Visual place recognition from Google Streetview images

Ananya Ganesh, Hadi Nishad, Srideepika Jayaraman
CS 670, Computer Vision, Fall 2017
University of Massachusetts Amherst
aganesh@umass.edu
hnishad@umass.edu
srideepikaja@umass.edu

Abstract

Visual place recognition, or geolocation from images aims to identify the location where the picture was taken without using geotags, that is, just by utilizing the information in the image. In this paper, we will explore finding where a streetview image was taken from. Machine learning approaches to this task involve training a model on a large collection of images labelled with geographical data, where handcrafted features are extracted from the images before training. In this paper, we present the results of three machine learning models which use different kinds of features and evaluate their suitability for this task. The first approach uses a color histogram to represent the image; the second approach uses the GIST global descriptor as image features; the third approach uses raw images with a convolutional neural network without explicitly extracting features. We describe the performance of these approaches in two scenarios: first when geolocation is treated as a classification problem over discrete labels like cities, and second as a regression problem over continuous values like location coordinates.

1. Introduction

The task of determining the location of an image is an interesting and challenging problem. Potential applications include helping locate kidnap victims from photos or automatically geotagging photos. Outdoor images typically have distinctive features that carry information about the location at various levels of granularity. For instance, vegetation and landscape provide cues about the country or continent whereas more specific location information such as city or street can be elicited from buildings, architectural styles, etc.

We were motivated to consider a machine learning approach to this task after observing human performance on GeoGuessr, a game in which the player is placed in a random Google streetview scene from anywhere in the world and has to identify the location only by looking at his surroundings. We observed that with every game, our scores improved significantly as we learned to focus on the right features. We were also able to pick up on finer details with more examples, such as how a certain model of car is more common in Eastern Russia. Therefore, with enough data and the right kind of features, a computer system should be able to perform at least as well as humans.

We first formulate the problem as a classification task, similar to [WKP16]. Given a set of images taken in three cities - Manhattan, Orlando, and Pittsburgh, we aim to correctly predict which city an unseen image belongs to. The images are represented in three ways: using a GIST descriptor, a color histogram, and raw pixel values. Each representation is used to train a different model. Every image in the training data is labeled by the city where it was taken. We evaluate the performance on this task based on classification accuracy score alone, that is, whether the prediction is right or wrong.

Next, we perform finer-grained prediction on the location of a test image. We modeled this as a regression problem on the coordinates of the image. Each image in the training data possesses two labels - the latitude and longitude of its location. The model correspondingly predicts two labels for the coordinates of the test image. We evaluate the performance on this task based on how far the predicted location is from the true location.

We present three main contributions through this paper: 1) locating an exact point in a city as opposed to naming the city as other papers do, 2) analyzing the performance of pipeline model where the output of a classification model is used to pick a regression model 3) finding out if some views in StreetView provide more information than others. The rest of the paper is organized as follows: Section 2 describes related work, Section 3 describes our approach, the dataset, and implementation details, and Section 4 contains

the results of our experiments.

2. Related Work

Previous approaches to geolocation from images make use of image retrieval. Im2GPS [HE08] is a system, which when given a test image, retrieves similar images from millions of geotagged Flickr images, and then assigns the location of the closest match. The images in this dataset are represented by a combination of six different global image descriptors. There are over 6.5 million images, which are matched using a nearest neighbour approach.

Google's PlaNet [WKP16] is a system that employs the same classification approach to place recognition that we adopt. However, they target an unrestricted area, basically anywhere in the world where Streetview data is available, while we limit ourselves to three cities. This is partly to reduce computational load involved while training, but we also find that much better accuracy can be achieved on a restricted subset. PlaNet divides the earth into geographical cells which form the classes, and then outputs a discrete probability distribution over all cells.

NetVLAD [Ara+16] proposes a convolutional neural network architecture that is end to end trainable for weakly supervised place recognition. They design a generalized layer inspired by the Vector of Locally Aggregated Descriptors, called NetVLAD. This can be incorporated into any CNN and trained using back-propagation. The NetVLAD system also approaches this problem as image retrieval, and assigns the location based on the image most similar to the query point.

