Chinese Landmark & Attraction Recognition

Hao Wu

A Capstone Project Report
submitted in partial fulfillment of the
requirements of the degree of

Master of Science in Computer Science & Software Engineering

University of Washington

2018

Project Committee:

Dr. Min Chen, Committee Chair

Dr. Dong Si, Committee Member

Dr. Yang Peng, Committee Member

Abstract

With the development of deep learning algorithm and computing hardware, image classification technology has been remarkably improved over the past years. To continue advancing the state-of-the-art in computer vision, many researchers are now putting more focus on fine-grained recognition problems. For example, instead of recognizing general entities such as buildings, lakes and mountains, researchers are developing machine learning algorithms capable of identifying Seattle Public Library, Lake Washington or Mount Rainer. Landmark recognition is a useful yet challenging task, due to the lack of large annotated datasets. Previous related work neglects non-English speaking area since they collect data using English only. The commercial landmark recognition APIs test also indicates the incompleteness of Chinese landmark data. This project releases Chinese landmark & attraction dataset for recognition of human-made and natural landmarks in China. The dataset contains 42,548 images depicting 987 unique landmarks in China. Moreover, we build a landmark recognition model with two Convolution Neural Network representations (Residual Network and VGG) and presents the results. Lastly, an iOS application is built based on our landmark recognition model. It allows user to do landmark prediction on images taken by camera and images from device gallery.

Contents

List of Tables

List of Figures

1. Introduction

1.1 Problem Description

When we travel in a new place or when we go through our vacation photos after the trip, we may ask ourselves: what is the name of this temple? This building looks so familiar, where did I take this picture? Landmark recognition can help! Landmark recognition, as a sub-domain of fine-grained recognition, aims to automatically discriminate landmark categories with subtle visual differences. It can predict landmark labels directly from image pixels to help people better understand and organize their photo collections. However, there is no such applications available. We may use image search engine such as Google Image, but there are limitations for using Google Image search. First, we need to upload our pictures to the search engine, which needs Internet access. Second, it is not designed for landmark recognition task, so it fails to recognize some landmarks, especially not-so-famous ones. Take my homeland, China, for example, which is the world's number one tourist destination [44]. People are fascinated with the different culture, the natural highlights, ancient cities, and fast modernization in China. But, if we test Google Image with some less popular landmarks in China, we may get unsatisfied results. Our commercial landmark recognition APIs review in Section 3 also indicates that there is lack of sufficient data for Chinese landmarks and attractions in their datasets.

Google released Google-Landmarks in March 2018, which is the largest worldwide landmark dataset, but this dataset only provides a unique ID for each landmark without other much needed annotations. The authors also mentioned there are region bias in their database [11]. First, they only use English when they explore the Internet for landmarks images. Second, the most users of their data source such as picasa.google.com and panoramio.com, are located in Europe and North America.

Therefore, build a Chinese landmark & attraction dataset is a complement for current work, consequently we can deploy our own landmark recognition model and application based on this new dataset.

1.2 Challenges

Currently, the biggest challenge of landmark recognition research is lack of sufficient annotated datasets. Compared with ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [4], landmark recognition task has similar large number of categories to be recognized, yet with much less images for each category. ILSVRC evaluates algorithms for object detection and image classification across a wider variety of objects (e.g. amphibian, bird, person). More specifically, the training and evaluation data for the classification and localization tasks for ILSVRC2017 consist of 1.35 million photographers of 1,000 object categories. On average, there are 1,350 images for each category. However, for Google-Landmarks [10], the largest worldwide landmark datasets released by Google in March 2018, the average number of images in each category is only around 70. Google-Landmarks contains two million images depicting thirty thousand unique landmarks from across the world. Even with such a large dataset, there might not be much training data available for less popular landmarks. From our analysis to their released dataset on Kaggle [45], there are 7,096 landmarks contain less than or equal to ten images in their training set.

Furthermore, there is not enough data for Chinese landmarks in Google Landmarks because of region and language bias we mentioned in section 1, most of the landmarks are located in Europe and North America. Moreover, from the screenshot of their dataset in Figure 1, for each instance, they only provide URL of the image with a landmark id, it takes time and effort to retrieval Chinese landmarks from their dataset. Therefore, we want to build a more comprehensive Chinese landmark & attraction dataset from scratch.

	A	B	C	D
1	id	url	landmark_id	
2	51260f07e042f2fc	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/100904585.jpg	10073	
3	3842b07b9ea17c6c	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/10197183.jpg	4562	
4	d921c928e4e235a3	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/10375176.jpg	7505	
5	c194134cf2964e7f	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/10528666.jpg	8023	
6	81819a3aef4d4a69	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/10532774.jpg	10958	
7	4acaed75f511ce05	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/10538096.jpg	2228	
8	a50a4e2b04c19b56	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/10752496.jpg	4093	
9	be09d7d2ddaf085d	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/11259619.jpg	14619	
10	03d6f7378fa81f12	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/11264223.jpg	14533	
11	ca607636daf70311	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/11568265.jpg	14007	
12	a3e7a46b4a88ee1d	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/11677078.jpg	4045	
13	7b93bea122d6c9f6	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/11712243.jpg	4571	
14	5afd2be0b51444e0	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/11839471.jpg	10820	
15	650386a22a33d1a7	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/11849064.jpg	6197	
16	4e04f30413f9a439	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/11980574.jpg	11925	
17	8820679e28ccc528	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/11980585.jpg	11925	
18	e8704c900253b5c5	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/12088103.jpg	3899	
19	9a32c795f814bb2d	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/12126232.jpg	4867	
20	dbb4a4446ae609a4	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/12324476.jpg	13104	
21	0eaa8056b322b4c6	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/1254772.jpg	10855	
22	da66b8ebac344836	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/1256207.jpg	2330	
23	11be513ea5ab9f73	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/1256444.jpg	4313	
24	3189091687cf7839	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/12631338.jpg	9245	
25	6f1a91687d0b0199	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/12665083.jpg	8063	
26	7265d045914b890a	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/12771087.jpg	4685	
27	d5f16691900f6f76	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/12771437.jpg	4685	
28	a144a6d28518ef54	http://commondatastorage.googleapis.com/static.panoramio.com/photos/original/13152648.jpg	11	

Figure 1 Screenshot of Google-Landmarks

1.3 Project Objectives

The objective of this project is to:

(1) Develop Chinese landmark & attraction dataset from scratch. The new dataset is a complement to current data and can benefit research for other image recognition problems.

(2) Build a Chinese landmarks recognition model based on advanced deep learning technologies and integrate it to mobile applications.

To build a comprehensive and annotated Chinese landmark & attraction dataset, we leverage the vast amount of multimedia data on the web, utilize image clustering and retrieval techniques. The landmark recognition model is built on existing research concepts that are based on deep learning and transfer learning. Previous research papers show image classification performance is improved when transfer learning is applied to middle-scale datasets with less or equal to 200 categories (i.e. Pascal VOC [18] with 20 categories, Oxford Flowers with 102 categories, Caltech-UCSD Birds [20] with 200 categories) [22,23]. However, they do not discuss about the datasets that contain large number of categories. In this project, we develop landmark recognition models with our own dataset, which consists of 987 unique landmarks. By taking advantage of feature extractors derived

from ImageNet, our model delivers satisfactory results. Additionally, we successfully integrate the landmark recognition model into mobile application.

1.4 Contribution and Beneficiaries

The contributions of this project are: (1) We build Chinese landmark & attraction dataset to complement the current landmark dataset. (2) We apply Unique approaches that combine multiple technologies from data collection, data preparation and analysis to obtain comprehensive and validated dataset so as to build a reliable landmark recognition model. (3) We explore a new experiment for applying deep learning and transfer learning on dataset with large number of categories.

The benefits of the project are as follows: (1) Our project helps people recognize Chinese landmarks and attractions during their trips and better understand and organize their photo gallery after trips. (2) Our dataset benefits other researchers by providing a new landmark dataset. (3) We also validate the performance for learning and transferring image representations using Convolutional Neural Networks.

1.5 Report Outline

The rest of the paper is structured as follows:

Section 2 covers literature review including related work on dataset and algorithm. Section 3 describes the project workflow and introduces each step with proposed research methods, including data collection, data preparation, dataset analysis, model training, model evaluation and model integration to mobile application. Section 4 presents implementation detail. Section 5 presents results. Finally, section 6 summarizes conclusions and explores future work.

2 Literature Review

2.1 Dataset

Landmark recognition, as one sub-domain of computer vision, detects popular natural and man-made structures within an image. The biggest challenge of landmark recognition research is the lack of large annotated dataset [10]. Zheng and Zhao [11] develop a landmark recognition engine incorporating 5,312 landmarks from 1,259 cities in 144 countries, among them, most landmarks are located in Europe and North America. They attribute this distribution bias to the user community of their data source. Only 101 landmarks in China are included in this database. Noh and Araujo [12] propose a local feature descriptor suitable for large-scale image retrieval. Based on Zheng and Zhao's work, they create Google Landmarks, which contains ~30K landmarks from 4,872 cities in 187 countries. They do not give detailed distribution information about the new dataset, bur from figure 2, we can tell that most landmarks are located in Europe and North America as well.

Figure 2 Geolocation distribution Google-Landmarks

2.2 Algorithm

Modern machine learning and deep learning techniques have improved the state-of-the-art in computer vision. In 2012, Hubel and Ciresan etc. [1, 2] demonstrated extremely low error rate and

fast learning process on MNIST and NORB database using Convolutional Neural Network (CNN). MINIST is a large database of handwritten digits. NORB contains fifty low-resolution grayscale images of five generic categories objects. At the same year, Krizhevsky etc. [3] trained a large, deep CNN model in the ImageNet LSVRC 2012 [4] contest and achieved a winning top 5 test error rate. After that, deep learning techniques have enabled rapid progress in the ImageNet competition, even surpassing human performance. In 2014, Simonyan etc. [5]'s VGG model further extended the ideas of CNN with less parameters and introduced more non-linearity into the model. In 2015, He and Zhang's team [7] introduced skip connections to the model to avoid degradation problem while using deeper layers. Their ResNet architecture was the first to pass human level performance on ImageNet.

In computer vision area, the touristic landmarks have interested many researchers. To mine a clean set of landmark images, Li et al. [6], Avrithis and Kalantidis [19] employ the community photo collections, by analyzing the geometric, visual, geographical and textual cues. [11] shares the similar methods to explore landmarks at a world-scale. However, these works were done before 2010, while the common use of CNN in computer vision area began from 2012. After that, [10] use a CNN based model for large-scale image retrieval and introduce Google-Landmarks dataset. Contrasting to [6], [19], [11], the principal focus of our approach is to explore landmarks at China. During the process of building Chinese landmark & attraction dataset, we take advantage of our data source, which has labeled images for landmarks. We also apply CNN and transfer learning to do visual clustering and image cropping. Contrasting to [10], we build a landmark classification system, which can easily integrate to a local mobile application without the existence of the whole image datasets.

3 Methods

In this section, we describe the overall workflow of our project, the algorithms, test of current landmark recognition APIs, each step of the project, including developing Chinese landmark & attraction dataset: data pre-processing and data preparation, model testing and evaluation, model integration with mobile application and testing methodology. We also provide tools and frameworks used for landmark recognition system.

3.1 Project Workflow

Figure 3 presents the workflow of the project.

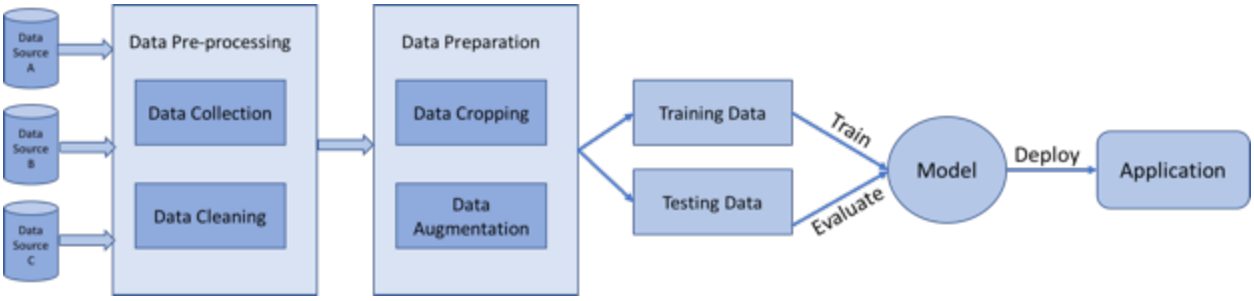


Figure 3 Project workflow

3.2 Algorithms

This section covers the high-level algorithms we use for our project. Section 3.4 and 3.5 cover more methods detail. In general, we apply deep learning with CNN architectures and transfer learning when we build the dataset as well as training the landmark recognition model. There are two types of machine learning technologies we use, and the last part of this section covers transfer learning. Implementation detail will be discussed in Section 4.

3.2.1 Supervised Machine Learning

Supervised learning refers to algorithms guided by pre-existing patterns and feedback based on known outcomes. In practice, supervised learning works by showing data to the machine including the correct value of the data. The machine then applies a supervised learning algorithm to decipher patterns that exist in the data and develops a model that can reproduce matching results with new data. Having sufficient data that is representative of variations is very important for supervised

machine learning. Moreover, the data should be relevant and without bias if taken from a larger dataset. Some common supervised learning algorithms include regression analysis, decision trees, *k*-nearest neighbors, neural networks, support vector machine, etc [43].

