Small-Scale Photo Geolocation With Feature-Based Convolutional Neural Network

Ahmed Cheema

Abstract

Can we accurately predict the location at which an image was taken using the information provided by the image's pixels? This question describes the problem of photo geolocation. Previous foundational research has approached the problem at the global scale using either classical features or convolutional networks. In this paper, I analyze images from a data set of a smaller scale, specifically the country of Spain. I adapt previously implemented photo geolocation techniques using either classical features or convolutional networks, and then I implement a model that incorporates both methodologies. I find that a convolutional network is a significant improvement over an approach that only considers classical features. Furthermore, incorporating classical features into a convolutional neural network along with an image input has a minor positive effect on performance. These findings suggest that classical features do not have significant utility in the problem of small-scale photo geolocation when convolutional neural networks are available as an option.

1 Introduction

Photo geolocation is described as the task of determining an input photo's location using just the image's content itself.[2] The photo geolocation problem is a familiar computer vision task due to the proof of concept provided by human example[1] and the value that a viable solution would hold. The accurate geotagging provided by a viable photo geolocation solution could be used for secondary computer vision tasks such as object recognition[2] and could even benefit response time for emergency aid.[6]

Notable past approaches to the photo geolocation problem include IM2GPS[2] and PlaNet.[9] Both papers leveraged a vast global data set featuring both urban and rural photos - they differed in predictive model framework. While the authors of the IM2GPS paper employed feature descriptors to predict an image's geolocation, the authors of the PlaNet paper instead depended upon a convolutional neural network for this task. Both approaches achieved state-of-the-art success at the time of their publishing.[2,9]

In the context of prior literature, this project contributes a slightly different framework to the photo geolocation problem. Instead of a large-scale outlook, the analysis will be limited to the national level, specifically the country of Spain (mainland). The motivation for this choice is based upon the assumption that image variability increases over larger distances - identifying visual differences between north and south Spain should be a more difficult task than identifying visual differences between North America and South America. Thus, it is my intention to determine how well a photo geolocation model can perform at a smaller scale where geographic and biological variance is lower.

Furthermore, the model framework also differs from those of the IM2GPS and PlaNet studies. In this paper, I will combining aspects of both models by training a convolutional neural network that incorporates various global feature descriptors. While doing so, I will also explore the relative performance between both techniques (i.e. only using feature descriptors or only using a convolutional neural network) and compare it to the performance of the final model.

The final model takes in an input photo of fixed size and calculates certain global feature descriptors for that image. Then, the image passes through a series of convolutional and pooling layers. Afterwards, each of the feature descriptors passes through fully-connected layers yielding an output fully-connected layer for each input. These layers are concatenated together into one fully-connected layer for all inputs which is then passed through final layers and results in an output of a predicted longitude and latitude.

2 Methodology

2.1 Data Collection & Preparation

The project began with the collection of 13,099 RGB images of mainland Spain from the Google Street View Static API. Mainland Spain was divided into 21 grids, and 1000 images were randomly sampled from each full-size grid with a proportional amount of image sampled from smaller grids based on their relative area. Each of the sampled images had size 600x600x3 and contained a copyright signature in the bottom right corner. If the photograph was taken by an official Google Street View vehicle equipped with a 3-D camera, the copyright signature simply reads "Google." However, the data set also contains images taken by regular people who upload the photos onto the Internet - these are marked with the photographer's name rather than "Google." With this convenient signature, I was able to classify the source of all 13,099 images as being the Google Street View vehicle or not. With the input data fully prepared at this point, the calculation of feature descriptors could begin.

2.2 Feature Calculation

The next step was to calculate the feature descriptors that were found to have some predictive value for geolocation in the original IM2GPS paper.[2]

First, "tiny images" can be obtained by downscaling an image into a much lower resolution. Despite their low resolution, past research has revealed that downscaled images maintain high recognizability while reducing computational expense.[8] For our purposes, I downscaled each 600x600x3 RGB image twice - once into a 16x16x3 RGB image, and again into a 5x5 CIELAB image. The CIELAB color space is device-independent, meaning that the values used to produce a color will have the same output through any device. This characteristic is not shared by the RGB color

space.[3] Furthermore, the CIELAB color space is based upon the human perception of color, so it has held particular value in some computer vision applications.[5] Thus, I also constructed a color histogram in the CIELAB color space for each image. The L* (lightness) dimension contained four bins, while the a* (green-red) and b* (blue-yellow) axes feature 14 bins, yielding a histogram with 784 dimensions for each image.

