Comparison of Classification Regions for AI Geolocation

Ishan Patnaik

Abstract

Geolocation is the foundation of almost all technology that deals with spatial data, from the GPS-based navigation software in a vehicle to the built-in tracking system on a mobile device. At its core, geolocation is simply the process of determining or approximating the geographical position of an object. However, the type of geolocation we are interested in is slightly more specific; it is the process of locating a single image solely based on its internal information. Until recently, it has been extremely difficult—if not impossible—to do this reliably. These techniques are often applied by intelligence agencies, in which specialized analysts use geolocation to track down wanted individuals by investigating details in the images they appear in. Even then, it is by no means straightforward to check these images against endless amounts of map data and satellite imagery, let alone pinpoint an exact location. However, with the increasingly powerful capabilities of artificial intelligence and machine learning, we are starting to see a shift in the applications and techniques used in geolocation. In this article, we aim to determine the most effective way to teach a computer how to match images to their respective locations on a map.

Introduction

Before we explore the specific implementations of machine learning in geolocation, let's take a step back and understand the significance and practicality of doing so. We can start by considering tasks for which we already have systems in place, but could be automated or at least facilitated with more efficient geolocation. The iOS Photos application, for instance, has a feature that allows users to tag their photos with locations. This makes it possible to search for and retrieve images by place, automatically create photo albums for vacations, and even to view the camera roll on an interactive map. If the initial process of labeling images could be done by a computer—without forcing the user to allow constant access to their location or enter it manually—it would likely be a lot more feasible and accessible. Recall the role of geolocation in intelligence agencies for tracking down people or objects in images. This could not only be made significantly easier with machine learning, but also be applied to other similar situations. In particular, it could be used by emergency responders to pinpoint the geographical origin of a crisis or natural disaster based on photos, or by law enforcement to analyze evidence and locate suspects or crime scenes. The same technology could even be used for wildlife conservation, and could locate endangered animals or identify their migration patterns using anonymous or alleged sightings of them.

In recent years, the use of machine learning for geolocation has grown considerably more common, which can largely be attributed to the rising popularity of a game called GeoGuessr. In a typical round of GeoGuessr, the player is given an image from Google Maps and is challenged to place its coordinates on a map as closely as possible. In order to do this accurately, players have learned to analyze terrain, architecture, languages, road signs, and countless other details that provide insight into the image's location. Naturally, people began to wonder how well a computer could play this game, and machine learning seemed the most appropriate way to answer this question. This was the motivation behind numerous "GeoGuessr bots" which subsequently evolved, some reaching surprising levels of proficiency. It was not until recently, however, that one such bot became strong enough to outplay even the most skilled human players. More specifically, the PIGEON—Predicting Image Geolocations—AI (Haas 2023) is able to guess roughly 92 percent of countries correctly, with an average distance of just 27 miles from the correct location.

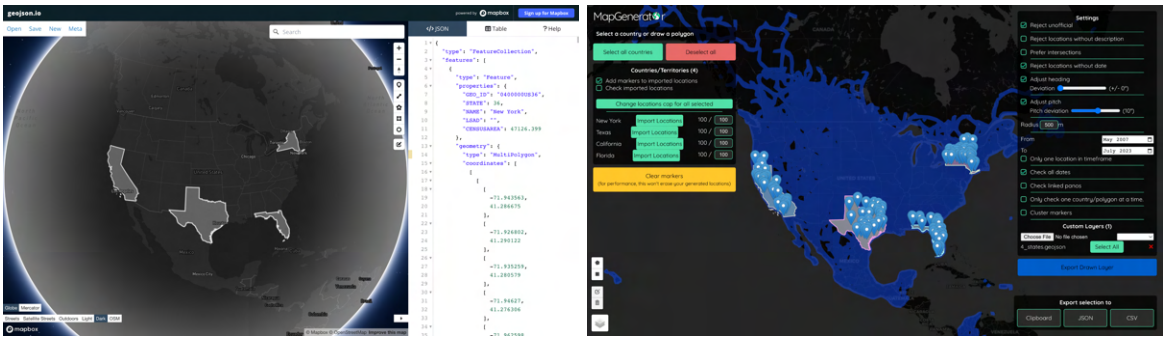
PIGEON was built over OpenAI's CLIP model and trained on data directly from the game of GeoGuessr, as opposed to Google Maps. It performed image classification—sorting images into categories by recognizing patterns between groups of pixels—on specially selected areas of land called "geocells." These cells were defined based on a number of unique characteristics, including "road markings, infrastructure quality, and street signs" (Claburn 2023). However, as effective as these cells may be, they are extremely difficult to generate, and will vary based on the scope and intent of the model. This raises an important question: What is the most effective way to classify the world for a machine learning model with a basic, well-defined set of rules? We propose three distinct methods that could potentially answer this question: region-based, grid-based, and cluster-based classification. Region-based classification separates the world into its national or administrative subdivisions (e.g. countries, states or provinces), grid-based classification separates it into square cells based on latitude and longitude, and cluster-based classification defines arbitrary regions based on where the images are concentrated.