Our project, landmark recognition is an image classification problem, which is considered an instance of supervised learning. Currently, neural networks, especially CNNs get the best results in most of the public visual recognition challenges [4]. An illustration of capabilities of CNN architectures is given by the ILSVRC [4], which is a benchmark in object classification and detection. Humans tend to have trouble with classifying objects into fine-grained classes, such as the particular species of flower or breed of dog, whereas CNN handle this with ease. In our project, we use CNN to handle landmark recognition problem.

3.2.2 Unsupervised Machine Learning

In the case of an unsupervised learning environment, there are no such known patterns from which to base the analysis on. Instead, the model must uncover hidden patters through the use of unsupervised learning algorithms. Clustering is one common task for unsupervised machine learning. We conduct image clustering in dataset pre-processing step to get images of landmark from the noisy raw data. In general, most research works on image clustering are based on feature encoding, which can largely reduce the dimensionality of image features so as to make large-scale clustering possible. However, reducing the dimensionality of features is equivalent to decreasing the representational power, leading to unsatisfactory closeting performance [50]. CNN architectures have proven to be able to learn significantly better discriminative visual representations for images compared to traditional hand-crafted features or features learned by shallow neural networks, given a sufficiently large labeled training set [50]. Thus, we also utilize CNNs to do image clustering.

3.2.3 Transfer Learning

The CNN architecture usually contains millions of parameters. Directly learning so many parameters from thousands of training images is problematic. The key idea of transfer learning is that the internal layer of the CNNs can act as a generic extractor of middle-level image representation, which can be pretrained on one dataset, such as ImageNet, and then transferred on

other tasks. [21, 22] shows the outstanding performance of the transferred representation when applied to object and action classification tasks.

Considering the size of our dataset and the state-of-the-art performance of CNNs, we deploy our model by transferring image representation learned from ImageNet to our landmark recognition task. We replace the last fully-connected layer (this layer's outputs are the 1000 class scores for ImageNet) with a new linear classifier and train the whole model with our own data. For image clustering task, we also use pretrained CNN representation but without the last fully-connected layer as fixed feature extractor to get "fingerprint" for each image in our raw data. Then cluster the fingerprints of images for each landmark by certain metric of similarity, thus we separate the images with landmarks and noises. Details will be explained in following sections.

3.3 Landmark Recognition APIs Testing

There are three ways to do landmark recognition.

(1) Landmark recognition APIs, such as Google Cloud Vision API [8] and Microsoft Computer Vision [9]. They enable users to understand the content of an image by encapsulating machine learning models in a ready-to-use REST API. They also provide website user interface to use the API. We cannot modify the machine learning algorithm or the data.
(2) Machine learning platform, such as Google Cloud AutoML and Microsoft Custom Vision. They offer user interface and allow user to train custom machine learning models with their own dataset. We can feed our own data but cannot modify the machine learning algorithm.
(3) Open source machine learning libraries, such as Tensorflow, Keras, etc. They are more flexible and allow users to deploy machine learning model from scratch. We can try different algorithms with our own data.

Table 1 shows the differences of these three options. Considering the flexibility and cost, we decide to build our model using open source libraries.

	UI	Flexibility	Fee	Mobile
API	Yes	Low – Pretrained model	Middle - API call	Yes
ML platform	Yes	Middle – Own data	High - Upload, train, predict	Yes
Library	No	High – From scratch	Low - GPU for training	Yes

Table 1 Comparison of landmark recognition methods

We test Google Cloud Vision and Microsoft Computer Vision. They both offer pretrained landmark recognition models, which are based on CNNs. For 406 test images with Chinese landmarks, Google Cloud Vision recognizes 145 (35.7%) images correctly and has 6 false positive, while Microsoft Computer Vision recognizes 48 (11.8%) images correctly and has 4 false positive. Figure 4 shows sample of the test images and according results. The tag on up left of each image shows whether it is recognized by M= Microsoft Computer Vision or G = Google Cloud Vision. Images with no tag mean both models fail to recognize them. Microsoft Computer Vision recognizes relatively less landmarks. Neither Microsoft or Google are not able to recognize known landmarks with some degree of variations. The API review results indicate the lack of Chinese landmark data of current landmark dataset in their models.

Figure 4 Landmark APIs testing images and results

3.4 Dataset

In machine learning context, building datasets is one of the most time-consuming and crucial processes which involves many different tasks and cannot be fully automated [47]. In our project, it includes collecting data form data source, pre-processing and preparing the data into a form suitable for further analysis and processing. The following sections discuss research methods for each step.

3.4.1 Data Source

The explosion of personal digital photography, together with Internet, has led to the phenomenal growth of landmark photo sharing in many websites. To get a comprehensive and organized Chinsed landmark & attraction dataset, we explore the following sources on the Interne: (1) China travel service provides, Ctrip [13], which is currently the largest online travel agency in China. (2) TripAdvisor [14], which is an American travel and restaurant website company. Both of them are

free to users, who provide contents such as comments, image reviews and ratings for landmarks on the websites. These travel websites provide popular landmark lists and plenty of potential landmarks images for our project. (3) Search engines, such as Google, Baidu and Google Image [15,16]. We use them to complete our landmark lists and image dataset.

3.4.2 Data Pre-processing

Data pre-processing refers to collecting data from data source and cleaning the raw data for further use.

3.4.2.1 Data Collection

By web scraping of the data sources, we got a list of popular landmarks in China, together with a dataset of potential landmarks images. There are several ways to extract information from the web. (1) Use of API provided by the websites allows us get access to structured data from the provider. (2) There is no API offered by Ctrip and TripAdvisor, so we use open source libraries, such as Beautiful Soup and Puppeteer to scrape the website to transform the unstructured data (e.g. HTML format) into structured data (e.g. a CSV file). Finally, we got ~500K potential images for 3,359 landmarks.

3.4.2.2 Data Cleaning

The raw image dataset contains lots of noise such as unrelated images, mislabeled images and low-resolution images. We need to do data cleaning to get rid of noise as much as possible.

By using image clustering algorithm and help functions, we want to: (1) Group the images under each landmark folder in such a way that images in the same group are more similar to each other than to those in other groups. (2) Keep the groups which the objects in the images are belong to the landmark. (3) Discard unrelated groups such as images of food, people or images with inaccurate labels.

The first task is done by unsupervised machine learning with CNN, details will be covered in next paragraphs. The second and third tasks are done with help functions involve human decisions. At first, we consider about applying strategies, such as keep the group with most images, to help us

to complete all three steps automatically. However, clustering results are different from each landmark. For some landmarks, the largest group contains unrelated objects such as people, plants or food instead of landmarks. There are also groups contain less images, but these images are actually what we want. These pictures are landmarks taken from different point of view, or under different weather condition, so the algorithm cluster these similar images to sperate groups. In order to get relevant data as much as possible, we use help functions to iterate each group and make decisions (keep, or discard) based on the contents of the group.

For the first task, there are two aspects we need to think in order to get a satisficed image clustering results. (1) How to present the images? (2) How to cluster the images?

How to present the images? In Computer Science, data are represented by a fixed number of features which can be binary, categorical or continuous. Things are similar when referring to images. How we present the images to our clustering algorithm decides the performance of the results. Guerin [35] uses a CNN pretrained model on ImageNet to do image clustering and achieves state-of-the-art performance. It proves that supervised training of deep neural network on large datasets, with a large variability of classes, extracts better features than most carefully designed engineering approaches.

How to cluster the images? Image clustering performs unsupervised learning, aims to find a mapping of the archive images into clusters such that the set of clusters provide essentially the same information about the image archive as the entire image-set collection. There are several algorithms for clustering. They can be categorized into two groups: hierarchical and partitional clustering [33, 34]. The biggest difference between hierarchical and partitional clustering is whether the set of clusters is nested.

> A partitional clustering is a division of the set of data objects into non-overlapping subsets such that each data object is in exactly one subset. The most widely used partitional algorithm is k-means clustering.
>
> A hierarchical clustering is a set of nested clusters that are organized as a tree. The algorithm begins by assigning each sample to its own cluster. At each step, the two clusters

that are the most similar are merged; the algorithm continues until all of the clusters have been merged.

The k-means algorithm makes a number of assumptions about the data. The most notable assumption is that the data is well separated into sphere-like clusters. We also need to decide a k value to well suit the shape of the data. In contrast, hierarchical clustering is more flexible and has fewer assumptions about the data distribution, the only parameter (which k-means also shares) is that a distance metric can be calculated each pair of data points.

Taking above into consideration, we use a CNN architecture to hierarchical cluster Chinese landmark & attraction dataset. Take a CNN pretrained on ImageNet, remove the last fully-connected layer (this layer's output are the 1000 class scores for ImageNet), then treat the rest of the network as a fixed feature extractor for our dataset. In a VGG19 [5], this would compute a 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier. The algorithm can be summarized as follows:

1) Set VGG19 as our base mode, set the weights to ImageNet, remove the top layer.
2) For each landmark image list in the dataset:

   For each image in the landmark list:

   i. Use the VGG19 model to extract the image fingerprint as an array of 4096 values.
   ii. Save the image as the key and the image fingerprint as the value to a dictionary.

   Apply hierarchical clustering to the dictionary values:

   i. At the start, treat each point as one cluster. Therefore, the number of clusters at the start will be N, while N is an integer representing the number of data points.
   ii. Form a cluster by joining the two closet data points resulting in N-1 clusters. The Euclidean distance is used to find distance between the clusters.
   iii. Form more clusters by joining the two closet clusters resulting in N-2 clusters.

iv. Repeat the above three steps until one big cluster is formed.

v. Once single cluster is formed, dendrograms are used to divide into multiple clusters depending upon the similarity index value we provide as a parameter.

After data cleaning, we have 48,732 images for 1059 unique landmarks.

3.4.3 Data Preparation

After data cleaning, we get datasets mainly contain landmarks. In our project, we deploy our landmark recognition model for the first iteration. We train a CNN model using the 1059 landmarks dataset. However, our model encounters underfitting issue as it has poor training and validation performance. Common causes of underfitting include insufficient training data to adequately cover all possible combinations, and situations where the training and test data were not properly randomized. Thus, before we modify our model's hyperparameters, we go back to retouch our dataset. We find there are images contain the objects of interests at various scales with complex background. We further apply certain data preparation techniques to transform data sets so that their information content is best exposed to the deep learning model. In our project, we apply image cropping and data augmentation techniques.

3.4.3.1 Image Cropping

For image cropping, we want the object of interest, the landmark, to be in the center of the image since the complex background would inevitably affect recognition performance when training and testing across different domains [48]. In order to do image cropping, we have two options.

The first option is object detection. The goal of object detection is to detect multiple instances of object from a known class, such as people, buildings or faces in an image. Figure 5 shows sample object detection results. Ideally, by applying object detection algorithm to the landmark images, we want to get objects such as buildings, mountains or people detected within bounding boxes, then we can crop the wanted boxes (buildings or mountains) and discard irrelevant objects (people). Google offers TensorFlow Object Detection API as well as pretrained object detection models [36]. Figure 6 shows the API detection results on our landmark images. It turns out the pretrained models

provided by Google do not fit our needs. It is good at recognizing objects such as person, flags, vehicles, umbrellas, etc., but fails to detect temples, buildings or mountains. The datasets they use do not have enough landmark related categories. We certainly can train a model with new data, but it takes time and effort. Therefore, we use the second technical to help us.

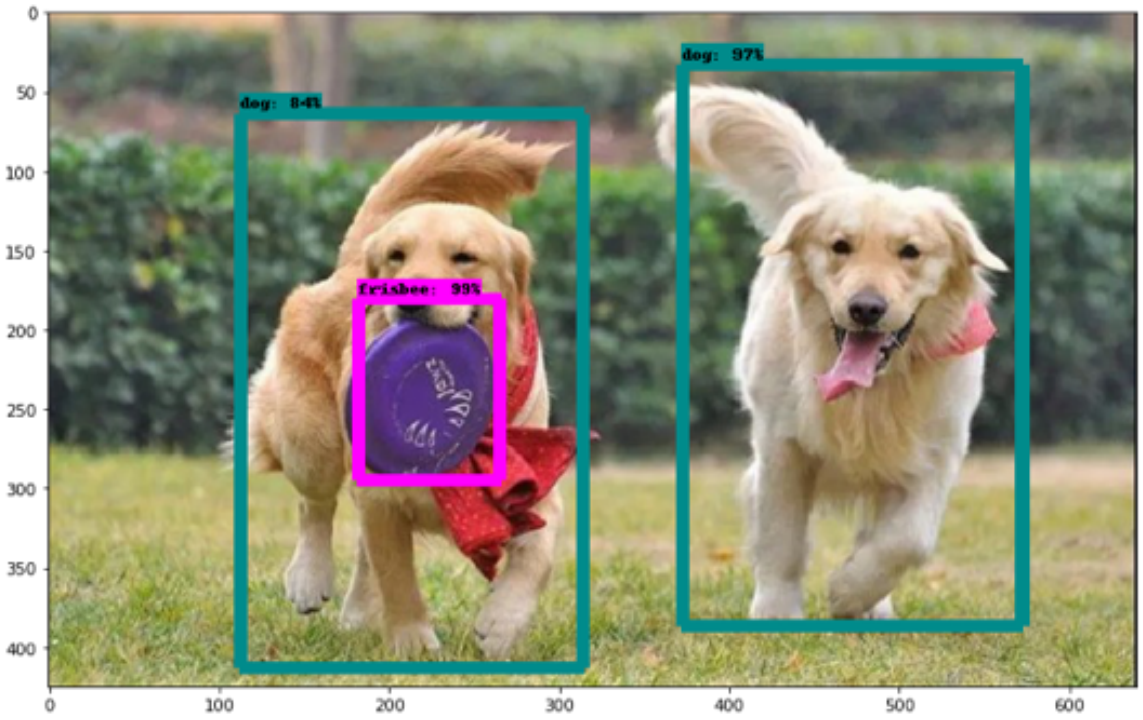


Figure 5 Object detection

Figure 6 Object detection on landmark images

The second approach is to do content-based image retrieval. Content-based means that the search analyzes the contents of the images rather than the metadata used by traditional concept-based approaches. [12] proposed a new feature descriptor called DEep Local Feature(DELF). It is a CNN-based local feature with attention, which is trained with weak supervision using image-level class labels only, without the need of object- and patch-level annotations. Figure 7 illustrates the overall procedure of feature extraction and image retrieval.