Next, I calculated a histogram of textons for each image. Calculating this feature required the creation of a universal texton dictionary. First, I applied a filter bank (a steerable pyramid with four scales and two orientations) to four randomly selected images from each grid (the 21 grids dividing mainland Spain) of our image data set, thus providing 84 total training images. I collected he filter responses for each image, yielding a 360000x8 data matrix because each image has 360,000 pixels. With 84 training images and their individual data matrices concatenated, we obtain a 30240000x8 data matrix. Then, I applied K-means clustering to find 32 cluster centers within this data. Once the 32 cluster centers are obtained, I applied the filter bank to all 13,099 images in the data set and captured the 1x8 vector for each pixel representing the filter responses in each band of the steerable pyramid. I assigned each of these vectors to one of the 32 cluster centers, essentially yielding a 600x600 matrix of cluster labels for each image, which was then converted into a 1x32 vector representing texton histogram counts.

Another feature of interest was the statistics pertaining to detected straight lines within the image. Straight lines were detected using the methodology presented in Video Compass.[4] First, I identified edges in the grayscale version of an image with the Canny detector. Then, I found the connected components of the binary edge matrix and iterated through them. All connected components of length less than 5% of the image size were discarded. As described in Video Compass,[4] the best fit line was computed for the remaining connected components and lines with satisfactory fit quality remained. Using the parameters of the best fit line for these connected components, I obtained line angle and length. After completing this process for all connected components within a single image, I computed histogram count vectors with 17 bins for both line angle and line length.

Finally, I calculated the gist descriptor[7] for all input images in the data set. The gist descriptor is a global feature descriptor that has been found to perform well in scene classification.[2] I calculated it for each image by applying a bank of Gabor filters with four scales and six orientations to each image, yielding 24 bands. I divided each of these patches into a 4x4 grid (16 grids of size 150x150 in the case of a 600x600 input image) and calculated the mean filter response in each cell. Completing this process for all 24 patches means that the total output for a single image is a vector of length 384, which we call the gist descriptor.

At this point, each input image had the following corresponding features calculated: a 16x16 RGB tiny image, 5x5 CIELAB tiny image, CIELAB color histogram, texton histogram, line feature histogram, and gist descriptor for all 13,099 input images.

2.3 Geolocation with features

The next step is to use these features to predict an image's geolocation. First, I randomly split the 13,099 images into a training set and testing set with 80% of the images in the training set. Then, the feature differences for each testing image were calculated with each training image. In other words, I calculated a 7x10479 matrix for each image in the testing set with each row representing a feature and each column representing a training image. This process was completed for all 2620 testing images, thus yielding a matrix of size 2620x7x10479. Then, the cell at $4, 2, 5$ represents the distance between the gist descriptor of the fourth testing image and the fifth training image.

The computation of feature distances was completed as described in the IM2GPS paper.[2] χ^2 distance was used to compare the color and texton histograms, L1 distance was used for line features, and L2 distance was used to compute feature distances for gist descriptors and tiny images. All feature distances were standardized so that their influence on predictions are equal.[2]

For each testing image, I found the training image for which the mean feature distance was minimized and considered the location of that training image the predicted geolocation. Then, I approximated the distance between the true location and the predicted location using the Haversine formula.

2.4 Convolutional neural network without classical features

Next, I trained a convolutional network that takes in an input image of shape 300x300x3 (the original 600x600x3 images were downscaled due to memory limitations) and passes it through a series of convolution and pooling layers.

First, the image passes through a convolution layer with 64 filters of size 7, followed by a pooling layer.

Then, the image passes through three sets of layers, each following a pattern of C-C-P (convolution, convolution, pooling) with the number of filters varying from 64 to 256 and the kernel size varying from 1 to 5.

Finally, the output is flattened and passed through non-linear fully-connected layers with 64 and 32 units respectively before the final fully-connected layer with 2 units. This final output represents the prediction, an array containing the predicted latitude and longitude values.

I use a loss function of mean squared geodesic error (the mean squared approximated distance between the predicted and true locations - the approximation is based on the Haversine formula). Three main metrics are used for evaluation: mean error (km), mean squared error (km^2), and the percentage of predictions with an error of less than 200 kilometers.

2.5 Convolutional neural network with classical features

The final model takes in eight inputs:

- Image of shape 300x300x3
- Color histogram vector of length 784
- Gist descriptor vector of length 384
- Texton histogram vector of length 32
- Line angle vector of length 17
- Line length vector of length 17
- Tiny RGB image vector of length 768
- Tiny LAB image vector of length 75

Notice that while the image takes the form of a matrix, each of the feature descriptor inputs have been flattened into vectors. This pre-processing flattening step is necessary because the non-image inputs are not passed into the convolution layers. Rather, they are only incorporated into the fully-connected stage of the neural network, at which point they must have a one-dimensional shape.