In the context of GeoGuessr, we tend to think of the world in terms of cities, states, and countries. For instance, a player might identify the *country* of Japan by noticing its unique script, or distinguish between *states* of the USA by memorizing the flags or license plates associated with each one. However, there are times when this strategy may seem less reasonable. In other words, even if the player correctly determines that a location is in Russia, their guess could still be thousands of miles away from the true coordinates. In this case, it would make more sense to organize the world not into countries, but rather into regions with little variation in size or number of image samples. That is, if the general landscape remains relatively similar throughout each region, it may be easier to recognize. Our goal is to put these ideas to the test and figure out whether a computer guesses more accurately with one of these strategies than another, and what factors might be causing that discrepancy if so.

Methods

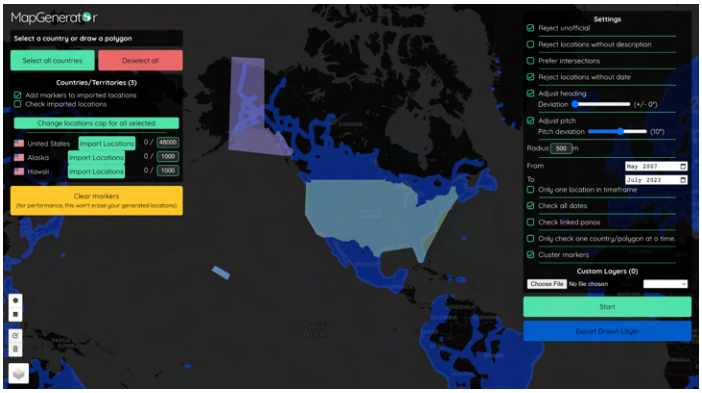
(GitHub Repository)

The basic structure of our analysis is to create custom image datasets for each of the three proposed methods of classification—grid-based, region-based, and cluster-based data labels—and use a standard neural network to compare their respective performances. To make sure we properly adapt our model to the new geographic data, we first test it out on a simpler situation. Our test case is a classification among the four most populated U.S. states: California, Texas, Florida, and New York.



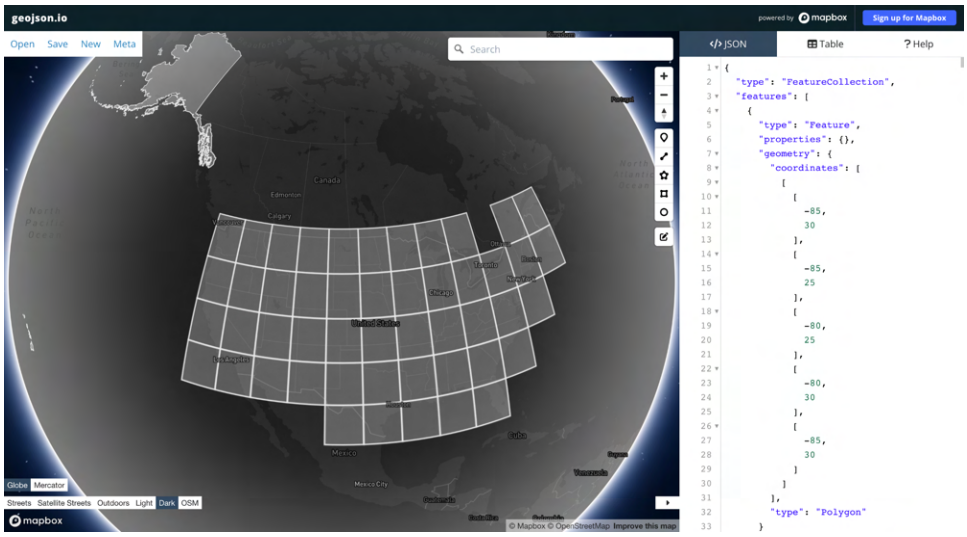
Randomized image collection with MapGenerator and GeoJSON.io