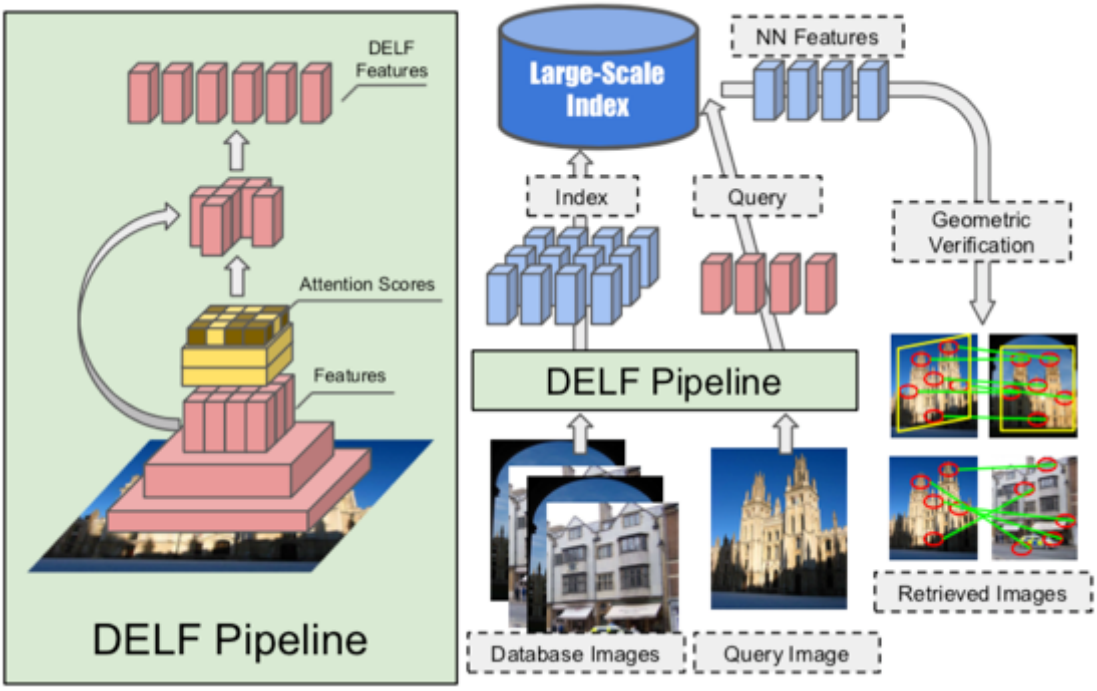


Figure 7 Overall architecture of DELF image retrieval system [12]

By providing an image with clear view of the landmark as a mask, we use DELF to identify semantically useful local feature among images. Then we got a bounding box of the landmark and cropped the landmark from the image. Figure 8 shows pairs of images with same landmark.

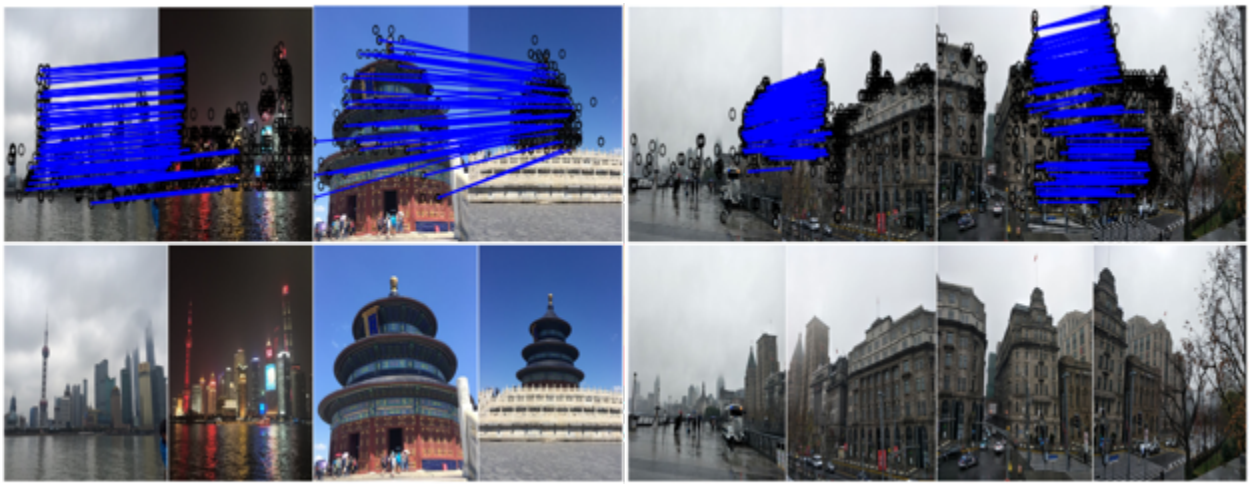


Figure 8 Visualization of feature correspondences between images using DELF
There are four groups of images, each group has two images contain the same landmark. The 2nd row shows the original images. And the 1st row presents the visualization results after using DELF. The blue lines indicate the same landmark DELF locates.

After applying image cropping, the dataset contains 42,548 images depicting 987 unique landmarks in China.

3.4.3.2 Image Augmentation

High-quality data is the key to deep learning models. One way to get around lack of data is to do data augmentation. Reasonable approaches to programmatic data augmentation can increase the amount of relevant data in our dataset. Currently we use techniques include flip, rotate and scale the original images. Figure 9 shows the original image and according results.

In our final dataset we have unbalanced dataset, 6 of 987 landmarks contain eight times more than the average category images. We keep these six categories untouched and apply data augmentation to rest categories.

Figure 9 Data augmentation
From the left, we have original image, flipped image, rotated image, and scaled image

3.5 Modal Training & Evaluation

3.5.1.1 Model

We mentioned before that CNNs achieve outstanding performance for image classification tasks, so we adopt CNNs together with transfer learning to deploy our landmark recognition model. Neural networks are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. Based on neural networks, convolutional neural network architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the number of parameters in the network. A simple convolutional neural network for image classification could have the architecture as below:

- INPUT [224x224x3] will hold the raw pixel values of the image, in this case an image of width 224, height 224, and with three color channels R, G, B.
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [224x224x12] if we decided to use 12 filters.
- RELU layer will apply an elementwise activation function, such as the max (0, x) thresholding at zero. This leaves the size of the volume unchanged ([224x224x12]).
- POOL layer will perform a down sampling operation along the spatial dimensions (width, height), resulting in volume such as [112x112x12].
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of landmark objects. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

For CNN architectures, the output from the convolutional layers represents high-level features in the data. While that output could be flattened and connected to the output layer, a fully-connected layer is widely used of learning non-linear combinations of these features, to transfer the feature representations to classification results. In a CNN model pretrained with ImageNet, the last fully-connected layer's outputs are the 1000 class scores for ImageNet dataset. When we apply transfer learning, we retrain the model to recognize new landmark objects, so instead of a fully-connected layer, we need to replace it with other classifiers. In practice, for classification tasks, most of deep learning models employ the SoftMax activation function at the top [51]. In a SoftMax classifier, the function mapping from previous layer as the input is:

$$f(x_i; W) = Wx_i$$

Where W is the parameter, x is the image input. We have probabilistic interpretation for the classification results as:

$$P(y_i|x_i; W) = \frac{e^{f_{yi}}}{\sum_j e^{f_j}}$$

The above representation can be interpreted as the normalized probability assigned to the correct label y_i given the image x_i and parameterized by W. For example, given 10 possible classes, the SoftMax layer has 10 nodes denoted by p_i, where $i = 1, \dots, 10$. p_i specifies a discrete probability distribution, therefore, $\sum_{i=1}^{10} p_i = 1$.

There are couple of CNN representations we can choose from, we choose two up-to-date pretrained networks, ResNet and VGG, to train and evaluate our recognition mode. Figure 10 shows VGG19 and Resnet34 network architectures for ImageNet. VGG network was introduced in 2014, it was characterized by the fact that it uses a simple 3 x 3 convolutional layer stack, on top of each other in increasing depth. At that time, a 19-layered neural network was considered very deep. In general, increasing the depth should increase the accuracy of neural network, but an obstacle of deeper neural network is the problem of vanish/exploding gradients. In 2015, Resnet introduced skip connections to the network to avoid degradation problem and at the same time deployed an even deeper neural network. Figure 11 and 12 show the VGG19 and Resnet50 model summary separately. In figure 12, we skip similar middle layers of ResNet50 model to save some space.

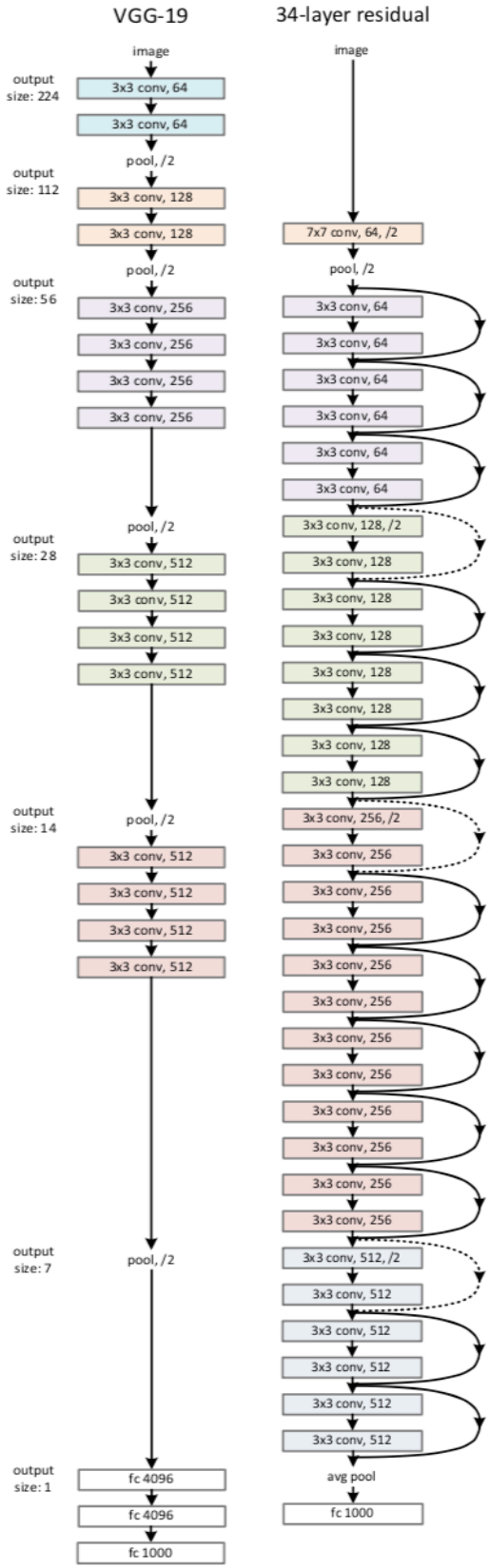
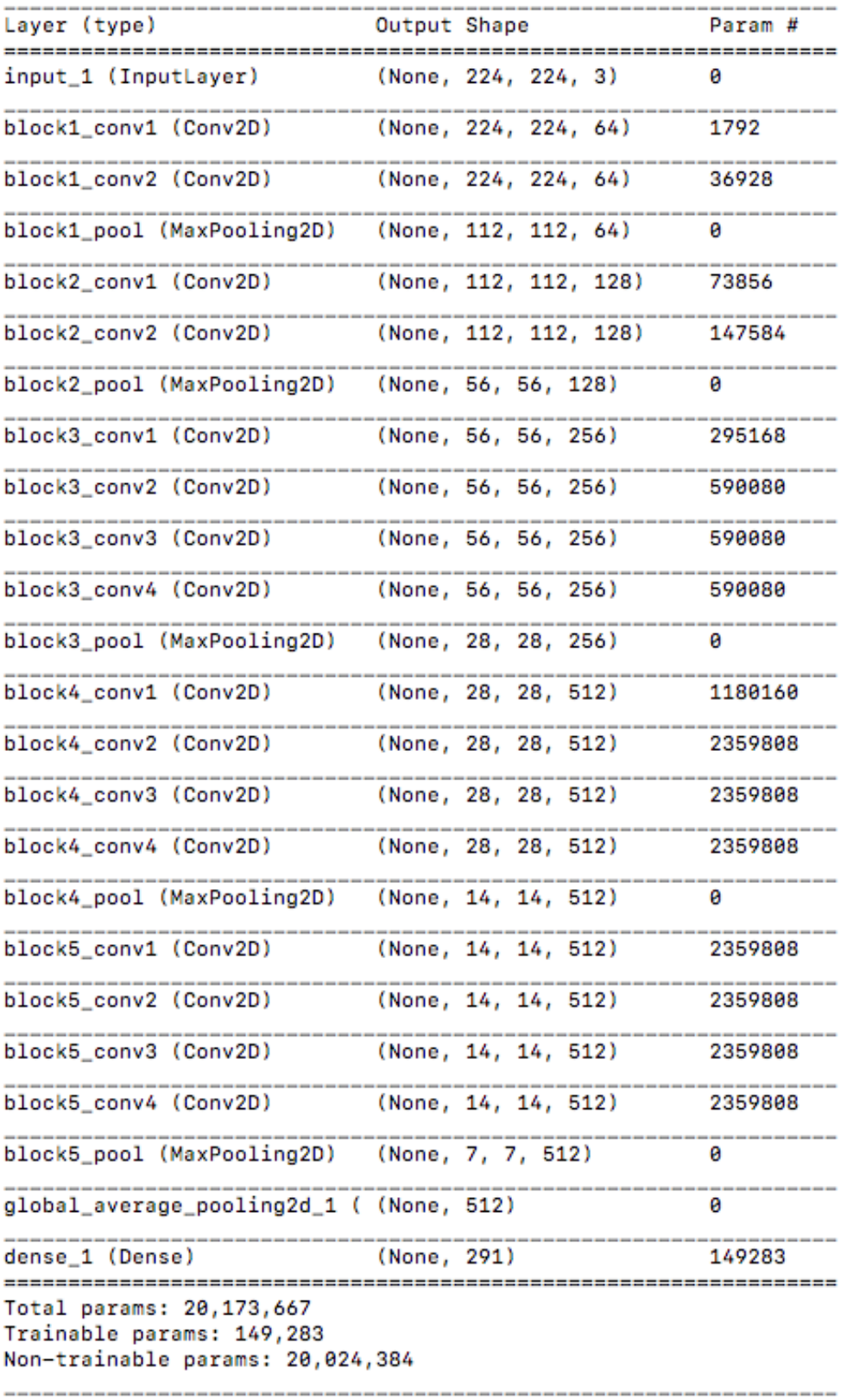