The model is identical to the aforementioned convolutional neural network without the classical features up to the point at which the output of the convolution layers is flattened. Then, each of the seven non-image inputs are passed into fully-connected layers and their outputs are concatenated to that of the image layer. With eight fully-connected layers consisting of 128 units, the concatenated layer has shape 1024. Similarly to the previous model, this output passes through final non-linear fully-connected layers until it outputs a predicted latitude and longitude.

The same loss function and evaluation metrics from the previous convolutional neural network are used.

3 Results

After training, I used each model to make predictions using the testing data set. Predictions outside of the bounds of mainland Spain are mapped to the nearest point along the border of Spain. The three evaluation metrics for these predictions for each model are shown in Table 1.

Note that the "Random" approach involved picking a random training image for each test image and using its location as the prediction.

Each of the three models outperforms the random approach. The increase in performance is most drastic for the two convolutional neural networks. The model that performs the best in all three metrics is the feature-based convolutional neural network.

Table 1: Model performance comparison

	MAE (km)	MSE (km^2)	< 200 km
Random	385.0	184658	0.183
Feature-Based	328.6	154317	0.288
Basic CNN	277.2	97370	0.328
Feature-Based CNN	270.5	92272	0.350

The optimal model has an average prediction error of 270.5 kilometers and predicts an image's location within 200 kilometers approximately 35% of the time. The optimal model's mean prediction error marks a 29.7% decrease from the random approach and a 17.7% decrease from the feature-based approach.

The predicted locations for the testing set of the optimal feature-based CNN are shown in Figure 1.

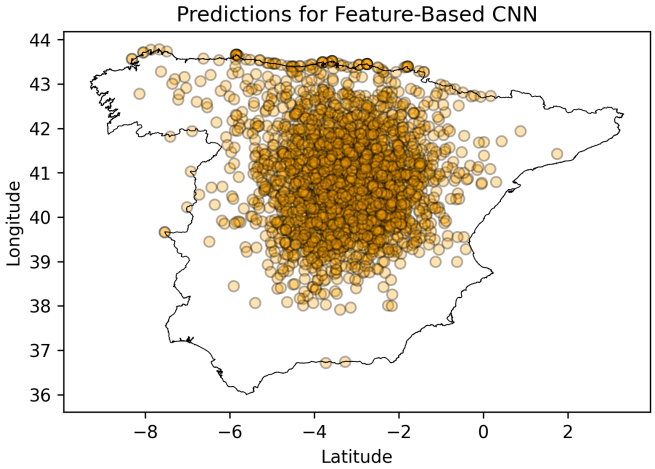

Figure 1: Predicted locations of the feature-based convolutional network

Notice that the vast majority of points are clumped in a round cluster in central Spain. There are a few points scattered around other regions in Spain, specifically towards the northwest. There is also a distinct line of predictions along the north border of mainland Spain, representing predictions that were north of Spain entirely and were clipped to the border.

4 Discussion

4.1 Challenges & Limitations

While significant progress was made in addressing the problem of small-scale photo geolocation, I ran into a few challenges, some of which were not overcome.

While I attempted to replicate the features used in the IM2GPS paper, I was unable to do so for one of its features: geometric context. It is described as the geometric class probabilities for the ground, sky, and vertical image regions.[2] I was not able to implement this myself and it has been disregarded from this analysis altogether. However, the IM2GPS paper found in its initial evaluation that the geometric context feature had poor predictive power and it was subsequently discarded.[2] Thus, I do not think its omission from this analysis is particularly worrisome. Although, a few challenges were also encountered in features that were able to be calculated.

While the filter bank in the texton histogram process described by IM2GPS used eight orientations, two scales, and two elongations, such a granular filter bank was not possible in my case due to computational limitations. I was forced to limit the filter bank to a steerable pyramid of four orientations and two scales. Similarly, the universal texton dictionary was formed on just 84 training images due to memory limitations - while it is not clear how

many images were used to form the universal texton dictionary in IM2GPS, I expect that a sample of 84 is not optimal.

