Use of contrastive learning and CLIP model for Geoguessr game

Donggyu Kim(DK)

December 16, 2024

Abstract

There is a guy named Trevor Rainbolt, a media personality famous for his incredible skill in winning the Geoguessr game with remarkable accuracy, far above the average person. Geoguessr is an online game where, given a random photo from anywhere in the world, the player must try to pinpoint the location of the image as closely as possible on a world map. The closer and faster you guess the actual location where the image was taken, the more points you earn; the farther and slower your guess, the fewer points you get.

Rainbolt is known not only for his precise guesses—sometimes even pinpointing the location within a few kilometers of the actual spot—but also for his speed, often taking only a second. In many of his videos on Instagram and YouTube, he explains how he makes quick, accurate guesses. He identifies clues within the image itself, such as types of traffic lights, license plates, trash bins, roofs, trees, and more, which reveal a lot about where the photo was taken.

His approach is similar to how learning occurs in many deep learning models, where an encoder learns the semantic representation of given images. So, could it be possible to build a model that, when given a random image of a place, could beat him?

1 Introduction

In this project, my primary goal is to become familiar with the entire process of building a database, creating a model, and developing testing to evaluate both the model architecture's performance and the database's efficiency. In other words, I want to fully grasp the whole deep learning model-building procedure by doing everything from scratch (other than using pre-trained encoders and decoders). This is intended as an educational project, allowing me to learn step by step to thoroughly understand the process while also achieving meaningful results based on the papers I've read in this class. Although reading papers has been exciting and informative, I haven't done much hands-on work, so I've felt less confident about actually building or coding the concepts I've learned.

This project aims to leverage the CLIP model developed by openAI to match images with corresponding textual prompts while also classifying images by their geohashed coo

2 Dataset

The dataset was consisted of approximately 50 images of 190 capital cities in the world, excluding some areas that are restricted on the Google Maps API side, such as Beijing, Baghdad, Kabul, where countries have blocked the access to maps due to security issues. The initial dataset included images of other non-capital major cities or random locations of rural areas generated by random function and a function that checks whether or not the random coordinate was on the land or not. However, the majority of randomly generated images in the rural area included trees and grass that had no important features to be used in training process, and due to the limited GPU resources, I decided to exclude images of rural area, and created a dataset consisted only of capital cities in the world. I downloaded each image using the Google API, with coordinates randomly generated within specified bounds for each city. For example, images requested from Washington D.C. are based on coordinates within the following bounding box: "Washington D.C.": "north": 39.00, "south": 38.79, "west": -77.12, "east": -76.91. The bounding box is centered on each city's central point,with the latitude and longitude extending no more than 0.3 degrees in each direction and each box was automatically generated by ChatGPT-4o.

I plan to use images and coordinates as two modalities for contrastive learning. But later on, depending on the model performance, I may or may not use geocoding, meaning whether I should convert coordinates into addresses or descriptive names that define the location, such as city name or something more detailed.

3 Model Design

My implementation of model involves the use of contrastive learning, mainly through the use of pre-trained CLIP models. Contrastive learning is a learning process which involves the use of cosine similarity of text-image pairs. Contrastive learning learns features by comparing positive(large cosine similarity) and negative pairs (small cosine similarity) of each pairs in the selected number of images in a batch.

Each image (image of a random Google street view of capital cities) is encoded based on selected models (Resnet, and Vision Transformer), and each text, which is a prompt that says "A street of *cityname*" is encoded with the text encoder part of the clip model after each prompt is tokeized (GPT or BERT). I did not use different prompts in this case, such as "A rural area of *cityname*" or "A urban area of *cityname*" - due to lack of resources and capacity to distinguish every single image's landscape. Therefore, label (text) of each images was unified to one single prompt : "A street of *cityname*".