Figure 10 VGG19 and Resnet34 network architectures for ImageNet [7]



Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_average_pooling2d_1 (	(None, 512)	0
dense_1 (Dense)	(None, 291)	149283

Total params: 20,173,667
Trainable params: 149,283
Non-trainable params: 20,024,384

Figure 11 VGG19 model summary

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_1[0][0]
conv1 (Conv2D)	(None, 112, 112, 64)	9472	conv1_pad[0][0]
bn_conv1 (BatchNormalization)	(None, 112, 112, 64)	256	conv1[0][0]
activation_1 (Activation)	(None, 112, 112, 64)	0	bn_conv1[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	activation_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
res2a_branch2a (Conv2D)	(None, 56, 56, 64)	4160	max_pooling2d_1[0][0]
bn2a_branch2a (BatchNormalizati	(None, 56, 56, 64)	256	res2a_branch2a[0][0]
activation_2 (Activation)	(None, 56, 56, 64)	0	bn2a_branch2a[0][0]
res2a_branch2b (Conv2D)	(None, 56, 56, 64)	36928	activation_2[0][0]
bn2a_branch2b (BatchNormalizati	(None, 56, 56, 64)	256	res2a_branch2b[0][0]
activation_3 (Activation)	(None, 56, 56, 64)	0	bn2a_branch2b[0][0]
res2a_branch2c (Conv2D)	(None, 56, 56, 256)	16640	activation_3[0][0]
res2a_branch1 (Conv2D)	(None, 56, 56, 256)	16640	max_pooling2d_1[0][0]
bn2a_branch2c (BatchNormalizati	(None, 56, 56, 256)	1024	res2a_branch2c[0][0]
bn2a_branch1 (BatchNormalizatio	(None, 56, 56, 256)	1024	res2a_branch1[0][0]
add_1 (Add)	(None, 56, 56, 256)	0	bn2a_branch2c[0][0] bn2a_branch1[0][0]

.....

activation_43 (Activation)	(None, 7, 7, 2048)	0	add_14[0][0]
res5b_branch2a (Conv2D)	(None, 7, 7, 512)	1049088	activation_43[0][0]
bn5b_branch2a (BatchNormalizati	(None, 7, 7, 512)	2048	res5b_branch2a[0][0]
activation_44 (Activation)	(None, 7, 7, 512)	0	bn5b_branch2a[0][0]
res5b_branch2b (Conv2D)	(None, 7, 7, 512)	2359808	activation_44[0][0]
bn5b_branch2b (BatchNormalizati	(None, 7, 7, 512)	2048	res5b_branch2b[0][0]
activation_45 (Activation)	(None, 7, 7, 512)	0	bn5b_branch2b[0][0]
res5b_branch2c (Conv2D)	(None, 7, 7, 2048)	1050624	activation_45[0][0]
bn5b_branch2c (BatchNormalizati	(None, 7, 7, 2048)	8192	res5b_branch2c[0][0]
add_15 (Add)	(None, 7, 7, 2048)	0	bn5b_branch2c[0][0] activation_43[0][0]
activation_46 (Activation)	(None, 7, 7, 2048)	0	add_15[0][0]
res5c_branch2a (Conv2D)	(None, 7, 7, 512)	1049088	activation_46[0][0]
bn5c_branch2a (BatchNormalizati	(None, 7, 7, 512)	2048	res5c_branch2a[0][0]
activation_47 (Activation)	(None, 7, 7, 512)	0	bn5c_branch2a[0][0]
res5c_branch2b (Conv2D)	(None, 7, 7, 512)	2359808	activation_47[0][0]
bn5c_branch2b (BatchNormalizati	(None, 7, 7, 512)	2048	res5c_branch2b[0][0]
activation_48 (Activation)	(None, 7, 7, 512)	0	bn5c_branch2b[0][0]
res5c_branch2c (Conv2D)	(None, 7, 7, 2048)	1050624	activation_48[0][0]
bn5c_branch2c (BatchNormalizati	(None, 7, 7, 2048)	8192	res5c_branch2c[0][0]
add_16 (Add)	(None, 7, 7, 2048)	0	bn5c_branch2c[0][0] activation_46[0][0]
activation_49 (Activation)	(None, 7, 7, 2048)	0	add_16[0][0]
global_average_pooling2d_1 (Glo	(None, 2048)	0	activation_49[0][0]
dense_1 (Dense)	(None, 291)	596259	global_average_pooling2d_1[0][0]

Total params: 24,183,971
Trainable params: 596,259
Non-trainable params: 23,587,712

Figure 12 Resnet50 model summary

3.5.1.2 Training & Evaluation

Once we have the dataset and the algorithm ready, the next job is to split the data into training and validation sets. For our VGG model, the ratio of the two splits in our case is 80/20. We use 80% of the data for training, while reserve the rest 20% for validation the accuracy of the model. It is vital to split the data randomly, this helps to avoid bias in the machine learning model in case the original dataset is arranged in ascending or descending order.

For Resnet50 model, during our experiments, we encounter overfitting issue, as the model presents higher training accuracy with much lower testing accuracy. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on the new data. To improve overfitting issue, we use cross-validation technique. Instead of splitting the whole dataset into training and validation set, we iterate over different validation sets and averaging the performance across these. For example, in k–fold cross-validation, we split the data into k equal folds, use $k-1$ folds for training and 1-fold for validation. Then we iterate over each fold for validation, evaluate the performance, and finally average the performance across different folds. Figure 13 is a data split example for 5-fold cross-validation.

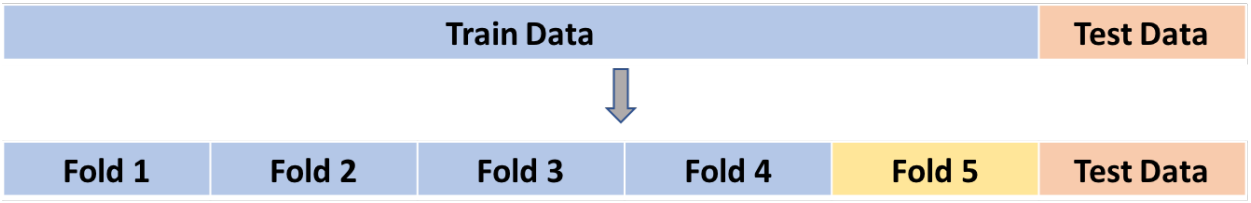


Figure 13 5-fold cross-validation example.
Here fold 1-4 are training set, fold 5 is validation set. Cross-validation iterates over the choice of which fold is the validation fold, separately from 1-5. In the very end, the model is evaluated a single time on the test data.

Which k value should we use in our project? An optimal number of folds depends on the sample size, number of classes, number of attributes, data distribution and computing recourses. Ideally, we can try different k values to decide which one is best. However, such process is expensive, it consumes time and computing recourses. Typical folds we can see in practice would be 3-fold, 5-fold or 10-fold cross-validation. According to a large experimentation describe in [52], Kohavi shows for real world datasets, the best method to use for model selection is 10-fold stratified cross-validation, even if computation power allows using more folds. However, in our datasets, there are

204 of 987 landmarks contain less than or equal to ten images. If we apply 10-fold cross-validation, in each iteration, there would be less than or equal to one image used for validation for 20.67% of our whole dataset. In other words, we don't have sufficient samples in validation set to confidently evaluate our model. Therefore, we use 5-fold cross validation in our project. In our Results Section, we will compare the performance with and without cross-validation for Resnet model.

The next step is to measure how well the model actually performs. We use classification accuracy for our landmark recognition model. It is the ration of number of correct predictions to the total number of input samples. A detailed result is in Section 5.

$$\text{Accuracy} = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

3.6 iOS Application Integration

3.6.1.1 Architecture

The mobile application is built on Model-View-Controller (MVC) architectural pattern. Figure 10 shows the application architecture. Class MLModel is an encapsulation of all the details of the machine learning model, including prediction methods, configuration, and model description. Our application has a simple user interface, which allows user to take photos with device camera or choose photos from device. It also displays the classification results from the machine learning model. Controller objects interpret user actions made in view objects, such as add photos from device, and communicate new or changed data to the model layer, such as return classification results.

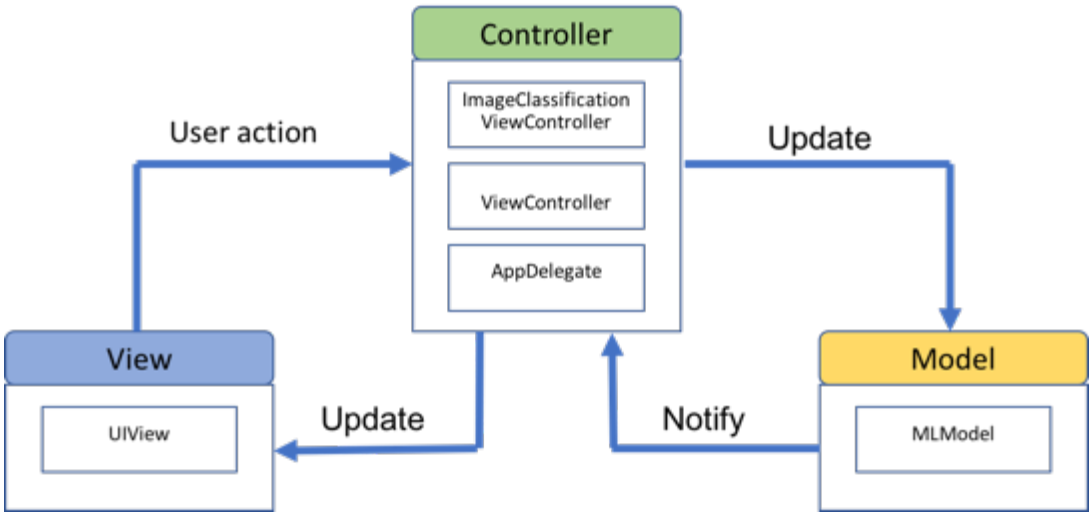


Figure 14 Mobile application architecture

3.6.1.2 Testing Methodology

We want to understand how real users interact with our model and the mobile application, so we perform usability test. In our test, we invite family members, friends and classmates to use our Chinese landmark recognition application on the developer device. We want to know: (1) Do they interested in Chinese landmark recognition? (2) Are they satisfied with the classification results? (3) Are they satisfied with the waiting time? (4) Are they satisfied with the user interface? (5) Do they have any suggestions about our application? The results are in Section 5.4.