Additionally, my implementation of the straight line detector is not quite as robust as desired. It sometimes detects straight lines in cases where it shouldn't (such as clouds) and attempting to resolve the issue with a stricter threshold goes too far in the other extreme by not detecting enough of the true straight lines. An example of the line detector not achieving the desired results is in Figure 2. Notice that four lines are detected in the sky. Furthermore, multiple straight lines are detected in the edges of shadows rather than actual physical objects, which is an inevitability addressed in the Video Compass paper from which the line detection process is described.[4]

Figure 2: Detected straight lines in an input image

4.2 Conclusion

In summary, I implemented and analyzed three models for small-scale photo geolocation: a 1-NN approach utilizing classical features, a convolutional neural network taking in a single input image, and a convolutional neural network taking in an input image along with seven classical features as separate inputs.

I found that all of the models outperformed the approach of random predictions, but their respective performance varied. Specifically, the feature-based 1-NN approach exhibited the worst performance as both convolutional networks were massive improvements over the feature-based model.

A comparison between both CNN implementations reveals that the incorporation of feature descriptors did yield a slight improvement in performance. However, this increase in accuracy is not significant enough to confidently draw any conclusions and further analysis is necessary to confirm the value (or lack thereof) of classical features as secondary inputs into a convolutional neural network for photo geolocation. In any case, it is clear that convolutional neural networks are a significant improvement over a feature-based approach.

However, it should be noted that even the optimized convolutional neural networks do not achieve accuracy anywhere near the level necessary to be used as a means for geotagging. A mean error of approximately 270 kilometers is still significant (Figure 3) even if it's a clear improvement over the random approach.

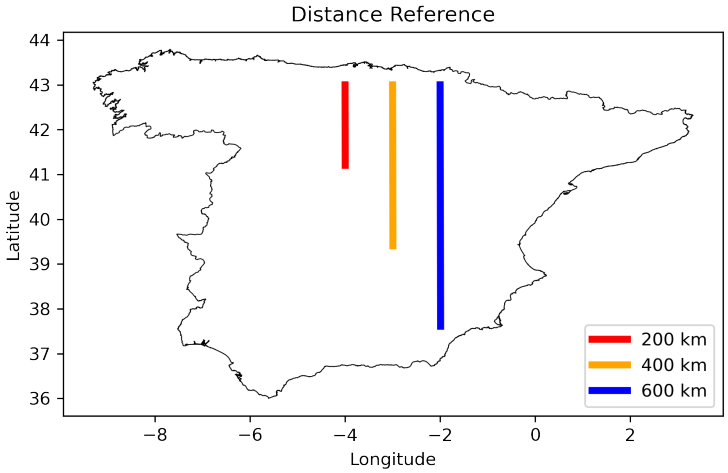

Figure 3: Reference for distance values relative to mainland Spain

Attempting to use these models for precise geotagging would yield extremely misleading results and should thus be avoided. A more practical use would be to tweak the convolutional neural network so as to output a probability distribution rather than a single value, which would allow for a degree of uncertainty and could be used as a prior for other computer vision tasks such as object recognition.

Self-Acknowledgement

Note: Various passages in this paper were reused from the progress report for this project written by Ahmed Cheema.

References

[1] Browning, Kellen. “Siberia or Japan? Expert Google Maps Players Can Tell at a Glimpse.” The New York Times, The New York Times, 7 July 2022, https://www.nytimes.com/2022/07/07/business/geoguessr-google-maps.html.

[2] Hays, James, and Alexei A. Efros. ”IM2GPS: estimating geographic information from a single image.” Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition. IEEE, 2008.

[3] Kaur, Amanpreet, and B. V. Kranthi. ”Comparison between YCbCr color space and CIELab color space for skin color segmentation.” International Journal of Applied Information Systems 3.4 (2012): 30-33.

[4] Košecká, Jana, and Wei Zhang. ”Video compass.” European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2002.

[5] Krieger, Louis WMM, et al. ”Method for improving skin color accuracy of three-dimensional printed training models for early pressure ulcer recognition.” Innovations and Emerging Technologies in Wound Care. Academic Press, 2020. 245-279.

[6] Murgese, Fabio, et al. ”Automatic Outdoor Image Geolocation with Focal Modulation Networks.” (2022).

[7] Oliva, Aude, and Antonio Torralba. ”Modeling the shape of the scene: A holistic representation of the spatial envelope.” International Journal of Computer Vision 42.3 (2001): 145-175.

[8] Torralba, Antonio, Rob Fergus, and William T. Freeman. ”Tiny images.” (2007).

[9] Weyand, Tobias, Ilya Kostrikov, and James Philbin. ”Planet-photo geolocation with convolutional neural networks.” European Conference on Computer Vision. Springer, Cham, 2016.