4 Geohash

There are various ways to pinpoint a location on the globe. The most common method is the use of numerical coordinates, designated as lat, long. Although this is a straightforward way to learn features of locations, numerical coordinates create large dimensionality due to their numeric nature. Since our goal is to beat the average human player (or possibly a genius like Rainbolt), we are trying to create a model that pinpoints the location of a given Google Street View image close within some degree of error, rather than the precise location of the image (which may be too ambitious and an overfitting goal to achieve given the size of the dataset and the computing power). To balance our constraints and errors on guessing location of an image on the map, converting regression problem which uses numerical coordinates into a classification problem could provide an insight into our project solution.

Therefore, the output of our model, which is built upon existing CLIP model(that uses 4 different backbone models, RN50, RN101, ViT-B16,ViT-B32), is replaced with classification head. The total number of existing geohashes from our dataset of 190 capital cities are 109. Therefore, the total size of classes in our classfication problem would be 109 as well.

Geohash[1] is a hashing algorithm, invented by Gustavo Niemeyer to convert geographic coordinates into a short string, combined with letters and digits.It converts numerical geographic coordinates to some hashed string to uniquely identify positions on the Earth. Geohash uses a base of 32, all digits 0-9 and all lower case letters except "a", "i", "l" and "o". Longer the hash is, the more precise the location is with the small errors. Therefore, we set length of geohash to convert regression problem into classification problem since each regions of the earth are given specific geohashed value.

Shorter geohashes cover larger areas, while longer ones denote finer resolutions. For example, the coordinate pair of 57.64911,10.40744 is hashed to u4pruydqqvj, with the use of Geohash. Geohash length of 2 has an kilometer error range of 630km, while geohash length of 8 has a kilometer error of 0.019km. To satisfy our initial project goal, combined with dataset/computing power constraint, the error distance of few hundred kilometers, which indicates some error within a country of neighboring countries, seemed to be adequate to be our goal - at least average player would get a country correct or get it entirely wrong, which would indicate an error range of at least few hundred kilometers, very close to the error range of the geohash length of 2.

5 Loss functions and Hyperparameters

The model uses the cross-entropy loss from the torch.nn package in torch. The optimizer is AdamW with a weight decay of 0.01. Due to limited computing power, I used 3 epochs for each training session on 4 different models, performing a grid search on three different hyperparameters: a fixed learning rate of 0.0001, batch sizes of 8, 16, and 32, and λ values of 0.25, 0.5, and 0.75. The λ value indicates the relative weight of the two different losses contributing to the total loss. The total contrastive loss is calculated using the cross-entropy loss of a given image (e.g., a street view) and a given prompt (e.g., "A street view of Washington D.C."), as described in the original CLIP paper.[2]

I introduced an additional loss function, which is the classification error between the predicted geohash label and the actual label. During training, the model predicts the geohash value of a given image-text pair using a classification head, which replaced the final regression projection head. Therefore, the total loss function not only utilizes the contrastive learning of the image-text pair data but also incorporates the loss of the predicted geohash label versus the actual geohash label (converted from numerical coordinates into a geohash value during dataset preparation):

$$loss_{total} = loss_{contrastive} + \lambda loss_{geohash}$$

, where λ is a value between 0 and 1, which adjusts the weight of two different losses.

6 Result

The model was tested using four different backbone architectures (RN50, RN50x4, ViT-B/16, and ViT-B/32) with a fixed learning rate of 0.0001. All model runs were fixed to epoch of 3. Resource constraints, using only 16GB of RAM and a single RTX 4060 GPU, limited the batch size and prevented the use of larger backbone models(RN101, RN50x16, 'ViT-L/14', 'ViT-L/14@336px') - the training epochs stopped due to lack of memory inbetween. Most experiments were therefore conducted with batch sizes of 8 or 16. A batch size of 32 was successfully achieved only in a few runs using the ViT-B/16 backbone, likely due to its smaller computational requirements compared to the other tested models.

The baseline model was the use of CLIP using RN50, with learning rate of 0.0001, and batch size of 8, and $\lambda = 0$. The baseline model used classification head in line with other models for consistency of model output, but it disable the use of classification error(of geohash labels).