The closest work to ours is LittlePlaNet [HW] from the cs231N class at Stanford. They use a classification based approach on a limited dataset of Streetview images from 10 cities, using a convolutional neural network. They also test their model on non-Streetview images from Flickr, which we don't attempt. Also, we try to predict the exact location by regression, which has not been the focus of any system so far.

3. Our approach

3.1. Dataset

We use the Google Street View Dataset [ZS14] provided by the University of Central Florida which contains 62058 Google Street View images. The images cover three cities: Pittsburgh, PA; Orlando, FL; and Manhattan, NY. Specifically, the images include downtown and neighbouring areas of the three cities. In Figure 2, samples from each of these cities are displayed for view 4. We can see that they are visually different, in terms of the architecture, materials, colors, etc.

For each image, the coordinates of the images and the compass direction indicating the viewpoint are also available. A single location is captured from 6 views: views 1 to 4 are from the side and view 5 is an upward view. View 0 has the streetview markers intact. Figure 1 shows the views 0-5 from left-right. For each image, precomputed GIST features and color histogram are also provided. The GIST descriptor is a 960 dimension vector whereas the RGB color histogram is 60 dimensional, 20 for each channel. Although the features were precomputed for the training data, we compute them for the query images.

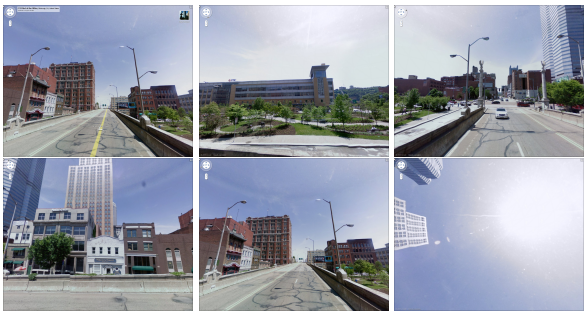

Figure 1. Views 0-5

Figure 2. View 4 for Pittsburgh, Orlando and New York

3.2. GIST descriptor

GIST [OT01] is a global descriptor used to extract a low dimensional representation of the scene using spectral and coarsely localized information. It generates a multidimensional space in which scenes that are semantically similar are projected together. Specific information about object shape is not required, and a holistic representation of the scene gives more information about the semantic category.

3.3. Color Histogram

The color histogram represents how colors are distributed in the image. It shows different types of colors appeared and the number of pixels in each type of the colors appeared. A histogram of an image is produced first by discretization of the colors in the image into a number of bins, and counting the number of image pixels in each bin. For example, a RedBlue chromaticity histogram can be formed by first normalizing color pixel values by dividing RGB values by R+G+B, then quantizing the normalized R and B coordinates into N bins each. The histogram is represented as a 60 dimensional feature, with 20 for each channel.

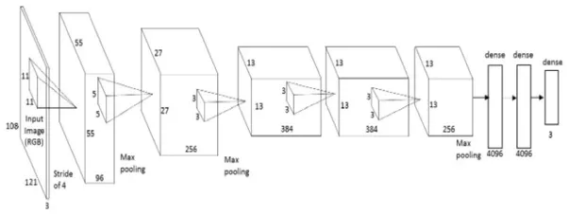

Figure 3. Architecture of CNN

3.4. Implementation

3.4.1 Support Vector Machine

Using the features extracted above as input, we perform image classification to categorize each image into the city where it was taken. The first model takes in 960 dimensional GIST descriptor of an image labelled by the city as input, and passes it to a Support Vector Machine which returns a class label. The second model takes in 60 dimensional RGB color histogram and also uses an SVM for training. For color histograms, SVM with RBF kernel was found to capture the necessary features and work well [CHV99]. For both the GIST features as well as the color histogram, we found that an RBF kernel worked best.

We then used 5-fold cross validation to tune the hyperparameters of both the models. The hyperparameters for color histogram were tuned over the range of $1, 100$ for C and $10^-6, 10^6$ for gamma and the best hyperparameters were found to be $C = 1$ and $gamma = 10^-6$ for the best model.

