under different thresholds

C. Lowe's Ratio Test: How to determine the ratio of distance?

In this section, the experiment results were given by running individual one-to-one FLANN instead of one-to-all to visualize the result more efficiently.

When Lowe proposed the ratio test to solve the problem of unnecessary keypoints [3], he argued that the matching gives the most number of correct matches when the ratio is set between 0.4-0.6. Our experiments have proven this opinion. As shown in Figure 4 and Table III, when the ratio value is set to 0.3, the restriction is so strict that only two matchings are found. On the contrary, when the value equals 0.8, the limitation is too mild, a lot of messy and inaccurate matchings are found. The statistics showed that 606 matches were found in this pair of images, all of them are noise. After checking the process log, we found that the actual match image has much fewer matches than this pair so that the model ignored it. Eventually, when the ratio is 0.5, the matches seem even and precise, and the best match image is much more accurate than the others.

It is worth mentioning that, after the combination with FLANN matcher, the ratio value was adjusted to 55% and proven to be most effective.



A. ratio = 30%

B. ratio = 50%

C. ratio = 80%

Fig. 4: The best match computed under different ratios of distance

Ratio value	# of matches	Time consumption (s)	Correct match?
0.3	3	396	No
0.5	27	395	Yes
0.8	606	394	No

TABLE III: The numerical performance under different ratios of distance

D. FLANN matcher: one-to-one or one-to-all?

In the early stage of the experiments, as shown in the Pseudocode 1, a new FLANN matcher was constructed for each match. The improved version (Pseudocode 2), however, adds all the describers of the training images to the matcher and trains at once. The most direct effect of this adjustment is the significant reduction of time complexity. After adding and training, the FLANN matcher builds a K-d tree, and the time complexity of finding the nearest neighbour is reduced

from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$. In practice, this approach can save the computation time from approximately 50 hours to 4 minutes.

Apart from reducing time complexity, training one matcher for all the training images can also help improve the matching accuracy. Unlike CNN, it is difficult to give a numerical result for accuracy since we cannot tell if the match is the truly best one even by human intervention. But we did prove the improvement through witnessing the decrease in the *Mean Absolute Error (MAE)* in the final testing stage, where the distance from the computed coordinates to the proper coordinates was calculated. The MAE value was successfully reduced from 6.36 to 3.88.

Algorithm 1 One-to-One FLANN Matcher

for $testing = test1, test2, \ldots, test1200$ **do**
  for $training = train1, train2, \ldots, train7500$ **do**
    new FLANN matcher
    $flann.knnMatch(des\_test, des\_train)$
    Compute the best match
  end for
end for

Algorithm 2 One-to-All FLANN Matcher

new FLANN matcher
for $training = train1, train2, \ldots, train7500$ **do**
  $flann.add(des\_train)$
end for
$flann.train()$
for $testing = test1, test2, \ldots, test1200$ **do**
  $flann.knnMatch(des\_test)$
  Compute the best match
end for

IV. Result

Table IV shows the *Mean Absolute Error* under different combinations of thresholds . The best result was given by combining CNN with SIFT (Hessian = 0) and one-to-all FLANN (distance ratio = 0.55).

CNN	SIFT	FLANN	FLANN(ratio)	MAE
Yes	\	\	\	11.93
Yes	Yes	one-one	0.5	6.36
Yes	Yes	one-all	0.5	3.88
Yes	Yes	one-all	0.55	**3.69**
Yes	Yes	one-all	0.6	3.87

TABLE IV: The MAE value under different approaches

V. Conclusion

In this project, we proposed two methods to find the fine-grained geolocation from images, a CNN model and a feature matching method using SIFT and FLANN. Each solution has its own merits and demerits, the combination of them incorporates their advantages, and gives an overall more satisfying result.

To better understand how neural network could helps the fine-grained geolocation problem, more thorough experiment with different network structure will be required in the future. Also, computing the likely change in pose between the training and test images using geometric constraints after finding the matching images may helps to predict more accurate location.

References

[1] 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), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[2] M. Muja and D. G. Lowe, "Scalable Nearest Neighbor Algorithms for High Dimensional Data," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 11, pp. 2227-2240, 1 Nov. 2014, doi: 10.1109/TPAMI.2014.2321376.

[3] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60, 91–110 (2004).