The results, while somewhat below expectations, provided valuable insights to take a looked at. With a geohash size of 2 (approximately 600 km expected error), the training data encompassed 109 geohash classes, rather than thousands of possible geohash grid cells because we excluded geohash grids of the ocean and many rural areas, as random image retrieval from the Google API often produced landscapes (e.g., forests, fields) lacking distinctive features useful for training. The choice of geohash size balanced computational feasibility with the need for spatial specificity, though finer granularity might improve model performance in future experiments by capturing more localized patterns. However, increasing the geohash granularity to 3 or 4, which has much less expected error, in a sample run, resulted in larger classification errors due to too many classes being used. The idea of this project was to incorporate the generality and educated guesses of the human mindset—that instead of being exquisitely correct, we intended to make a generalized guess but with some degree of error.

[1] https://en.wikipedia.org/wiki/Geohash

[2] https://github.com/openai/CLIP

Validation accuracy of 0.05 may be somewhat disappointing, but let's take a look in terms of human perspective. The total number of capital cities included in the dataset were 190 countries. Therefore, a blindfolded guess would yield an accuracy of less than 0.5 percent, which would be 0.005. Even if the human were given information of 109 geohash grid locations, uneducated guess would yield accuracy of 0.01 without any prior knowledge. The best model(RN50, batch size = 16, $\lambda = 0.25$ achieved an validation accuracy of 0.1082 , outperforming all other tested models included the baseline model. This clearly illustrate the power of contrastive learning, using more images in a match creates more semantic space for images and text pairs to compare to. Also, it is worth looking that putting more emphasis on classification and less emphasis on constrative loss yielded best result.

It is interesting to see that most of the results yielded a validation accuracy of 0.05—regardless of the backbone model type or the lambda values used to adjust the loss function. The only hyperparameter that made a meaningful difference in these models was the batch size, as it determines the number of images used in each batch. The core of contrastive learning lies in finding the best image-text pair by comparing it with other pairs within the batch. Therefore, when learning the semantic spaces of images given a prompt, it is not the complexity of the backbone model or the number of layers that matters most, but rather the model's ability to leverage the data in the batch to make meaningful decisions about the optimal pairs. This indicates the quality of images, and quality of 'how' images in a batch is chosen, understanding the semantic spaces of image-text(prompt) pairs in respect to the other pairs, and the optimal batch size for an objective could be a key to future work to extend work on this project.

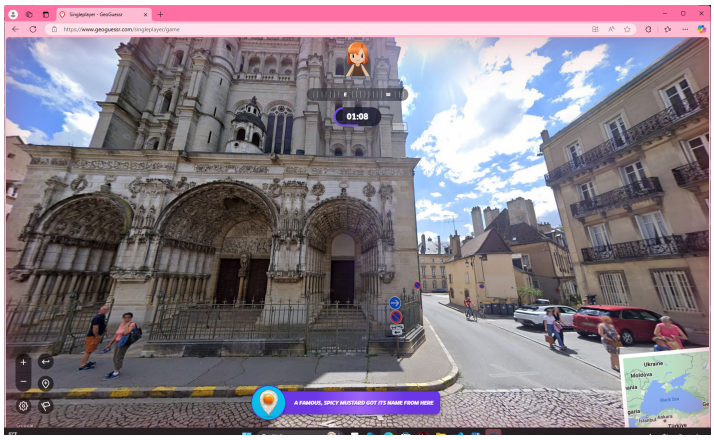

Figure 1: GeoGuessr game format - you can click on your guess by enlarging a map in the bottom-right corner.

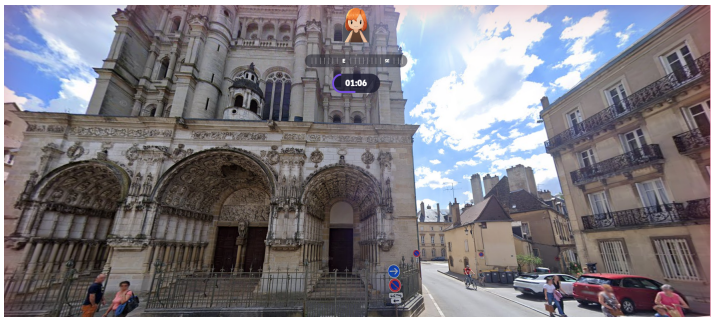

Figure 2: Taking a screenshot that excludes unnecessary components of the images.