3.7 Tools and Frameworks

3.7.1 Software Resources

Category	Technology	Description
Language	Python	Python was used for the end-to-end development of the system. I choose Python for its ease of use and rich ecosystem.
	Node.js	Node.js was used for web scraping website involved JavaScript.
Tool	PyCharm	The whole project was developed using PyCharm.
	Terminal	Terminal was used to run Node.js file.

	X2Go	X2Go was used to connect to the XFCE graphical desktop on Azure virtual machine.
Operating System	MacOS High sierra 10.13.4	Entire project was developed in Apple Mac operating system. However, the language and tools used are platform-independent.
	Data Science Virtual Machine for Linux(Ubuntu)	The model training was done on Azure Linux virtual machine with GPU for efficiency purpose. However, it can also be done on CPU.
Library	Urllib [23]	Urllib was used to work with URLs.
	Beautiful Soup [24]	Beautiful Soup to pull data out of HTML and XML files.
	Puppeteer [25]	Puppeteer was used for web scraping website involved JavaScript.
	PIL [26]	Python image library. Used to download the image from internet.
	CSV	CSV was used to import and export format for databases.
	Numpy [27]	Scientific computing with Python.
	Panda [28]	Panda was used to deal with CSV file.
	Keras [29]	Keras is a high-level neural networks API. It was used for image clustering and deep learning model deploying.
	Matplotlib [30]	Matplotlib was used to produce data analysis figures.
	Tensorflow [31]	Tensorflow is an open source machine learning framework. It was used for image clustering and deep learning model deploying.
	Coremltools [32]	Coremltools is an Apple framework. It was used to convert the machine learning model to ".mlmodel" format, which can be integrated into iOS applications.
	Scipy [40]	Scipy is a Python library. It was used to perform hierarchical clustering.

Table 2 Software resources

3.7.2 Hardware Resources

Platform	Operating System	CPU/GPU	Memory
Mac	MacOS High sierra 10.13.4	CPU: 2.4 GHz Intel Core i5	8GB
Linux(Ubuntu)	Ubuntu 16.04.5	GPU: Tesla K80	10,758 MB

Table 3 Hardware resources

4 Implementation

In this section, we describe implementation detail to help others to replicate our work.

4.1 Dataset

4.1.1 Landmark List

We use Python to do web scraping for its ease and rich ecosystem. There are two Python modules used here. (1) Urlib package to work with URLs. (2) Beautiful Soup to pull data out of HTML and XML files. It provides idiomatic ways of navigating, searching and modifying the parse tree.

Before we write codes to do web scraping, we need to inspect the website and locate the information of HTML format we need. For this task, we are interested in the popular landmarks in China, which serves as the category for our landmark recognition model. On Chrome, open the website we are interested, right click and choose “inspect”, we can get source code of the website. By right click and choose “inspect” again on the elements we want to extract, the according HTML selectors will highlight on the source code page. Figure 15, 16 are the website screenshots with according elements we are interested from Ctrip. First, we get the list of popular cities in China, then for each city, wo get the list of popular points of interest. Processes are similar for TripAdvisor, Google and Baidu. The most important thing is to inspect the website and locate the HTML elements.

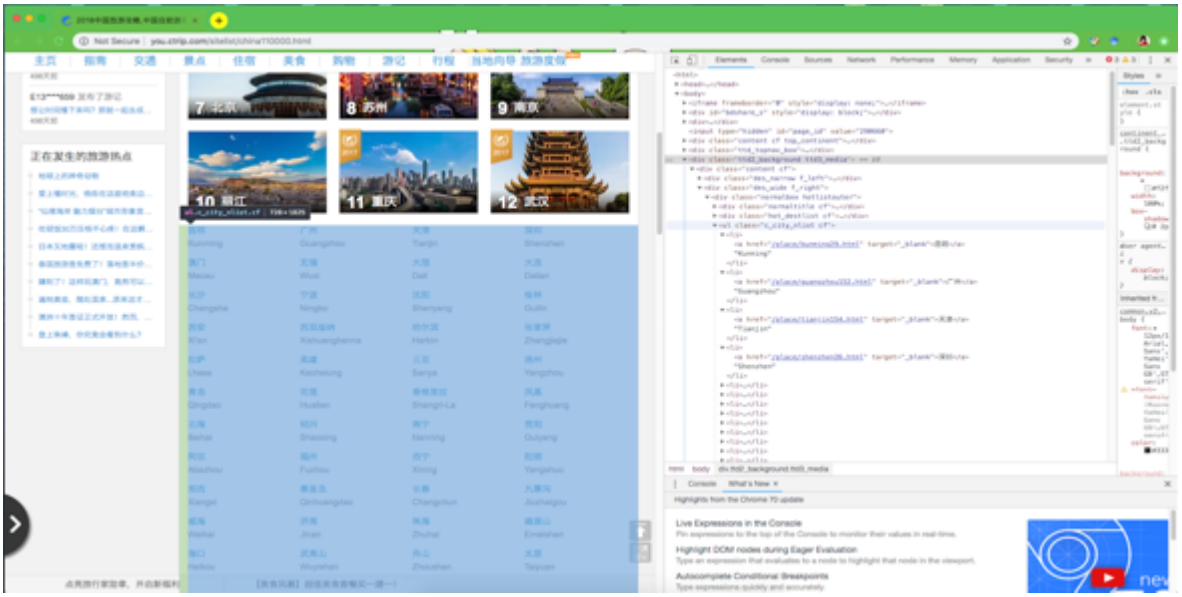


Figure 15 Step 1: Scrape the URLs for cites

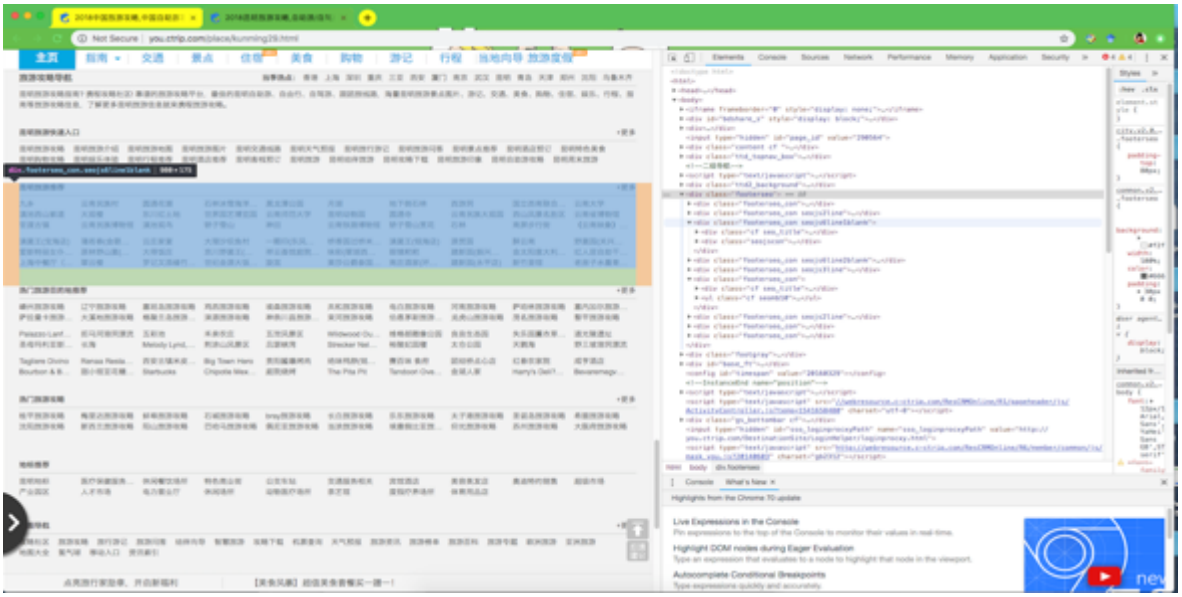


Figure 16 Step 2: Scrape URLs for landmarks

4.1.2 Landmark Images

With the list we get from the first step, we can further web scraping the URLs for the images. The images are from: (1) Images user upload as comments on Ctrip and TripAdvisor. (2) Images returned by Google Image search.

Ctrip. Processes are similar with web scraping the landmark list. As long as we locate the HTML elements, which contains the URLs for the images, we can extract them from the website. Figure 17 is the website screenshot and according elements.

Figure 17 Scrape the URLs for images from user comments for each landmark

TripAdvisor. Instead of using URLs to display the image gallery, TripAdvisor use JavaScript to send a request to remote server to load the user comment images as a gallery page. More specifically, we need to click a button labels as “All photos” to send the request and get the images showed on another web page. We use Node.js and node library Puppeteer to solve this issue. Like the name suggests, Puppeteer provides methods such as “click” to click a DOM element to acted like a person. It is also a powerful tool for automated testing. Figure 18, 19 are the website screenshots and according elements from TripAdvisor.

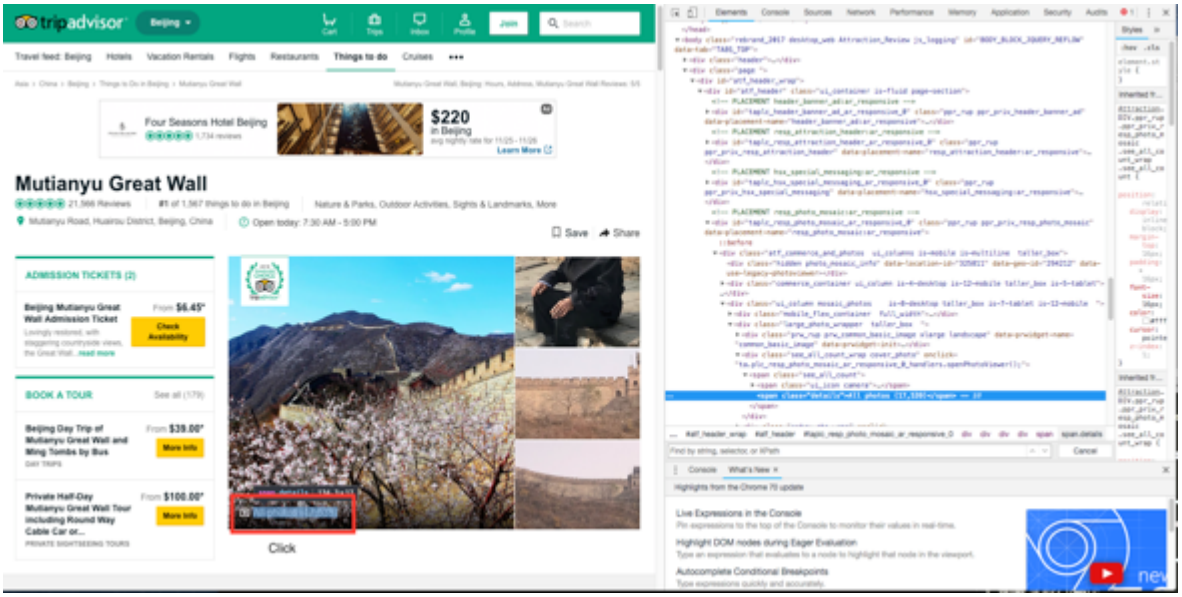


Figure 18 Step 1: Click the button to send the request

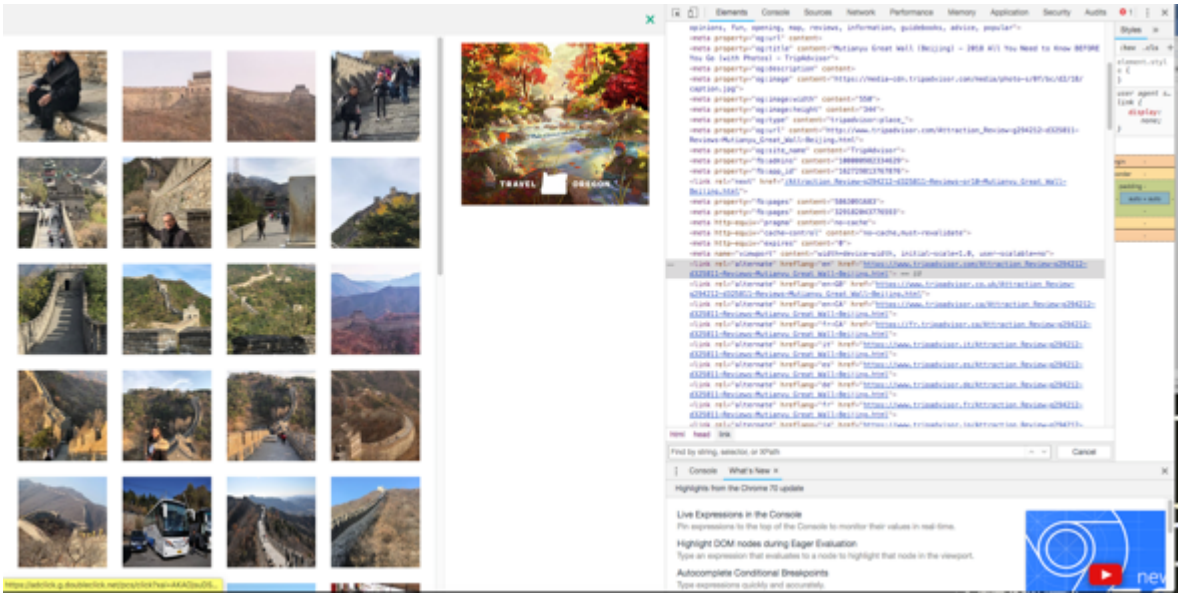


Figure 19 Step 2: Scrape URLs for images from the gallery page

Google Image. We use landmark names from the landmark list as the search query to search Google Image and scrape the URLs for the images. Figure 20 is the website screenshot and according elements.

