Photo Geolocation Recognition Based on Convolutional Neural Networks

Yancheng Li
Department of Statistics
Stanford University
lycheng@stanford.edu

Yifan Yu
Department of Statistics
Stanford University
yifanyu@stanford.edu

Yao Chen
Department of Electrical Engineering
Stanford University
yaochen1@stanford.edu

Abstract

Due to the increased availabilities of open-source photos with geotags and the recent advancements in computer vision, photo geolocation based on photo content seems to be a more realizable task. Lots of researches have been done on extracting location information from the buildings captured in photos, yet most of them focus on landmark buildings only. For our work, we used convolutional neural networks (CNNs) to obtain location information from both landmark and non-landmark buildings. We created our own dataset using photos from Flickr.com [1] and applied different CNN architectures with both self-trained weights and pre-trained weights. A customized version of the well-known AlexNet model achieved a 56.7% accuracy of classifying photo from 9 different cities based on architectural information.

1 Introduction

Automating photo geotagging process remains one of the most challenging tasks in the field of computer vision field. Some of previous researches pay attention to the buildings captured in photos, as architectral features and style contain rich location information. Yet most of them chose to focus only on landmarks buildings [2]. In our project, we study both landmark and non-landmark buildings, as we believe both of them capture valuable information about places. Each city's non-landmark buildings are also a major part of its culture and defines it. Also, non-landmark buildings are far more frequently captured in photos than landmark buildings. Therefore, if we can utilize location information hidden in non-landmark buildings, geotagging general street-view photos will be much easier. Therefore, the goal of our project is utilize CNN techniques to construct a model, which takes an image as an input and generate a probability distribution over location labels. To achieve this goal, We started with creating a dataset consisting of building images of nine cities from Flickr.com [1]. Then, we treated the photo geolocation problem as a classification problem, and implemented 3 different CNN architectures with softmax classifiers. We experimented with both training a model from scratch and fine tuning existing weights. Our best model is a customized version of the AlexNet model, which achieves a 56.7% accuracy in this 9-label classification problem.

2 Related work

Landmark recognition systems have caught significant public attention over the past decade. Avrithis et al. [3] and Gammeter et al. [4] proposed an image clustering scheme and recognize the landmark in each query photo using image mining and retrieval. With the success in landmark recognition tasks, researchers began to dedicate into solving the city-scale geolocalization problems incorporating the landmark identification system. Similarly, at early stage, most of them were done by retrieving invariant features using non-deep learning methods, including decision tree, K-Nearest Neighbors, K-means clustering, and SVM classifier. Schindler et al. [5] explored a large image database for each city and built vocabulary trees at city level using the most informative features. Bergamo et al. trained SVM on a set of local feature descriptors obtained by motion computation on training data [6]. Hays and Efros trained a Pre-exemplar SVM classifier for each location in the database and yield a significant boost in accuracy with model Im2GPS [7]. All these models require image retrieval from the large image database, which is space inefficient and lacks matching flexibility.

Unlike previous researchers who approached the photo geo-localization problem using feature retrieval methods, in 2016, Google posed a brand-new technique that treats it as a classification problem. They realized photo geolocation with a simpler and less resource-intensive model called PlaNet that aims to geolocate arbitrary photos[8]. Instead of using the nearest neighbor matching approach, Google trained a CNN with batch normalization on millions of geotagged photos, and it produces a probability distribution over multi-scale geographical regions for a given new image. With 3.6% accuracy on recognizing images at street-level and 10.1% at city level, Planet outperforms all the other geo-localization models so far.

Similar to the PlaNet, we chose to approach the geolocation problem by treating it as a multi-classification problem and utilized CNN techniques instead of feature retrieval methods. Yet instead of studying arbitrary photos, we focus on photos capturing buildings, which we believe contain rich location information.

3 Dataset and Features

The Flickr collection consists of billions of high-quality photographic images of over 75 million users. Based on Flickr search engine, we first built a dataset consisting of city labeled architectural photos. We chose nine major cities around the world and collected approximately 4500 photos containing both landmarks and non-landmarks photos for each city.After building the dataset, we resized the original images, split them into train set and dev set, and then used them as the input for our models. The detailed information about the original dataset is given below.

City(Label)	Shanghai(0)	New York(1)	Kyoto(2)	Rome(3)	Morocco(4)	Paris(5)	Seoul(6)	Amsterdam(7)	Vienna(8)
Train Size	3589	3650	3579	3588	3454	3573	3575	3496	3597
Dev Size	898	913	895	897	864	894	894	874	900

Table 1: Dataset

3.1 Data Augmentation

We also implemented several data augmentation techniques such as random distortion, left-right flipping, zooming and rotatation to obtain extra photos for training, which were used when we encountered overfitting in model experiments.