7 Applying to the game

The screenshot is then saved into a local directory, where a Python file continuously runs, checks for new images in the folder, picks them up, and uses the trained model to make inferences. The output of the model, which is a geohash value, is dehashed into numerical coordinates and plotted on a Cartopy library world map so I can use it to make a guess in GeoGuessr. Figures 1 and 2 explain how the image is generated, Figure 3 illustrates how data is processed to reference the map, and Figure 4 explains how the score is calculated, primarily based on distance metrics.

I have attempted to use the model's reference to the game approximately 20–30 times. Most of the attempts were completely incorrect, while a few were somewhat close but exceeded our approximate margin of error, which corresponded to an error range equivalent to the length of 2 geohashes. The significant errors during the inference stage could be attributed to a lack of dataset volume, poor data quality, or even differences in the initial image format (e.g., timers and game-related elements included in the photo, as seen in the screenshot).

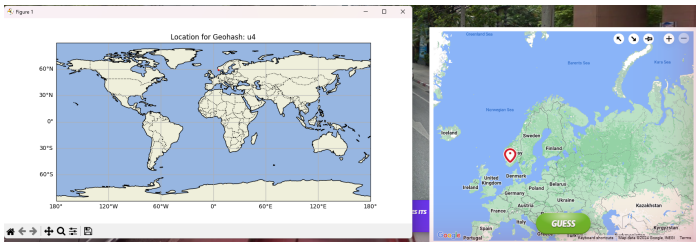

Figure 3: Using inferred coordinates to make a guess.

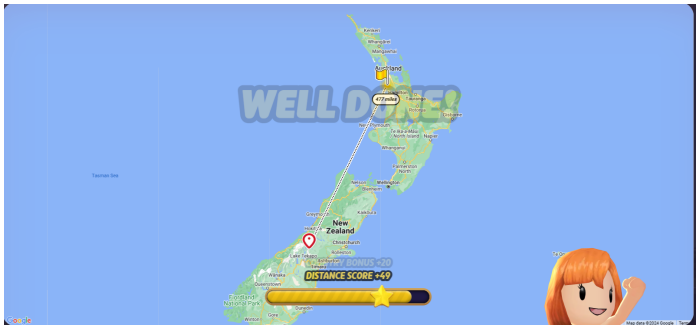

Figure 4: Scoring metric based on distance and time taken.

8 Discussion and Conclusions

Without the help of the model I built, my educated guesses turned out to be much more accurate. Small details that the model may not have learned—such as the type of license plates, a wrinkled flag hanging on a wall, a store sign, or the skin color of a bystander—are challenging for the model to capture in the semantic space due to the many limitations we discussed earlier. While I did not intend for this model to perform exceptionally well, it was a worthwhile project that allowed me to review the materials and papers from class and apply them in code.

Learning the world's landscape through a set of images is a daunting task—not only in terms of computation but also in terms of dataset requirements. It is a tremendous challenge to learn the semantic spaces of every intricate detail within the world's diverse landscapes. Although I was aware that my dataset would be insufficient for capturing the semantics of global landscapes, incorporating contrastive learning with a prompt to build a model structure was an exciting and rewarding process. As mentioned earlier, the dataset, which consisted solely of capital cities, was far from comprehensive. Many parts of the world were excluded during training, which may have been a design flaw, as these regions were not represented in the geohash classes.

There are several improvements I wish I could have implemented, such as using a larger dataset with greater granularity and classification to incorporate more geohash classes. Although I collected approximately 10,000 images, my initial expectation was closer to 100,000. Alternatively, sourcing higher-quality images, experimenting with different loss functions, or incorporating multiple images at the inference stage could enable the model to process more visual information before making a prediction. Larger batch sizes would also offer greater opportunities, as the contrastive algorithm benefits from comparing logits across more diverse images within the batch. Additionally, improved computational resources could have significantly enhanced both training efficiency and overall model performance.