To obtain the training images, we construct a GeoJSON—a file that stores both spatial and non-spatial information about a set of geographic shapes—using tools from GeoJSON.io, GeoJSON Grid Creator, and MapGenerator for custom map creation and data visualization. This allows us to define the relevant regions on a map and generate a specified number of images in each one. We used Google's Street View Static API to pull 100 images from each state at size 100x100 pixels. Once the images are appropriately formatted, we apply them to PyTorch's IMAGENET1K_V1 model with pre-trained weights, achieving an accuracy of roughly 60% after 100 epochs.

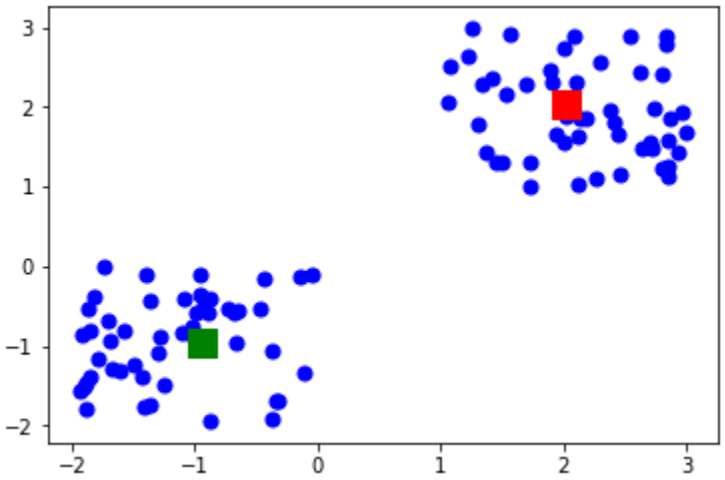


MapGenerator custom location distribution

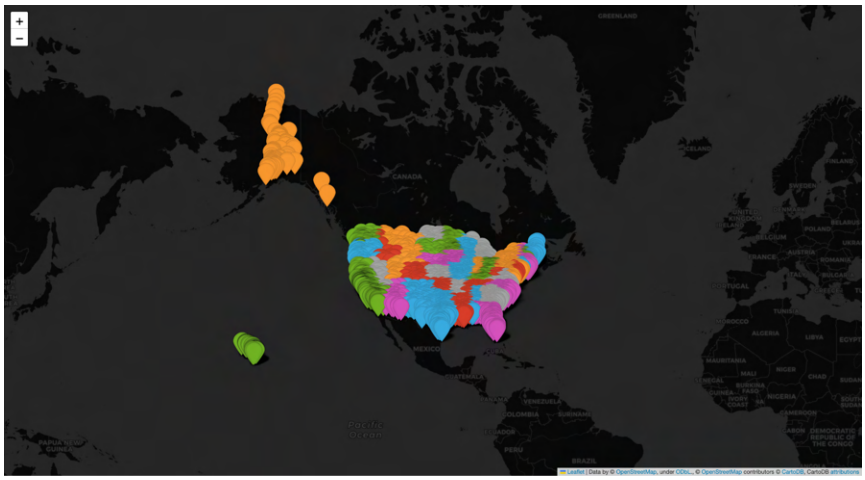
We follow a similar process to generate images for the entirety of the United States, this time with 50,000 samples at size 256x640 pixels each. The state-based labels are generated using a reverse geolocation API and government data, the grid-based labels were calculated based on a formula that assigns each pair of coordinates to a cell of width 5° longitude and height 5° latitude, and the cluster-based labels are calculated using the K-means clustering algorithm.