4 Methods and Experiments

As our project is forumlated as a classification problem, all the models were trained with the softmax cross-entropy loss using AdamOptimizer. The loss formula is the following:

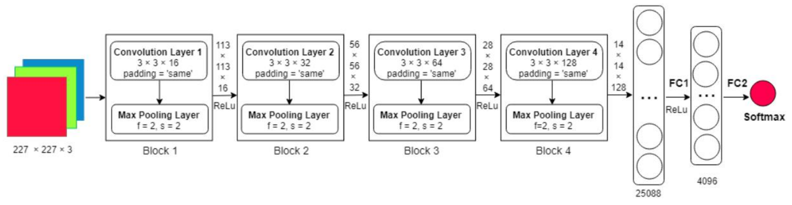
$$L = -\sum_i y_i log(p_i),$$

where y_i is the true class probability for certain input image i and p_i is the predicted probability generated by the model.

4.1 Baseline Model

Before borrowing any existing successful convolutional neural networks architectures, we first designed and trained a simple neural network based on CS230 starter code [9] as our baseline model. This model consists of 4 convolutional layers, 4 max pooling layers and 2 fully-connected layers. The detailed architecture of the model is shown below.

Baseline Model Pipeline

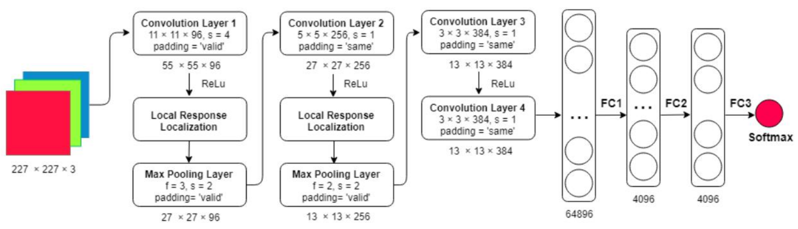


When training this model, we utilized the batch normalization technique, and implemented dropout for the fully-connected layers. We experimented with 4 different keep probabilities (keep-prob = 0.3,0.4,0.5,0.6) and 3 different learning rates (learning-rate = 1e-3, 1e-4, 1e-5).

4.2 Customized AlexNet Model

For this model, We started with the architecture of AlexNet-places365 [10], a convolutional neural network trained for scene classification task. We converted the original caffe model into a Tensorflow model[11] and implemented it. Compared with the AlexNet architecture mentioned in class, this model has two additional local response normalization layers[12]. We reduced the architecture's complexity by removing the fifth convolutional layer and the last max pooling layer, due to the overfitting issue we detected when we trained the model with the original architecture. The final network structure is shown below.

Customized AlexNet Model Pipeline



For this method, we specifically focus on reducing overfitting. We experimented with different combinations of 4 regularization techniques and their associated hyperparameter choices, which are: (1) L2 regularization on convolutional layers and fully-connected layers with 3 different scale values (scale = 0.1, 0.01, 0.001), (2) dropout with 4 different keep probabilities (keep-prob = 0.3,0.4,0.5,0.6), (3) smaller batch size (batch-size = 16, 32) and (4) image augmentation.

4.3 VGG16 Model with Pre-trained Weights

We performed transfer learning with VGG16 with two sets of pre-trained weights, one from the object detection task on the ImageNet dataset and the other from the scene classification task on the Places365 dataset. We conducted 4 experiments, which are: (1) retrain the last fully-connected layer only (4) retrain all the fully-connected layers (3) retrain all the fully-connected layers + the fifth block of convolutional layers and (4) retrain all the fully-connected layers + the fourth and the fifth block of convolutional layers.

5 Results

5.1 Performance Comparison Across Models

After performing experiments described in Section 4, we obtained the best model for each of the three methods. The "best model" here is defined as the model with the highest overall accuracy $((i.e. \frac{\text{number of correct classifications}}{\text{number of test samples}})$.

(1) The best baseline model: achieved when using dropout with keep-probability = 0.5 and learning-rate = 1e-3. (2) The best customized AlexNet model: achieved when using dropout with keep-probability = 0.5, L2 regularization on all the convolutional layers and fully-connected layers with scale value = 0.001, batch-size = 32, learning-rate = 1e-4, and without extra photos from image augmentation. (3) The best VGG model:

achieved when using ImageNet pre-trained weights for the first three blocks of convolutional layers and fine-tuning the fourth and fifth block of convolutional layers with learning-rate = 1e-4.

