CS230

Predicting Countries in Southeast Asia using Convolutional Neural Networks

Samuel Chian
Institute for Computational and Mathematical Engineering (ICME)
Stanford University
samchian@stanford.edu

Mishal Mrinal
Institute for Computational and Mathematical Engineering (ICME)
Stanford University
mishal5@stanford.edu

Abstract

This paper introduces a CNN model used to predict Southeast Asian countries using images scraped from Google Street View API. We introduce a grid-based method to train an 18 layer ResNet on this problem that managed to achieve an accuracy of 90% on test and validation sets of size 1000 after being trained on 38000 images of Singapore, Thailand, Cambodia and Malaysia for 5 epochs (after regularization by early stopping) using minibatch gradient descent with momentum.

1 Introduction

Geoguessr is a geographical browser game where players guess locations of images from Google street view images. In particular, there exists game modes where the human player is unable to move the camera upon viewing the image, making it harder to guess where the exact location is as they are unable to search around for external information such as languages of signboards/roads. In the game, there are also various scoring systems: (1) Correct classification of country, or (2) Latitude-Longitude-based scoring.

As avid Geoguessr players, we have often watched the world championships of Geoguessr and have always been fascinated by how the top Geoguessr players have different strategies to get extremely accurate results, even in the latitude-longitude mode. As such, it is of particular interest to us to see whether it is possible to build a model that could perform on par, or even better than human players in this no-move, latitude-longitude mode as we believe that this is the hardest mode of the game.

Beyond just playing the game, there exists other opportunities where identifying exact locations of a picture might be useful. For example, one might want to do automatic geotagging of locations based off the image, or even identifying where a missing person might be located if an image of their background can be found.

To build this model, we hope to use Google street view images as the input to the model, we then hope to use a Convolutional Neural Network (CNN) to first see if we can classify a country correctly based on the image, and possibly output where in the country the picture came from.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Project Novelty

2.1 Application

One area of project novelty that our project has lies in the problem in which we are trying to solve. While computer vision classification tasks are often solving to identify objects that are pretty similar between images (e.g. is an object a dog?), this task involves learning the different geological features of the region (i.e. plants, buildings, etc.) to learn about whether an image belongs to a certain country.

Previous work in this area has tried to do this task on areas that are geologically very different (e.g. Denmark vs. USA), which are easier tasks due to the difference in architecture. Furthermore, they are generally done on locations that are easy to distinguish such as city areas. However, we hope to solve this on a region with a lot more similarities between countries, and without only choosing specific parts of the country that are easy to classify.

2.2 Method

Previous work that has tried to do a location-specific prediction is generally limited to a single country such as the USA. However, we wanted to extend this idea to multiple countries as well. To get the location-specific prediction, we split each country into a set of 2,500 grids (for a total of 10,000 grids)[1], and assign a unique index to the grid location. Therefore, we can see if we can predict the actual location of the image based on a grid (see Figure below for an illustration).[2]

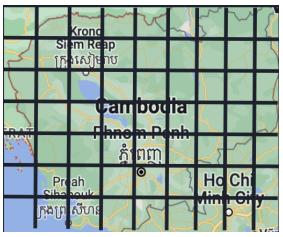

Figure 1: Illustration of Grid

3 Dataset and Features

For our preliminary results we scraped 10,000 images each for Cambodia, Singapore, Thailand and Malaysia from Google Street View API. This was done by first creating a bounding box around the particular countries, and uniformly sampling latitudes and longitudes within this bounding box. As Google Street View API does not have images for every latitude-longitude pair, it might lead to certain areas having slightly more sampling than others. However, due to the size of the sample collected, the data should be a good representation of the different landscapes across all the 4 countries.

To train, validate, and test our model, we used a training set of 38000 images (evenly split amongst all countries), and used a dev and test set of 1000 each (consisting of an even split of all countries). Some examples of the dataset images are found below:

4 Experiments/Results/Discussion

As the classes were very evenly balanced due to our manual data scraping, we mainly focused on looking at the accuracy of our results as ultimately we care the most about getting the prediction correct for the game.

[1] Ultimately, this left quite a few classes empty due to the fact that Google API only has street view in certain parts of countries, and so the actual number of grids with images in them was 251

[2] We also attempted to implement a custom loss function was attempted to penalize country and latitude-longitude. However, due to the time-constraints, we could not implement this custom loss function successfully

Figure 2: A Google street view image in Singapore

Figure 3: A Google street view image in Cambodia

4.1 Country Classification Task

In this task, we mainly work on the task of classification of the country based on cross entropy loss.