3.4.2 Convolutional Neural network

For the third model, we used a small modification of the AlexNet architecture by using images of 121x101 instead of on the images after scaling them down to 121x101 [KSH12] [Dam]. The architecture of the network is illustrated in the Figure. Before feeding the images to the network, they were normalized and scaled down to 121x101 from 1280x1024 and stored in Hierarchical Data Format file. It has 5 convolutional layers, 3 maxpool layers and three fully connected layers, the last of which gives the class scores

3.4.3 Linear Regression

To pinpoint the location at a finer level of accuracy than just naming the cities, we predicted the location using a linear regression model onto the coordinates. The intuition behind this was that the coordinates also provide information about the image because of the way cities are structured. For example, going further south could lead you to the financial district in Pittsburgh, meaning that an image containing a cluster of office buildings could be predicted to be in the financial district with a reasonable level of confidence.

Haversine Distance: Since latitude and longitude are not defined on the Cartesian coordinate system, Euclidean distance can no longer be used. We used the Haversine distance metric to compute how far away the predicted coordinates were from the actual coordinates of the image. Our loss function for regression was also modeled and optimized using the same metric. For two coordinates (ϕ_1, λ_1) and (ϕ_2, λ_2), the Haversine distance d_r is given by:

$$d_r = 2{*}r{*}\arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2-\phi_1}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$$

However, for the linear regression model trained on the entire dataset, we found that the average distance between the predicted location and the true location was around 52 kilometres on our best model, and almost 200 kilometres on the worst, which is unacceptably high. To bring this error down, we developed a pipelined architecture that substantially improves our results.

3.5. Pipeline

3.5.1 Classification

As described above, we first train an SVM on all the images in the dataset, which contains multiple viewpoints of the same location. Since an image represents only a single viewpoint of a location, we trained the model multiple times by excluding certain viewpoints to find out what worked best. We found that viewpoint 5, which is an upward angle, contributes the least information, and chose to omit it in the final trained model.

In the prediction phase, the query image is given to the trained models. GIST features are extracted from the query image for the SVM trained on GIST features, color histogram is computed for the second SVM, and raw pixels are passed to the CNN. The classifier then assigns a city to each query image.

3.5.2 Regression

Once the classifier has predicted which city the image belongs to, we train a linear regression model, but only on the subset of the training data which contains images from the predicted city. We only use the two SVM models here however, and not the CNN.

We then performed regression on the coordinates of the image location to predict an exact location for the query image. To measure how close the predicted location is to the true location, we continue to use the Haversine distance described above. After reducing the training data, the model was focused on a much smaller region, which reduced our distance error by nearly half.

3.5.3 Ensemble

After getting the predicted coordinates for an image from both models, we combined the predictions by taking the av-

Model	View 0	View 1	View 2	View 3	View 4	View 5	All	All but 5
GIST + SVM	87.09	93.1	87.9	93.6	94.13	56.2	83.4	93.3
Color histogram + SVM	88.5	88.3	87.2	90.5	94.4	56.2	95.1	90.67
CNN	-	-	-	-	-	-	98.49	-

Table 1. Classification accuracy

erage. We then computed the Haversine distance on the averaged predictions. However, we found that this didn't make a large difference as the Haversine distance error remained more or less the same.

4. Experiments

4.1. Classification Accuracy

We partitioned our data into a train set and a test set using an 80-20 split. We then trained three models on the train set, first with all views, and then by excluding view 5. We established the hyper-parameters through cross-validation and made predictions on the test set. The predictions are evaluated using an accuracy score with respect to true labels. The observed classification accuracy on the test set is reported in Table 1.

We also experimented with subsets of the data containing individual viewpoints to see how much information is contributed by each view. For color histogram, it is seen that the view with most information is the view 4, with 94.1%. View 0- view 2 do not provide much information on their own but still provide good accuracy. View 3 is considerably good compared to the others, with an accuracy of 90.5%. View 5, which points to the sky has the least information. It has the least accuracy of 56%. With all views, it has the best result of 95.1%. GIST features achieve similar results for the individual views, but aren't as effective as color histogram.

The convolutional neural network was trained with all views and achieved an accuracy of 98.49. Due to resource and time constraints, we could not run the classifier on all views separately. However we believe that once we are able to obtain the relative significance of each view, i.e. if we know how much information is present in each view, our accuracy can be improved further.