This project showed the importance of thoughtful dataset design and the challenges of balancing computational limitations with ambitious goals. While the model's performance was not optimal, the process was enjoyable and applying the theory into code to build a model was thought provoking process. It also helped me rethink each stages of data processing and model construction affects the result. Overall, this project served as an invaluable opportunity to bridge theoretical concepts with practical implementation.

9 Statement of Individual Contribution

This project was built alone.

10 Statement of Individual Contribution

A Result Table

Run	Model	LR	BS	λ	Train Loss	Train Acc	Val Loss	Val Acc
Baseline	RN50	0.0001	8	0	1.9962	0.0081	1.9981	0.0105
1	RN50	0.0001	8	0.25	3.0995	0.0504	3.0827	0.0504
2	RN50	0.0001	8	0.50	4.0831	0.0545	4.0297	0.0641
3	RN50	0.0001	8	0.75	5.0873	0.0578	5.0623	0.0588
4	RN50	0.0001	8	1.00	6.1235	0.0565	6.0613	0.0536
5	RN50	0.0001	8	2.00	10.0605	0.0558	9.9511	0.0504
6	RN50	0.0001	16	0.25	3.3450	0.1085	3.3587	0.1082
7	RN50	0.0001	16	0.50	4.3108	0.1044	4.3236	0.0924
8	RN50	0.0001	16	0.75	5.3658	0.0967	5.4531	0.0756
9	RN50x4	0.0001	8	0.25	3.0600	0.0532	3.0683	0.0462
10	RN50x4	0.0001	8	0.50	3.9972	0.0581	3.9640	0.0515
11	RN50x4	0.0001	8	0.75	5.1254	0.0529	5.0852	0.0473
12	ViT-B/16	0.0001	8	0.25	3.1044	0.0484	3.0931	0.0536
13	ViT-B/16	0.0001	8	0.50	4.1034	0.0511	4.0895	0.0483
14	ViT-B/16	0.0001	8	0.75	5.1523	0.0467	5.1099	0.0536
15	ViT-B/16	0.0001	16	0.25	3.7587	0.0482	3.7245	0.0515
16	ViT-B/16	0.0001	16	0.50	4.8016	0.0506	4.7478	0.0525
17	ViT-B/16	0.0001	32	0.25	4.4215	0.0557	4.4039	0.0515
18	ViT-B/16	0.0001	32	0.50	5.5323	0.0509	5.4885	0.0536
19	ViT-B/16	0.0001	32	0.75	6.5789	0.0494	6.5306	0.0536
20	ViT-B/32	0.0001	8	0.25	3.1041	0.0493	3.0924	0.0536
21	ViT-B/32	0.0001	8	0.50	4.1004	0.0469	4.0374	0.0515
22	ViT-B/32	0.0001	8	0.75	5.0858	0.0544	5.0401	0.0588
23	ViT-B/32	0.0001	16	0.25	3.8048	0.0499	3.7821	0.0525
24	ViT-B/32	0.0001	16	0.50	4.8374	0.0495	4.8018	0.0536
25	ViT-B/32	0.0001	16	0.75	5.8310	0.0469	5.7400	0.0483

References

[1] Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, et al. "Learning Transferable Visual Models From Natural Language Supervision.", 2021.

[2] Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A Simple Framework for Contrastive Learning of Visual Representations." *arXiv preprint arXiv:2002.05709*, 2020.

[3] Tobias Weyand, Ilya Kostrikov and James Philbin. "PlaNet-Photo Geolocation with Convolutional Neural Networks", 2016.

[4] Nam Vo, Nathan Jacobs, James Hays "Revisiting IM2GPS in the Deep Learning Era". 2017

[5] Kaiyang Zhou Jingkang Yang Chen Change Loy Ziwei Liu "Learning to Prompt for Vision-Language Models", 2020.

[6] R.H. van't Veer, P. Bloem, E.J.A. Folmer "Deep Learning for Classication Tasks on Geospatial Vector Polygons", 2019.