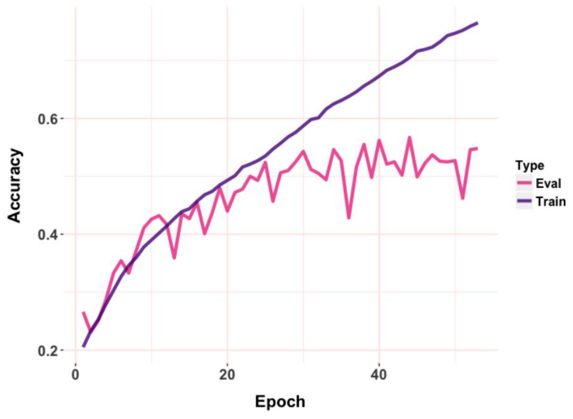
We compared the three models based on their overall classification accuracy on the evaluation set. The best customized AlexNet model achieved a overall classification accuracy of 56.7% in this 9-city classification problem. The detailed training and evaluation accuracy of each model is shown below.

Model	Train Accuracy	Evaluation Accuracy
Best Baseline	82.7%	50.4%
Best AlexNet	76.5%	56.7%
Best VGG	96.2%	42.7%

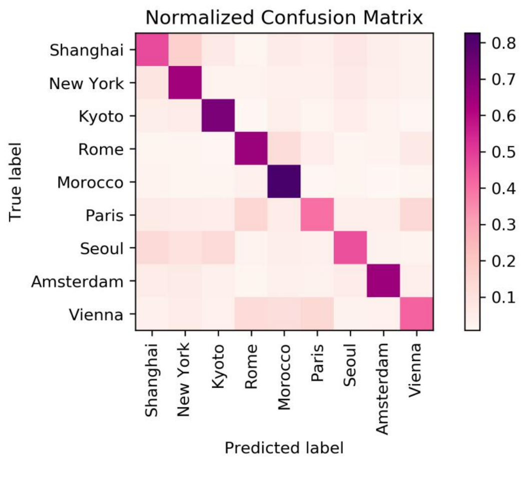
Table 2: Model Performance

5.2 Analysis of the Best-Performance Model

The training and the evaluation accuracy curve for our best-performance model is shown below in Figure1(a). We can see that after Epoch = 30, the training accuracy of our model continues to climb up, while the evaluation accuracy fluctuates around 55%. This means that even after implementing both dropout and L2 regularization, as well as experimenting with a variety of hyperparameter choices on keep probability, L2 scale value and batch size, we still have not completely solved the overfitting issue. This situation suggests one possible future improvement for our project. We also created a confusion matrix, shown below in Figure1(b), to examine the classification performance of our model on each class. Each cell in the confusion matrix is the percentage of the images from the row city that are classified as the column city, and a darker color corresponds to a higher percentage. The dark color along diagonal is good sign, which indicates that for each city, we perform most of the classification correctly. The off-diagonal elements (which correspond to incorrect classifications) with dark colors are what we are interested in further investigation for model improvements. Based on the confusion matrix, specifically, we observe that our current model experiences difficulties with distinguishing between the followings: (1) Shanghai's and New York's architectures (modernized buildings), (2) Paris's, Rome's, and Vienna's architectures (typical European architecture styles), (3) Seoul's, Shanghai's, and Kyoto's architectures (temples and palaces) and (4) Vienna's architectures (frequently incorrectly classified as Rome's, Morocco's, or Paris's)



(a) Train and Eval Accuracy for the Best AlexNet Model

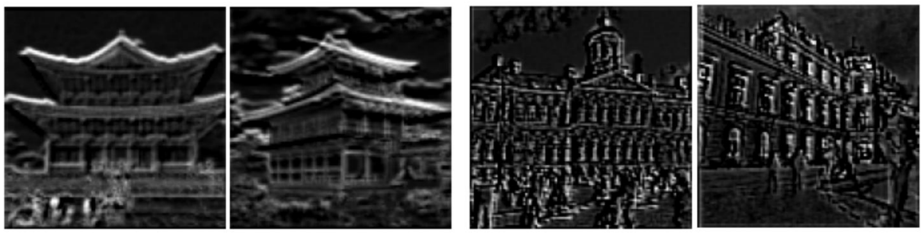


(b) Confusion Matrix

Figure 1: Result

5.3 Analysis of Transfer Learning

To better understand and interpret the performance of transfer learning using pre-trained VGG16 weights, we visualized all the ConvNets layers [13]. The figure below presents the visualization of the second convolution layer in the second block of the VGG16 architecture using pre-trained ImageNet weights. Both (a) and (b) group of images look very similar but actually they belong to different cities. This effect is caused by the fact that VGG16 weights were trained to learn broad object categories, such as temples and palaces, while our task focuses on identifying subcategories of architectures for different cities. More subtle architectural and geographical clues are required for distinguishing between cities. According to layer visualization result, we notice that VGG16 is capable of learning the architectural style but fails to distinguish between similar ones and hence incorrectly geolocates the photo. For instance, the architectures in the following images all belong to a broad category named temples and palaces; this ambiguity will lead to misclassification since a certain type of architecture might occur in multiple cities. This shortcoming of network may be mitigated if additional cultural and environmental-related information are provided so that it could reach better performance when distinguishing plausible cities.