4.1.1 Baseline

For our baseline model, we trained a convolutional neural network as we are working with image data. For the architecture of our baseline model, we went with the following:

$$CONV1 \rightarrow POOL1 \rightarrow CONV2 \rightarrow FC1 \rightarrow FC2 \rightarrow FC3 \rightarrow OUTPUT$$

We chose this architecture as we believe it is a fairly basic model that would give us a good target to try to beat. For our training we used mini batch gradient descent on a batch size of 32 with momentum ($\beta = 0.9$) and learning rate 0.001, and cross entropy loss as our loss function. Since we were training a baseline model, the choice of using $\beta = 0.9$ seemed reasonable as this is regarded as a standard setting for β. We used a learning rate of 0.001 as this appeared to give us a reasonable improvement across epochs. We generally stopped by 10 epochs as the results did not seem to be improving.

4.1.2 Baseline Results

Using our baseline model, we first tried the classification task on two countries that had very different geological features: Singapore, and Cambodia, to see the results on an easier task. When we did this, we managed to obtain a 96% training accuracy, 97% validation accuracy and 95% test accuracy. As it seemed like a basic model could pick up the differences between a more rural country like Cambodia and a more urban country like Singapore (i.e. see the two images above for reference), we added two additional countries that have more similar geological features: Malaysia and Thailand, into the mix.

For this harder classification task, our baseline model we had a training accuracy of 72 %, a validation accuracy of 69% and a test accuracy of 72% after 10 epochs. While the training accuracy was still going up, the test and validation accuracy plateaued and thus we decided that minimizing the loss further will not improve the predictions.

4.1.3 Model Improvements

While the baseline model had done much better than just random guessing (i.e. we only expect 25% accuracy on random guessing), we next tried much deeper neural nets, pre-trained on image data to see how much better they could perform. In particular, we tried various depths of ResNet and VGG (ResNet18, ResNet50, ResNet152, VGG11, VGG13, VGG16, VGG19). For these experiments, we also used cross entropy loss, and the same training hyperparameters for the minibatch gradient descent as the baseline.

As the models generally performed equally well, we chose the simplest model (ResNet18) for our further experiments. We also noted that there was significant overfitting going on, as the model was having extremely high training accuracy, but this did not reflect as well in the validation and tests sets. As we noticed that this generally happened before epoch 10, we generally ran until epoch 10, hoping to regularize by early-stoppage for our final model.

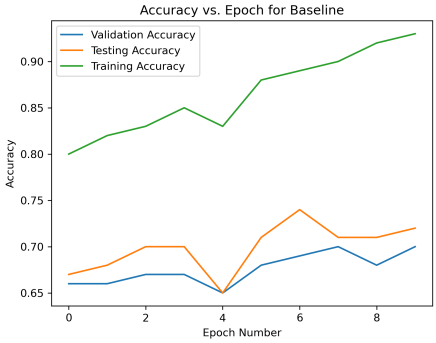

Figure 4: Training Curve for Baseline

Model	Training Accuracy	Validation Accuracy	Test Accuracy
ResNet18	99%	85%	86%
ResNet50	99%	84%	82%
ResNet152	99%	80%	86%
VGG11	99%	80%	85%
VGG13	99%	77%	85%
VGG16	99%	83%	79%
VGG19	99%	81%	83%

Table 1: Training, Validation Test Results for Deeper Models

Additionally, after selecting ResNet18 as the model that we wanted to proceed with and fine tune, we did a hyperparameter search on a much smaller sample to determine the optimal hyperparameters. These were determined to be a learning rate of 0.01, momentum at 0.9, and a mini-batch size of 16.

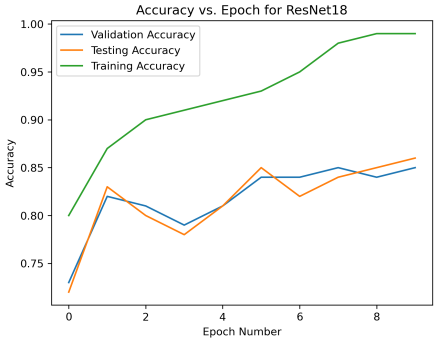

Figure 5: Training Curve for ResNet18 on Country

4.2 Location Classification Task

In this section, we focus on the task of classifying the image based on the particular grid number assigned to them, based on cross entropy loss and the tuned hyperparameters.

4.2.1 Grid-based Classification

Given that now we have 10,000 possible classes, the prediction problem became much harder. This is evident in the training curve below for 100 epochs, where we could not get reasonable results for all three subsets of the data. While we do see that training accuracy is continuing to go up, it seems like validation accuracy has already plateaued at approximately 5%. Therefore, it seems unlikely that we

would be able to achieve higher accuracy than this. While this definitely is a poor job at getting the exact location right, it is still much better than randomly guessing which would be 0.01% accuracy[3].