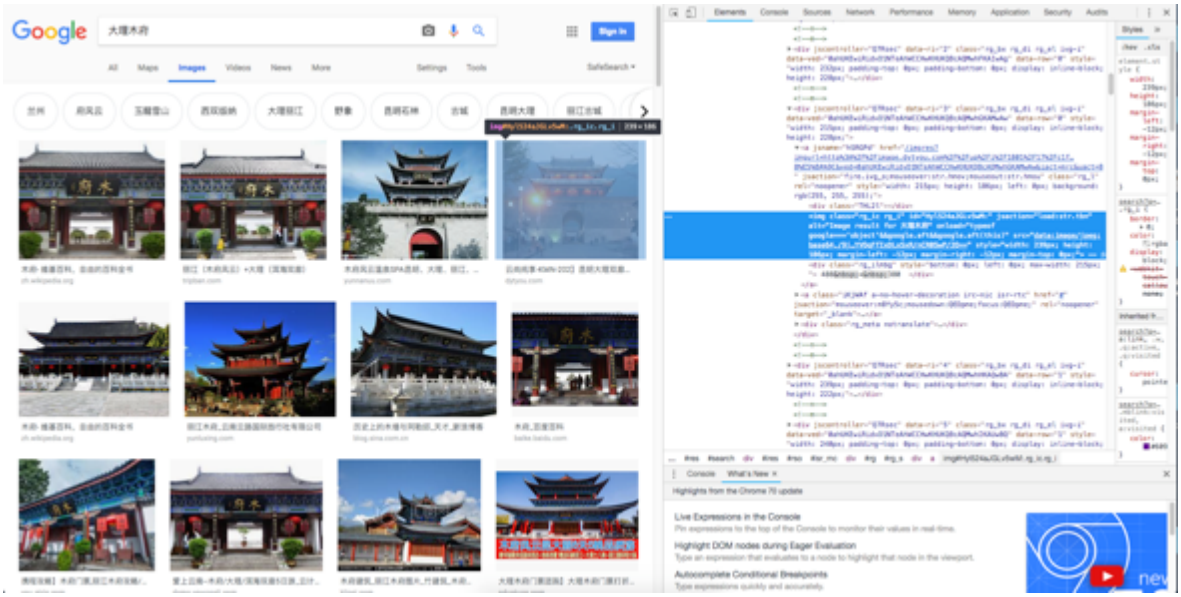


Figure 20 Scrape Google Image search results

4.1.3 Image Downloading

We use Pillow Image module to convert the raw pixel values from the URLs we get from above steps to image format.

4.1.4 Data Clustering

We use pretrained VGG19 model as a feature extractor to hierarchical cluster all the images. Figure 21 shows image clustering results for one landmark. The images in the same red box belong to a same group. Once we have done image clustering, we have to validate the results by deciding whether we keep or discard the image group base on the content of the group. For example, for the landmark id 92 in figure 22, we only keep the correct group on the right and discard other groups. For the landmark id 16 in figure 23, we keep all three groups, because even though there are differences among groups, actually they are the same landmark under different weather conditions or from different points of view, which are what we want for training a deep learning model. We write help functions to make the validation process efficiently. Help functions iterate through all the groups, show thumbnails of sample images in the same group, then expect keyboard input from us to keep or remove this group.

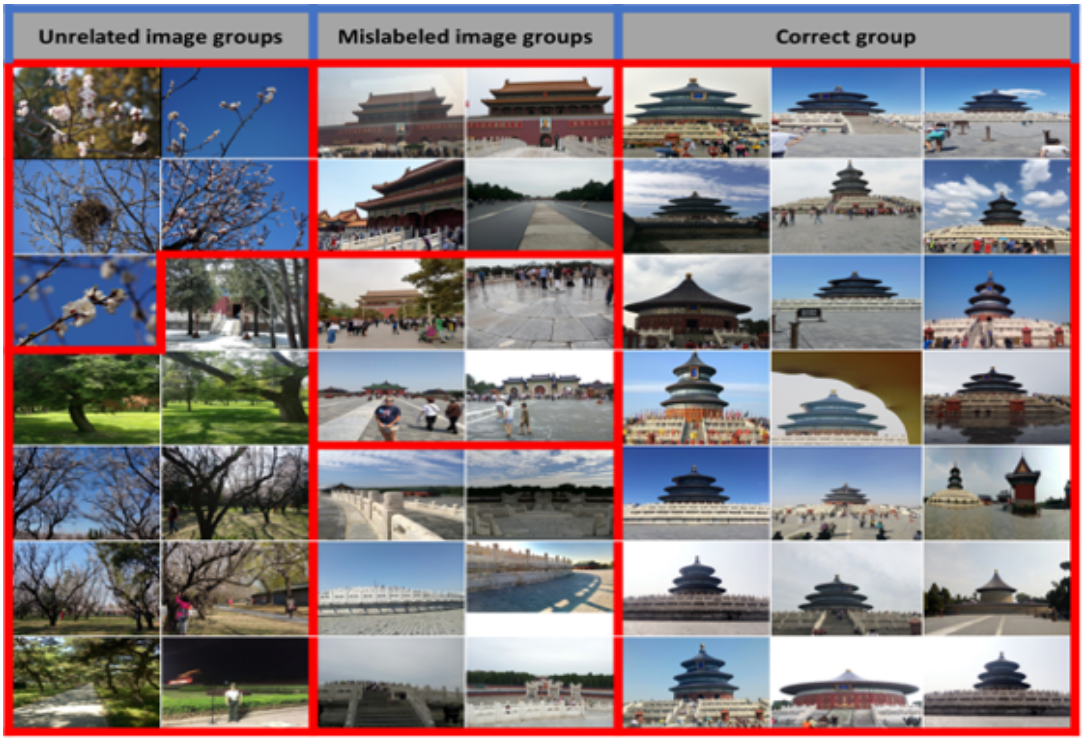


Figure 21 Image clustering results for landmark id 86
Images in one red box belong to one group. We manually keep the correct group and discard unrelated and mislabeled images.

Figure 22 Image clustering results for landmark id 92.
We keep the right group, discard the others.

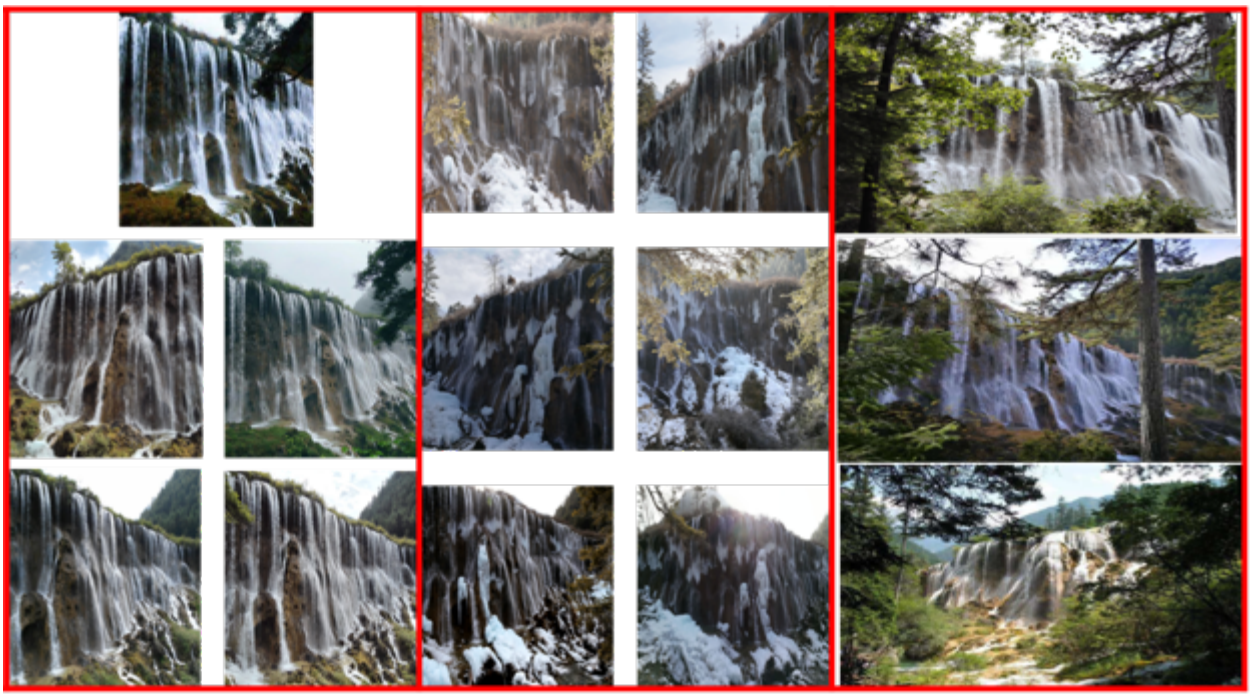


Figure 23 Image clustering results for landmark id 16.
We keep all groups.

4.2 Model Training & Evaluation

Deploying CNNs are computationally intensive task, because they need to update millions of parameters numerous times to minimize errors and produce a model as accurate as possible. Those updates are basically large matrix multiply operations, which can be run in parallel. Compared to CPU, GPU has more computational units and higher bandwidth. Therefore, the CNN architecture fits quite well with the kinds of computation that a GPU can efficiently perform. From our experiences, GPU is faster than CPU by at least 8 times when performing CNN training (base on the hardware resources described in Table 2). Section 4.2.1 discusses the cloud GPU options we use in our project since we do not own a one locally. Section 4.2.2 discuss models and hyperparameters.

4.2.1 Training & Evaluation with GPU

We test our model with sub-set of the whole database on Google Colaboratory with its free GPU. However, Google Colaboratory is intended for interactive use, long-running background computation on GPU may be stopped. Therefore, we use fee-based Virtual Machine for large dataset training and evaluation.

4.2.1.1 Google Colaboratory

Google Colaboratory [39] is a research tool for machine learning education and research purpose. It is a Jupyter notebook environment that can be connected with Google Drive. Most importantly, it provides free Tesla K80 GPU. Figure 24-27 present steps to set up Google Colaboratory, perform authorization, mount and save models to Google Drive.

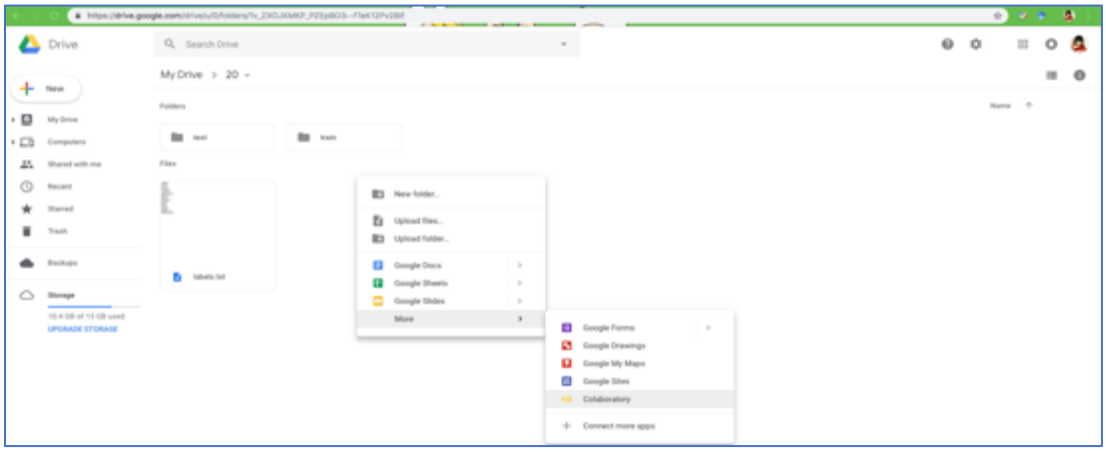


Figure 24 Step 1: Log into Google Drive, right click -> More -> Colaboratory

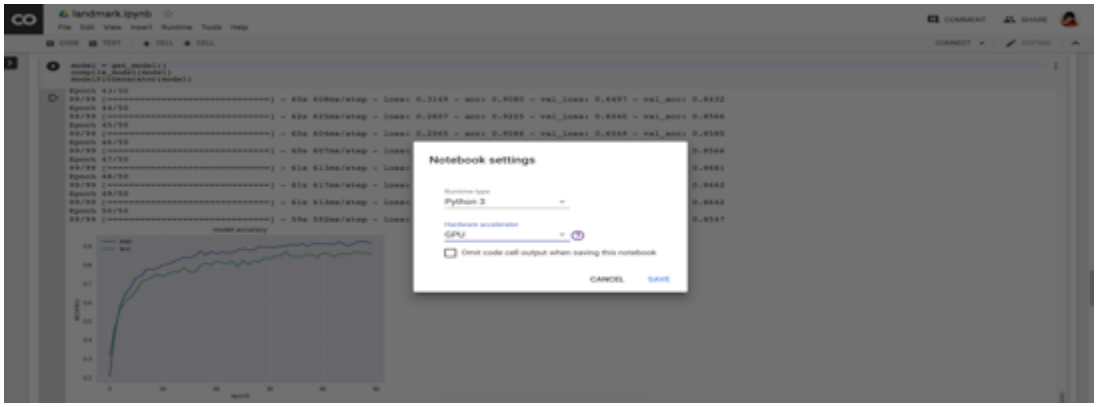


Figure 25 Step 2: Hardware setting, Edit -> Notebook setting-> GPU

```
# Step3: check whether you are using GPU
from tensorflow.python.client import device_lib
device_lib.list_local_devices()

# The output is as follows
[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 16357959345473311073, name: "/device:XLA_CPU:0"
 device_type: "XLA_CPU"
 memory_limit: 17179869184
 locality {
 }
 ……..
 incarnation: 7055823197323476595
 physical_device_desc: "device: 0, name: Tesla K80, pci bus id: 0000:00:04.0, compute capability: 3.7"]
```