(a) Seoul and Kyoto (b) Amsterdam and Vienna

Figure 2: Layer Visualization of VGG16 - Transfer Learning

6 Conclusion and Future Work

6.1 Conclusion

We implemented three convolutional neural networks to detect architectural styles of the buildings captured in photos and utilized extracted features to perform photo geolocation task. We experimented with both models training from scratch and fine-tuneing well-known CNN models. The former approach provides us with a better classification accuracy. Our customized AlexNet trained from scratch with dropout and L2 regularization achieves 56.7% accuracy on the 9-city classification task. The failure of transfer learning in this case is partially due to the fact that the pre-trained weights used were obtained from classification tasks with very broad categories, so that they are not suitable for our problem which focus on recognizing locally typical architecture styles.

6.2 Future Work

One future direction for our project is to further mitigate the overfitting problem in our customized AlexNet model. Two options we are considering are adding more training data and further reducing network architectures' complexity. We plan to experiment with different simplified versions of our customized AlexNet by removing certain layers, and at the same time collect more photos for each city and try a different series of data augmentation procedures. In addition, we are interested in retraining the entire VGG16 model to see if we can achieve a better performance without using the pre-trained weights. Lastly, we consider increasing the number of cities to classify to perfect our model in better addressing the real world problem of photo geolocation.

7 Contributions

Each member worked on collecting image dataset for 3 cities, implementing the baseline and the AlexNet model in TensorFlow [14], as well as applying and fine-tunning the pre-trained VGG16 model. In addition, Yancheng Li took responsibility for setting up AWS, performing the hyperparamter search for the AlexNet model and visualizing VGG16 layers. Yao Chen worked on the regularization implementation for the baseline model and visualized results for the writeup. Yifan Yu was responsible for converting Places365 AlexNet and VGG16 caffe models to tensorflow models, implementing data augmentation, and implementing regularization for the AlexNet model.

8 Link to Code Repository

`https://github.com/EmmaYChen/CS230_Project`

References

[1] Flickr dataset. `https://www.flickr.com/`.

[2] Amir R Zamir, Asaad Hakeem, Luc Van Gool, Mubarak Shah, and Richard Szeliski. Introduction to large-scale visual geo-localization. In *Large-Scale Visual Geo-Localization*, pages 1–18. Springer, 2016.

[3] Yannis Avrithis, Yannis Kalantidis, Giorgos Tolias, and Evaggelos Spyrou. Retrieving landmark and non-landmark images from community photo collections. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 153–162. ACM, 2010.

[4] Stephan Gammeter, Lukas Bossard, Till Quack, and Luc Van Gool. I know what you did last summer: object-level auto-annotation of holiday snaps. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 614–621. IEEE, 2009.

[5] Grant Schindler, Matthew Brown, and Richard Szeliski. City-scale location recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–7. IEEE, 2007.

[6] Alessandro Bergamo, Sudipta N Sinha, and Lorenzo Torresani. Leveraging structure from motion to learn discriminative codebooks for scalable landmark classification. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 763–770. IEEE, 2013.

[7] Petr Gronat, Guillaume Obozinski, Josef Sivic, and Tomá Pajdla. Learning and calibrating per-location classifiers for visual place recognition. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 907–914. IEEE, 2013.

[8] Tobias Weyand, Ilya Kostrikov, and James Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016.

[9] CS 230 teaching team. `https://github.com/cs230-stanford/cs230-codeexamples/tree/master/tensorflow/vision`, 2018.

[10] Khosla A Oliva A Torralba A. Zhou B, Lapedriza A. Places: A 10 million image database for scene recognition. *IEEE Trans Pattern Anal Mach Intell.*, 2017.

[11] Ethereon Russell91. Caffe to tensorflow, 2015.

[12] Local response normalization in convolutional neural networks. `https://prateekvjoshi.com/2016/04/05/what-is-local-response-normalization-in-convolutional-neural-networks/`.

[13] Interactive convnet features visualization for keras. `https://keplr-io.github.io/quiver/`.

[14] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.