On observing the confusion matrix for classification, we notice that Orlando is almost always classified correctly. Pittsburgh sometimes gets confused with Orlando and sometimes with Manhattan. Manhattan sometimes gets misclassified as Pittsburgh as well but not Orlando.

Our classification accuracy is well over 90% for all models, and is the highest for the CNN classifier. This result is mainly because our dataset is restricted to three cities which are quite different from each other. Also, our dataset only has images from Google Streetview, and not photos taken 'in the wild'. If the target geographical area had included rural areas such as in previous works, the problem would have been much more difficult.

Model	Non-pipelined	Pipelined
GIST	237.25	120.055
Color hist	231.94	68.44

Table 2. Haversine distance

Model	Score
GIST + SVM	4927.2
Color hist + SVM	4684
Humans	3881

Table 3. GeoGuessr Scores

4.2. Haversine Distance

After classification, we predict the coordinates of the image location using a model trained only on the city that the image was classified into. We evaluate the accuracy of this prediction by computing the Haversine distance between the predicted and true locations.

Our initial non-pipelined approach uses the entire dataset to predict location, which is less effective than the pipelined approach because this gives a larger range of possible coordinates. For the three cities in our dataset, the mean distance achieved by each model using both approaches is recorded in Table 2.

4.3. GeoGuessr Scores

We made the system compete on the online game GeoGuessr against two humans. Given a random Streetview image from the three cities of Manhattan, Orlando, and Pittsburgh, we input the predictions of the models into GeoGuessr. We also present the same image to three human subjects, and enter their guesses into GeoGuessr as well. While the human subjects haven't seen the entire training data, they can exploit features such as signboards and other markers. The humans are also allowed to walk around while the system is only allowed a single view. The average score out of 5000 with 10 query images is as shown in Table 3.

5. Future work

For future work, we could extend the system to classify cities besides New York, Pittsburgh and Orlando. Unfortunately, there aren't many good publicly available datasets for other cities. Building a dataset in itself could be a worthwile project.

Given the high accuracy when using a CNN, we could extend the pipeline by choosing a classification based on votes from the three classification models. This might reduce the overall error even more.

6. Conclusion

Given an image, we find that classifying it into cities is an easier problem than regression to predict real coordinates. We find that our pipelined architecture of training different models on different cities, first predicting the city by classifying the query image, and then predicting the coordinates on the specific models of each does significantly better than a naive regression approach, where we train a single model on all images from all cities to predict coordinates. However, we discovered that our ensemble approach of pipelining on two features, by classifying using one model, and using those labels to perform regression using two different models trained on two features(GIST and Color Histogram), and averaging the two predicted values did not provide a significant boost on the accuracy.

References

[CHV99] Olivier Chapelle, Patrick Haffner, and Vladimir Vapnik. "Support vector machines for histogram-based image classification". In: *IEEE Trans. Neural Networks* 10.5 (1999), pp. 1055–1064.

[OT01] Aude Oliva and Antonio Torralba. "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope". In: *Int. J. Comput. Vision* 42.3 (May 2001), pp. 145–175. ISSN: 0920-5691. DOI: 10.1023/A:1011139631724. URL: https://doi.org/10.1023/A:1011139631724.

[HE08] James Hays and Alexei A. Efros. "im2gps: estimating geographic information from a single image". In: *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2008.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: (2012). Ed. by F. Pereira et al., pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[ZS14] A.R. Zamir and M. Shah. "Image Geolocalization Based on Multiple Nearest Neighbor Feature Matching using Generalized Graphs". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PP.99 (2014), pp. 1–1. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2014.2299799.

[Ara+16] R. Arandjelović et al. "NetVLAD: CNN architecture for weakly supervised place recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[WKP16] Tobias Weyand, Ilya Kostrikov, and James Philbin. "PlaNet - Photo Geolocation with Convolutional Neural Networks". In: *European Conference on Computer Vision (ECCV)*. 2016.

[Dam] Aymeric Damien. *TFLearn Tutorial*. URL: http://tflearn.org/examples/ (visited on 09/30/2010).

[HW] David Hershey and Blake Wulfe. "Recognizing cities from Streetview images". In: *CS231n course project reports (2016)* (). URL: http://cs231n.stanford.edu/reports/2016/pdfs/422_Report.pdf.