Figure 26 Step 3: Check the GPU information

```
# Step 4: Perform authorization with Google Drive account
# Stpe 4.1: Generate auth tokens for Colab
from google.colab import auth
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from oauth2client.client import GoogleCredentials
auth.authenticate_user()
gauth = GoogleAuth()

# Sept 4.2: Generate creds for the Drive Fuse libirary
creds = GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret} < /dev/null 2>&1 | grep URL
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret}

#Step 4.3: After run step 1 and 2, it will show a link and wait for your input. Click the link and copy, paste the verification code.
#Then we can read our Google Drive with:

# Step 4.4: Create a directory and mount Google Drive using that directory.
!mkdir -p drive
!google-drive-ocamlfuse drive

# Stpe 4.5: Show ile list in our Google Drive
!ls drive/
```

Figure 27 Step 4: Perform authorization and mount Google Drive

4.2.1.2 Virtual Machine

There are couple of options for cloud service we can choose from. We use Microsoft Azure Data Science Virtual Machine (ubuntu) to deploy our model. Most of the libraries we need are preinstalled and configured to use. We use Secure Shell (SSH) for secure remote login and data transfer with the virtual machine. Moreover, we use X2Go to connect to the XFCE graphical desktop on the Virtual Machine.

4.2.2 Models and Hyperparameters

We use two latest CNN models provided by Keras to compare the results. (1) VGG19. VGGNet refers to a deep CNN developed by Oxford's renowned Visual Geometry Group(VGG), which achieved very good performance in ILSVRC 2014 by adopting an architecture with small convolution filters with 16-19 weight layers. A downside of VGG is that it is more expensive to evaluate and uses a lot more memory and parameters. However, most of these parameters are in the fully connected layer, which is removed from our landmark recognition model, so it is not a concern for us. (2) ResNet50. ResNet is an abbreviation for residual neural network. It solves the degradation problem of convolutional neural networks by using shortcuts between layers. It helps as the network gets deeper and accordingly gets better performance. ResNet50 is a 50-layered network trained on ImageNet.

Architecture	No. Parameters	No. Multiply and Adds	Top-1 Accuracy	Year
VGG19	138M	15.5B	70.5%	2014
ResNet50	25.5M	3.9B	75.2%	2015

Table 4 VGG19 and ResNet50 key figures.
Top-1 accuracy refers to their evaluation results for ImageNet dataset.

We use Python library Keras to implement our model for training and evaluation. Table 5 shows the parameters we used in our model.

	Parameters	Value
Basic model: VGG19, ResNet50	weights	imagenet
	inputsize	224, 224
	data_format	channels_last
	activation	activation
Compile model	Optimizer	Adadelta
	lr (Initial learning rate)	1
	rho (Adadelta decay factor)	0.95
	loss	keras.losses.categorical_crossentropy
	metrics	accuracy
Data augmentation	rotation_range	90
	horizontal_flip	True

(Only used for landmarks contain less than 300 images.)	vertical_flip	True
	zoom_range	0.4
Others (K only used in Resnet50 for k-fold cross-validation.)	epoch	50
	batch_size	16
	k	5

Table 5 Model parameters

5 Results

This chapter presents the dataset, the evaluation of the landmarks recognition model, the user interface demo and usability testing results.

5.1 Dataset

Chinese landmark & attraction dataset contains 42,548 images for 987 unique landmarks. Figure 32 is the geographic distribution of it. Figure 33 shows some landmark images from the dataset. Figure 34 and 35 show distribution of the dataset. We also categorized the dataset with 34 labels for a comprehensive understanding of the dataset.

Figure 28 Geographic distribution of Chinese landmark & attraction dataset

Figure 29 Images from Chinese landmark & attraction dataset

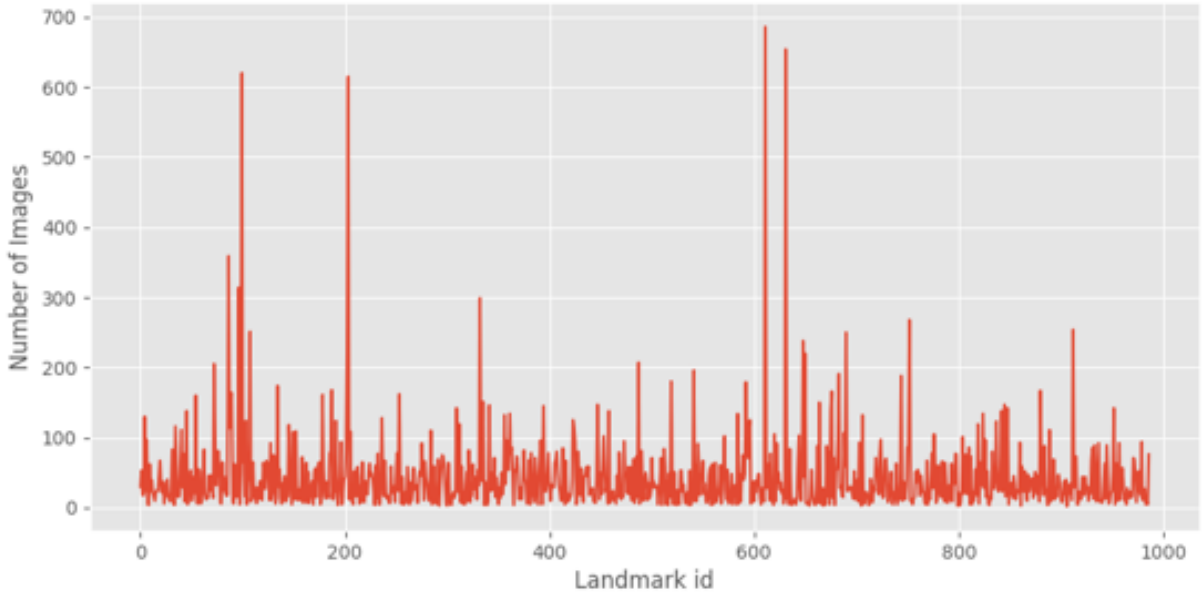


Figure 30 Landmark images distribution

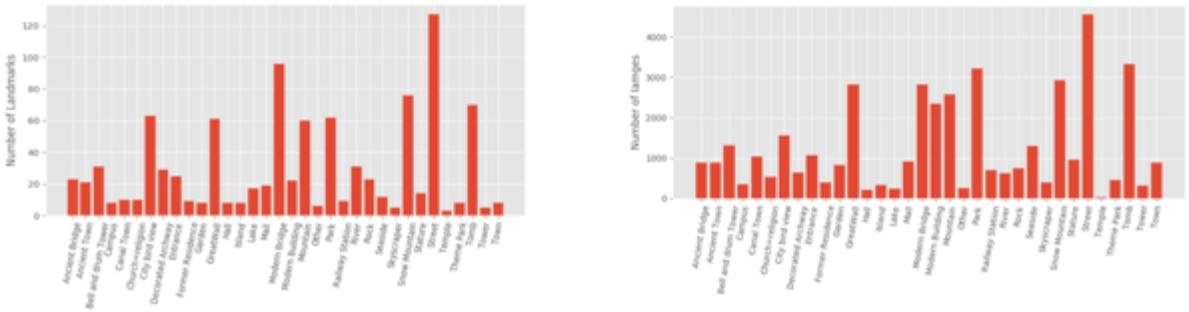


Figure 31 Landmark category distribution

5.2 Model Testing Results

We train models on different subset of Chinese landmark & attraction dataset with ResNet50 and VGG19 separately. Table 6 presents the four subsets we use for our experiments. The 1st dataset contains uncropped images with complex background. Most images in the 2nd datasets contain the landmarks in the center with a reasonable scale. For 3rd dataset, we remove categories with less than 10 images. For 4th dataset, we remove categories with less than 20 images. We expect as the average number of instances for each category increases, the more reliable the model would be.

	No. of categories	Total no. of instances	Average no. of instances per category	Description
1	1059	48,732	46	Dataset with uncropped images.
2	987	42,548	43	Dataset after image cropping.
3	494	32,808	66	Dataset with popular landmarks.
4	291	22,365	77	Dataset with popular landmarks.

Table 6 Datasets for landmark recognition model experiments

Table 7 presents the average training time for the four datasets. In general, VGG 19 takes longer training time than ResNet50.

Dataset	ResNet50	VGG19
1	64,140s	50,110s
2	50,841s	40,350s
3	40,986s	31,050s
4	23,220s	17,283s

Table 7 Training time for each experiment

Table 8 presents the training and validation accuracy of our models. We find: (1) In general, ResNet 50 model overfits our data by yielding much higher training accuracy than validation accuracy. For the 4th dataset, we do another experiment by applying 5-fold cross-validation, which improve the overfitting issue, but the model is still not good enough for further deployment. (2) Compare the 1st and 2nd experiment, we have a 15% performance increase in validation accuracy for VGG19 by cropping the landmarks out of the background. (3) The accuracy increases as the number of categories need to be classified reduces.

Dataset	ResNet50		VGG19	
	Training%	Validation%	Training%	Validation%
1	66.97	22.26	47.58	36.64
2	79.31	28.55	58.22	49.66
3	82.01	35.67	65.88	60.12
4	86.13	48.22	73.32	72.55
4 w/ cv	77.15	52.33	-	-

Table 8 Models training results
The last row is the model we use the 4th dataset with 5-fold cross-validation. And the highlight cells are the model we use for our iOS application.

Lastly, we integrate the fourth model, which is trained with 291 landmarks using VGG19 to our iOS mobile application. We also compare our model with other related projects. Table 9 presents the comparison results. From the table, our deep learning model is already integrated into the iOS application, which does not require a local database or Internet access. It takes less time for our model to do landmark recognition, and at the same time gives reasonable results.

	Accuracy	Avg. waiting time	Description
Tour the world [11]	46.3%	-	Image retrieval from a local landmark database.
Microsoft API	11.8%	2.50s	Cloud landmark recognition API.
Google API	35.7%	3.48s	Cloud landmark recognition API.
Scene maps [19]	80.65%	1.29s	Image retrieval from a local landmark database.
Our model	72.55%	0.87s	iOS landmark recognition application.

Table 9 Comparison with related projects
Tour the word [11] and Scene maps [19] require a local database to do landmark retrieval. Microsoft and Google API require internet access. Our model does not need local database, or internet access to use. Tour the world [11] does not provide retrieval time in their paper. Waiting time for Microsoft and Google API also depend on the Internet speed.

5.3 iOS Mobile Application

To use our landmark recognition model, we developed a mobile user interface for iOS, which is built with Apple CoreML framework. CoreML framework converts the landmark recognition model generated by Keras to a “mlmodle” format that recognized by iOS device. The user interface

allows user either take a photo or choose a photo from their device. Figure 36 shows steps to use the user interface.