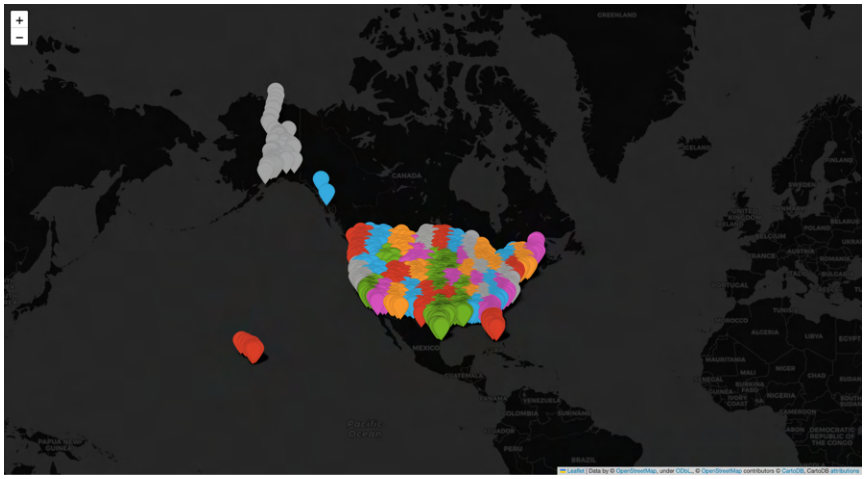
GeoJSON.io custom grid map



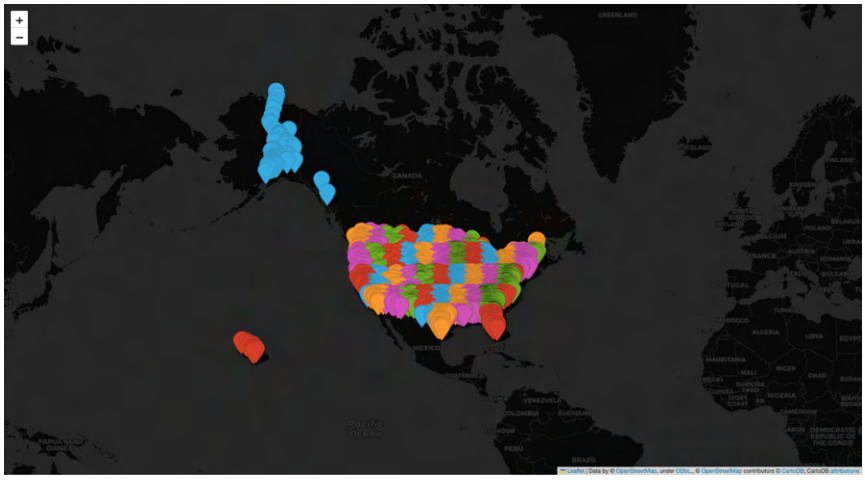
K-means clustering example with random points (source: Towards Data Science)



State-Based Classification (visualized with Folium)



