Predicting Geographic Location in the Continental United States Using Convolutional Neural Networks

Anuraag Warudkar
The William States Lee College of Engineering
University of North Carolina at Charlotte
Charlotte, United States
awarudka@uncc.edu

Tasha Asyiqin
The William States Lee College of Engineering
University of North Carolina at Charlotte
Charlotte, United States
tasyiqin@uncc.edu

***Abstract*—This paper shows a novel approach to geographic location prediction within the US using a Convolutional Neural Network. We set out to make a model that could play GeoGeussr, an online geography game, within a ~50 mile tolerance, given a single low quality Google Street View API image. If successful, this research could enhancing location-based services, and serve as a educational training tool for the GeoGeussr game. In this paper, we go through the challenges of generating a high quality dataset, creating a simple CNN architecture, and training the model using our own custom Haversine loss function. The resulting model performed fairly well on the GeoGuessr map, but failed to accurately guess the correct region of the country on real world test images. This paper concludes with insights into the shortcomings of CNNs for complicated image recognition tasks and and recommendations for future undertakers of such a project. Some key points were made about switching to a ResNet based model, using a higher quality dataset, and using data augmentation to create extra samples.**

Keywords—Convolutional Neural Networks (CNNs), Geographic Location Prediction, Computer Vision, Image Processing, Continental United States, Deep Learning, Geospatial Analysis, Google Street View API.

I. Introduction and Motivation

The reliable and accurate location and classification of images (by country, or by environment) is a very useful tool for many industries. Creating a model that could region-guess the United States with just one image (a low quality one at that) has a lot of real world uses. In addition, it could serve as a good training tool for the GeoGuessr, an online game in which users attempt to guess where in the world they are given a Google Street View location. Despite the immense potential, the task of latitude/longitude prediction from images is extremely challenging. It requires the model to learn complex spatial relationships, recognize landscapes, and discern contextual clues embedded within images. Convolutional Neural Networks (CNNs) have emerged as a powerful tool for image analysis, making them a natural choice for this task. However, developing a CNN-based model that can reliably predict geographic locations within the vast and diverse continental United States presents a unique set of challenges.

In this report, we delve into the methodologies and techniques employed to tackle these challenges. We discuss the architecture of our CNN model, acquiring the dataset (images), and the strategies for training and evaluation. Through rigorous experimentation and testing, we evaluate the performance of our approach and showcase the potential of deep learning in geospatial analysis.
The subsequent sections of this report will provide an in-depth exploration of our methodology, experimental results, and the implications of our findings. By the end of this report, readers will gain a comprehensive understanding of our CNN-based geolocation prediction model and its relevance in the broader context of computer vision and geographic information systems.

II. Approach

A. *Acquiring the Dataset*

Gathering a dataset appropriate for the training of this model is a very difficult task, as none currently exist. The only way to quickly and reliably get images from all around the US is to use the Google Static Street View API. The downside, however, is that the images are limited to 640x480, and the resulting image is really grainy. In addition, there is no way to ensure that the image actually contains useful information to determine the state. In some cases, the image was facing a tree, and nothing else was visible. These kinds of images aren't helpful for the model, and the inability to filter these images at a large scale put a damper on the performance of the CNN.

B. *Model Architecture*

Our model was based on the CIFAR-10 model, which classifies 32x32 images into 10 classes, including truck, ship, and many others. It contained convolution layers, pooling layers, used ReLU for the activation function, and the last 2 fully connected layers had dropout to reduce the chance of our model overfitting. The output layer just consisted of 2 tensors, one that predicted latitude and one to predict longitude.

C. *Evaluating the Model*

The evaluation of the model was done using the Haversine function, which returns the approximate distance in miles between two latitude/longitude coordinate pairs. It also played 2 games on GeoGuessr (particularly the "A Rural USA" map), which was most relevant to the data it was trained on.

III. Dataset and Training Setup

A. Finding Coordinates for the Street View API

In order to find coordinates to input into the street view api, we had to find coordinates of every road in the US. Since google only has coverage on roads, randomly selecting coordinates in a state resulted in a very poor valid locations to attempted coordinates ratio. We began by writing a Python script to combine county road files provided by a government website (TIGER) into a single Shapefile per state. Once this was complete, we could extract a point on every nth road, depending on how many locations we wanted to collect in that state.

B. Training and Preprocessing

The images were saved in a folder, and a CSV file containing all the labels (filename, latitude, longitude) was generated. The images were also normalized based on the average mean and std of every image in the training dataset. The dataset was split for training and testing, at a 10:1 ratio.

C. Training Parameters

The optimizer, loss function, learning rate, and batch size had to be iteratively adjusted based on results from the previous training session and parameters. We settled on SGD at a learning rate of 0.01 and momentum of 0.9, due to its higher rate of convergence in image related tasks. Batch size was limited by our GPU memory, and stayed constant at 32 throughout our training. Instead of using a typical MSE (Mean Standard Error) loss function, which doesn't capture the geographical distance between the predicted and target coordinate, we implemented a Haversine loss function, which returned the average distance divided as 3000. This loss function performed slightly better than the traditional MSE loss function.

IV. Results and Analysis

The final model that we settled on had approximately 39,000,000 parameters, and performed pretty well on the map "A Rural USA" GeoGuessr. The final validation loss was 355 miles, and the bot was approximately 1500 miles away on average on two 5 round games on the aforementioned map. However, it seemed like the model was just guessing in the middle of the country, in Missouri, Kansas, Oklahoma, Colorado, and never venturing further than West Virginia. This seems to suggest an imbalanced dataset, or an error at some other point during the training procedure. However, the locations in the map are also biased towards the center of the United States, so this works out quite well for predicting locations. In real world performance, however, this is far from ideal. After putting together a small test set of images in Maine, the bot repeatedly made guesses nearby and around Kentucky. There are many causes for this high prediction loss, which need to be investigated if this project is ever revisited by another party. Due to the difficulty of the dataset collection step, it was very difficult to get an equal amount of images per state, leading to an imbalance between the locations in smaller and more sparse states, such as Maine, compared to populated states such as Texas. The images were also of low quality, and sometimes the model had no important information to glean from the image. Another potential issue is the validation set used to evaluate the performance of the model. It was similarly imbalanced, although to a far higher degree than the training set. This could explain our low validation loss compared to the real world performance of the model. If all the validation data existed in Texas or big states in the center of the state, then that's the optimal loss that we would see at the end of training.

Lessons Learned

There are a lot of takeaways from this project and resulting model. We underestimated the complexity of the task, and didn't ever make the leap to large ResNet-based models, with more convolution layers and parameters. The simple CNN that we used wasn't able to understand the patterns in the data and essentially was reduced to guessing in the middle of the country on almost every location. There were a lot of issues during our initial dataset collection, and these could be remedied by collecting an equal amount of images per state, no matter the sparseness or size, and increasing the volume of images in our dataset. Additionally, data augmentation techniques, such as scaling or rotation could be used to add variability into the dataset, helping to prevent overfitting and increase the number of data points for imbalanced states. In conclusion, even though this project produced decent results for the GeoGuessr map it was trained on, the real world performance is significantly lacking. Using the above methods and using a ResNet model instead of a simple CNN would probably have yielded far better results.