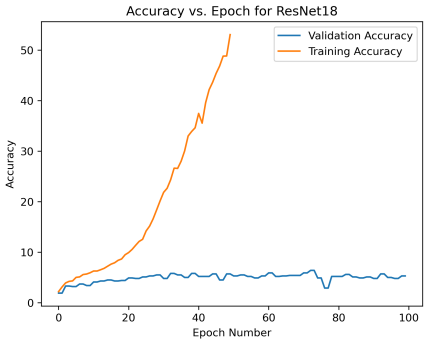

Figure 6: Training Curve for ResNet18 on Grid Number

4.2.2 Final Model

While counting the accuracy of when we get the exact grid right might not produce good results, there also existed the possibility that we could be predicting grid numbers that are closer to each other but not exactly the same, which would be missed out by this metric. Therefore, we also tried to use the grid-based prediction and map it back to the country. When we do this, we realized that we get better results than the original classification task, and got validation and training accuracy closer to 90%.

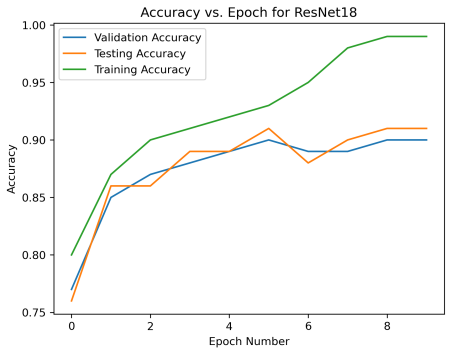

Figure 7: Training Curve for ResNet18 on Grid Number then converted to Country

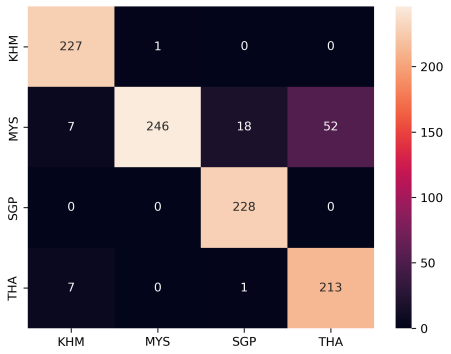

Figure 8: Confusion Matrix on the Test Set

5 Conclusion & Future Work

We weren't successful in our initial aim of trying to predict exact locations of pictures, however if we were to loosen our metric from exact guesses to rough distances from the actual location, then the model was actually able to make great progress and it was noticing this fact that gave us the idea to use the benefits of grid based classification training in order to predict countries. However, the minimal success that we were able to achieve with our grid based classification system suggests that there is some potential in being able to train a grid classifier given favorable conditions like a lot more training data, larger model architecture and longer number of training epochs.

Another potential area of future work is in implementing our custom loss function. Since we were taking the prediction of the model as the max over all the classes, there was a breaking in the computational graph of the model that didn't allow PyTorch to backpropagate when using our custom loss function. If we were able to side step this issue, it would be interesting to see the kinds of results the model is able to obtain.

[3] 0.4% accuracy if we only count classes with images in them

6 Contributions

Mishal: Worked on the final project write up and presentation slides. Scraped images for Cambodia and Thailand. Worked on code to convert grid label predictions to country predictions used in the training phase of all models. Did post-training analysis on ResNet18 model to produce confusion matrix.

Samuel: Worked on the final project write up and presentation slides. Scraped images for Singapore and Malaysia and uploaded all data onto AWS. Worked on code to preprocess data and label images according to grid labels. Wrote code for the training of the various models on AWS, worked on the hyperparameter tuning, and produced the different training graphs.

References

[1] Paszke, Adam and Gross, Sam and Massa, Francisco and Lerer, Adam and Bradbury, James and Chanan, Gregory and Killeen, Trevor and Lin, Zeming and Gimelshein, Natalia and Antiga, Luca and Desmaison, Alban and Kopf, Andreas and Yang, Edward and DeVito, Zachary and Raison, Martin and Tejani, Alykhan and Chilamkurthy, Sasank and Steiner, Benoit and Fang, Lu and Bai, Junjie and Chintala, Soumith (2019) PyTorch: An Imperative Style, High-Performance Deep Learning Library, Advances in Neural Information Processing Systems 32 8024–8035, Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015) Deep Residual Learning for Image Recognition https://arxiv.org/abs/1512.03385

[3] Karen Simonyan, Andrew Zisserman (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition https://arxiv.org/abs/1409.1556

[4] Sudharshan Suresh, Nathaniel Chodosh, Montiel Abello (2018) DeepGeo: Photo Localization with Deep Neural Network https://arxiv.org/pdf/1810.03077.pdf

[5] Hugovk (2021) Random Street View Github Repository https://github.com/hugovk/random-street-view

[6] Stelath (2021) GeoGuessr AI Github Repository https://github.com/Stelath/geoguessr-ai