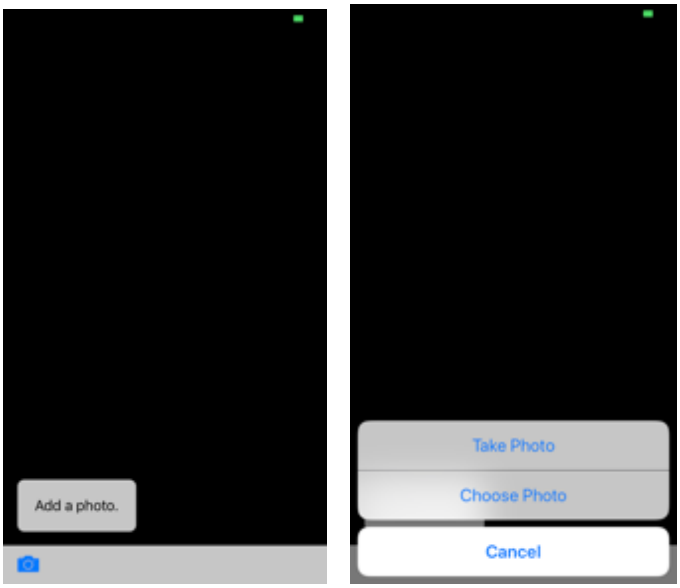
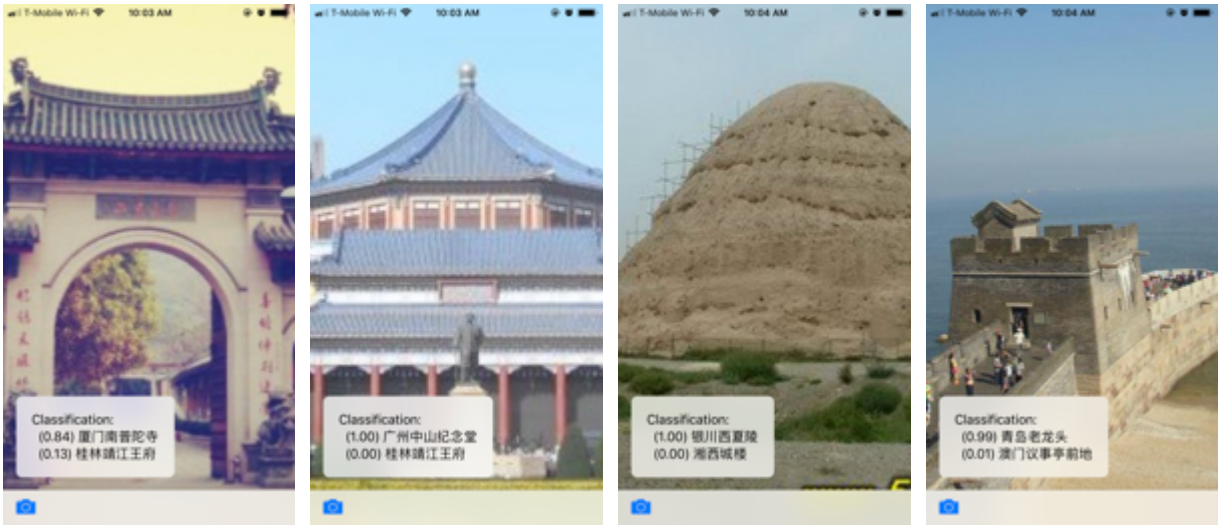


Figure 32 Mobile interface, two ways to add a photo

After we take a photo by the camera, or choose one from the mobile device, the landmark recognition model will classify the image automatically. Figure 37 is the collection of classification results.



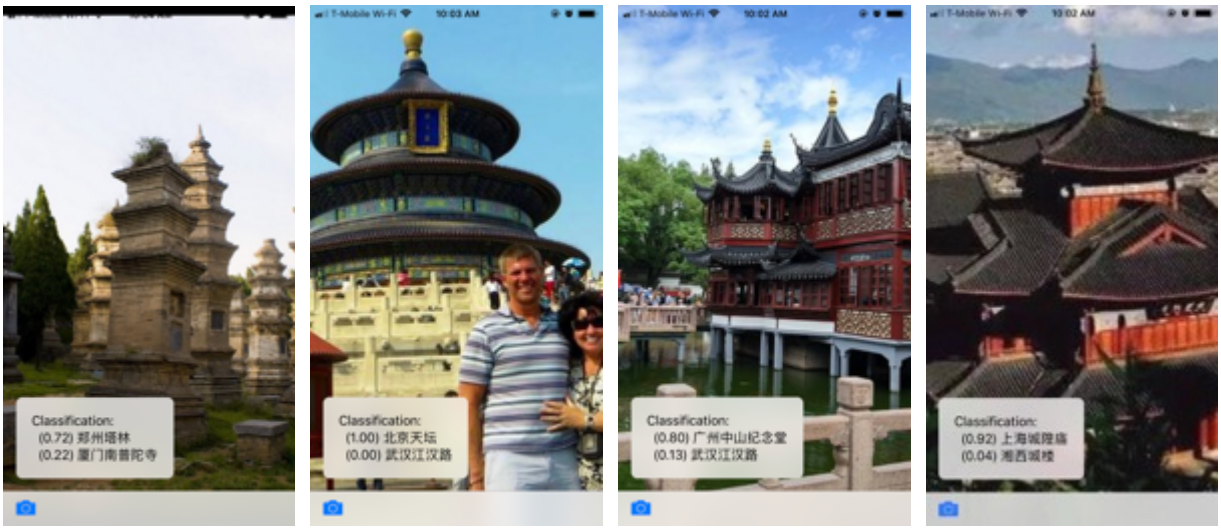


Figure 33 Screenshots of the mobile application

5.4 Usability Testing Results

We perform usability testing to our application. The app is installed on a developer device and we explained the functionality before the test. We obtained the following results.

(1) Users are interested in the idea the app is implemented for and express the willingness to give it a try.

(2) Most of the users did not use landmark recognition before. Couple of them use image search for a few times.

(3) The app is easy to use. Launch the app and touch on of the options (Take Photo vs Choose Photo). Only couple steps are required to reach the core functionality.

(4) They are satisfied with the waiting time. Most of the test cases get classification results instantly.

(5) The model targets for the landmarks in China. We and the users being selected are in United States. User tested the functionality via the pictures in their albums, pictures on the wall or the online images. For the popular national landmarks such as Forbidden City, The Great Wall, the app returns correct results. For some local landmarks, users were thrilled when the application returns the results. For images with people blocked the landmark, the application fails to recognize any landmarks. For some local landmarks, such as ZhenHaiLou in Fuzhou, the app recognizes it as Forbidden City due to the similarity of the building style.

Based on the test, most of the users expressed the curiosity on the technology that underpins the app. A detailed survey results can be found on [42].

Due to the social and geographical limitations, we couldn't conduct the usability test on the group of people that this app is designed to serve. If possible, we could have, for example, tourists who are travelling to China and test the usability with real landmarks, instead of the pictures.

6 Conclusion and Future Work

6.1 Conclusion

This project is motivated with a goal of helping people recognizing Chinese landmarks and attractions. We build Chinese landmark & attraction dataset using different technical, such as web scraping, unsupervised machine learning and content-based image searching. We use VGG architecture to perform the unsupervised machine learning for image clustering. We apply DELF, another deep learning model to retrieve landmark in the images and crop them. Then we use two CNN representations, ResNet50 and VGG19 to train our models and choose the best one to build an iOS application. Our new dataset complete current landmarks datasets and can benefit research for other image recognition problems as well. The mobile application offers convenient ways to help people to recognize Chinese landmarks.

6.2 Future Work

Based on the experiments and our findings, we believe that there are many possibilities for future research work.

(1) Keep updating Chinese landmark & attraction dataset. These include: add new categories and new images to the dataset; add different layers of annotation such as bounding boxes, part landmark and boundary segmentation.

(2) There are couple possible methods to improve the algorithm performance: further fin-tune the deep learning model; use advanced data augmentation techniques to our landmark datasets. For example, Conditional GANs [41] can transform an image from one domain to another. By using such techniques, we can transform photographs of summer sceneries to winter sceneries, which can make our model more robust.

7 References

1. Hubel, David H., and Torsten N. Wiesel. "Receptive fields and functional architecture of monkey striate cortex." *The Journal of physiology*195.1 (1968): 215-243.
2. Ciresan, Dan C., et al. "Flexible, high performance convolutional neural networks for image classification." *IJCAI Proceedings-International Joint Conference on Artificial Intelligence.* Vol. 22. No. 1. 2011.
3. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems.*
4. "Imagenet large Scale Visual Recognition Challenge 2017 (ILSVRC2017)." *Image-net.*, 2017, http://image-net.org/challenges/LSVRC/2017/index.
5. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:*1409.1556(2014).
6. Li, Xiaowei, et al. "Modeling and recognition of landmark image collections using iconic scene graphs." *European Conference on Computer Vision.* Springer, Berlin, Heidelberg, 2008.
7. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016.
8. "Google Cloud Vision API available to all." *Google, 2016*. https://cloud.google.com/blog/big-data/2016/02/google-cloud-vision-api-available-to-all
9. "Microsoft researcher's latest computer vision algorithm can outperform humans." *Dataconomy, 2015.* http://dataconomy.com/2015/02/microsoft-researchers-latest-computer-vision-algorithm-can-outperform-humans/
10. André Araujo, Tobias Weyand. "Google-Landmarks: A New Dataset and Challenge for Landmark Recognition." *Google blog, 1 March 2018.* https://ai.googleblog.com/2018/03/google-landmarks-new-dataset-and.html
11. Zheng, Yan-Tao, et al. "Tour the world: building a web-scale landmark recognition engine." *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on. IEEE,* 2009.
12. Noh, Hyeonwoo, et al. "Large-Scale Image Retrieval with Attentive Deep Local Features." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2017.

13. “Ctrip”. *Ctrip.* http://you.ctrip.com/sight/china110000.html.
14. “TripAdvisor”. *TripAdvisor.* https://www.tripadvisor.com/Tourism-g294211-China-Vacations.html.
15. “Google Map Search China.” *Google.* https://www.google.com/destination/map/china.
16. “Baidu Map Search China.” *Baidu.* https://lvyou.baidu.com/zhongguo/jingdian.
17. Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision. Springer, Cham,* 2014.
18. M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.* http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.
19. Avrithis, Yannis, et al. "Retrieving landmark and non-landmark images from community photo collections." *Proceedings of the 18th ACM international conference on Multimedia.* ACM, 2010.
20. Welinder, Peter, et al. "Caltech-UCSD birds 200." (2010).
21. Sharif Razavian, Ali, et al. "CNN features off-the-shelf: an astounding baseline for recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition workshops.* 2014.
22. Oquab, Maxime, et al. "Learning and transferring mid-level image representations using convolutional neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2014.
23. “urllib — URL handling modules”. *Python.* https://docs.python.org/3/library/urllib.html
24. “Beautiful Soup Documentation.” *Crummy.* https://www.crummy.com/software/BeautifulSoup/bs4/doc/
25. “Puppeteer”. *Pptr.* http://pptr.dev
26. “Image Module”. *Pillow.* https://pillow.readthedocs.io/en/3.1.x/reference/Image.html
27. “Numpy”. *Numpy.* http://www.numpy.org/
28. “Python data analysis library”. *Pandas.* https://pandas.pydata.org/
29. “Keas: the python deep learning library”. *Keras.* https://keras.io/
30. “Matplotlib”. *Matplotlib.* https://matplotlib.org/
31. “TensorFlow: an open source machine learning framework for everyone.” *Tensorflow.* https://www.tensorflow.org/

32. “Coremltools2.0: pip install coremltoos”.*Pypi.* https://pypi.org/project/coremltools/
33. Frigui, Hichem, and Raghu Krishnapuram. "A robust competitive clustering algorithm with applications in computer vision." *IEEE transactions on pattern analysis and machine intelligence* 21.5 (1999): 450-465.
34. Leung, Yee, Jiang-She Zhang, and Zong-Ben Xu. "Clustering by scale-space filtering." *IEEE Transactions on pattern analysis and machine intelligence* 22.12 (2000): 1396-1410.
35. Guérin, Joris, et al. "CNN features are also great at unsupervised classification." *arXiv preprint arXiv:*1707.01700 (2017).
36. Huang, Jonathan, et al. "Speed/accuracy trade-offs for modern convolutional object detectors." *IEEE CVPR*. Vol. 4. 2017.
37. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.
38. Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2015.
39. “Welcome to Colaboratory!” *Colab.* https://colab.research.google.com/notebooks/welcome.ipynb#scrollTo=-Rh3-Vt9Nev9
40. “SciPy library.” *Scipy.* https://www.scipy.org/scipylib/index.html
41. Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." *arXiv preprint* . 2017.
42. “Landmark Recognition Application Usability Survey”. Survey Monkey. https://www.surveymonkey.com/analyze/m7zjycOTG9XrArfX79GigOwNKZaMC6jHPUNzCOSh8js_3D
43. Russell, Stuart J., and Peter Norvig. Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, 2016.
44. “China Tourism — Current Trends and Facts” https://www.chinahighlights.com/travelguide/tourism.htm
45. “Google-Landmarks Dataset Label famous (and not-so-famous) landmarks in images.” https://www.kaggle.com/google/google-landmarks-dataset#train.csv
46. Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." *International Journal of Computer Vision* 115.3 (2015): 211-252.
47. Pyle, Dorian. Data preparation for data mining. *Morgan Kaufmann,* 1999.

48. Torralba, Antonio, and Alexei A. Efros. "Unbiased look at dataset bias." *Computer Vision and Pattern Recognition (CVPR),* 2011 IEEE Conference on. IEEE, 2011.
49. Zeiler, Matthew D. "ADADELTA: an adaptive learning rate method." *arXiv preprint arXiv:*1212.5701 (2012).
50. Hsu, Chih-Chung, and Chia-Wen Lin. "CNN-based joint clustering and representation learning with feature drift compensation for large-scale image data*." IEEE Transactions on Multimedia* 20.2 (2018): 421-429.
51. Mikolov, Tomáš, et al. "Recurrent neural network-based language model." *Eleventh Annual Conference of the International Speech Communication Association.* 2010.
52. Kohavi, Ron. "A study of cross-validation and bootstrap for accuracy estimation and model selection." *Ijcai.* Vol. 14. No. 2. 1995.