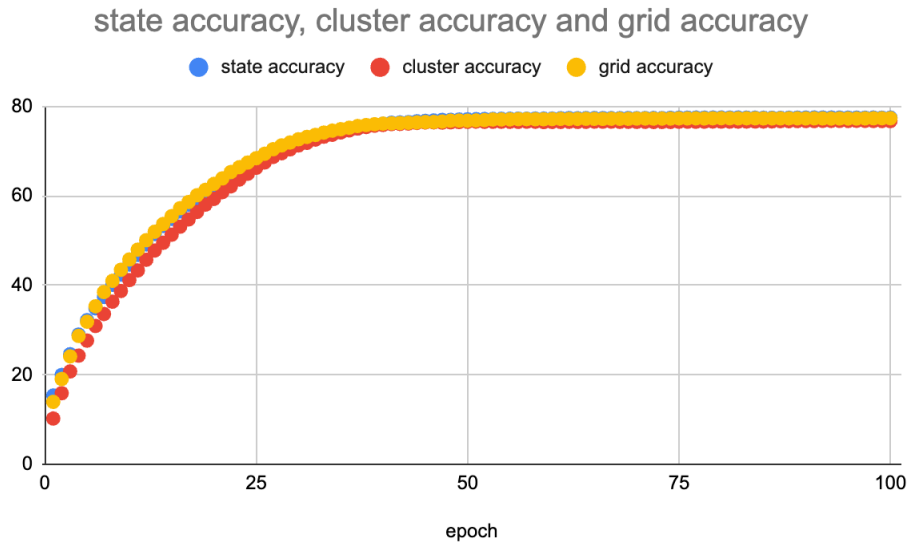
Cluster-Based Classification

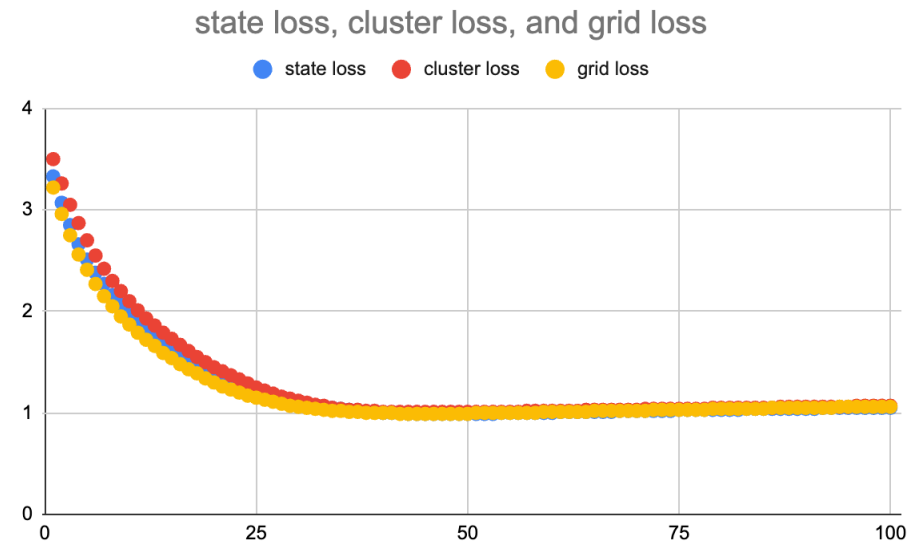


Grid-Based Classification

Results

After training our model on each of the three datasets for 100 epochs, they yield performance data that at first glance seems remarkably uniform. The state-based model performs at 77.49% accuracy, the cluster-based model performs at 76.72% accuracy, and the grid-based model performs at 77.35% accuracy. In addition, they each achieve a test loss of roughly 1.05. As shown below, the rate of learning for each model starts out relatively high and gradually decreases to zero, hence the logarithmic/exponential curves for these metrics.

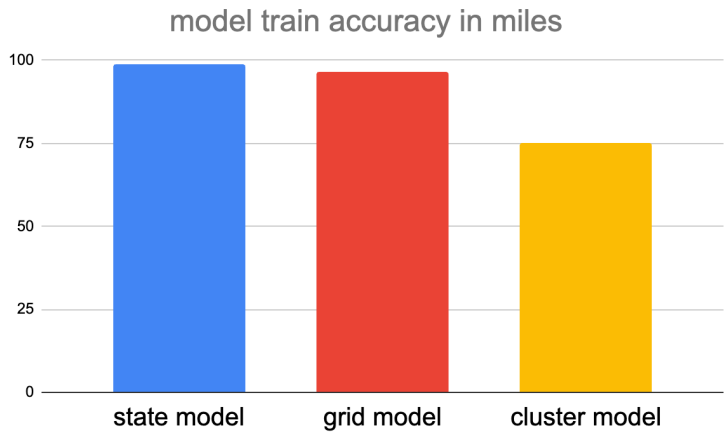




However, “accuracy” simply measures how often the model guesses in the correct geographic subdivision (state, cluster, or grid cell). It does not take into account how close the model’s incorrect predictions are, as well as the margin of error in its correct predictions. In other words, if the model misidentified North Dakota as South Dakota, it would technically be just as “inaccurate” as guessing Florida, even though we know intuitively that South Dakota is a much better guess. On the other hand, if the model correctly predicts Texas, it would be considered just as “accurate” as correctly predicting Vermont. Even though both are correct, we gain much far more information from knowing an image is in Vermont than in Texas, as the former narrows the possibilities down to a much smaller region. How, then, can we fairly account for these complications?

Our solution is rather simple: to create a new metric for accuracy that, instead of judging a prediction as “correct” or “incorrect,” outputs a number based on the distance between the predicted location and the true location on a map. This is precisely the same system used by GeoGuessr to assign a score to a player’s guess, which makes sense given the inherent connection between our process and the game of GeoGuessr. In our case, the model’s “guess” is the geographic center of the corresponding state, cluster, grid-cell. We then compare this to the true latitude and longitude of the image to calculate an “error”, which is its geographical distance from the true coordinates. Now that our model predicts numerical values instead of discrete categories, we have essentially turned a classification task into a regression task.

With this new measurement, we put each model through its testing dataset once more to achieve results that better fit our goals. We find that, on average, the state model is roughly 52.34 miles away from the true location, the grid model is 52.11 miles away, and the cluster model is 48.65 miles away. This means the cluster model performs 7.05% better than the state model, and 6.65% better than the grid model. We also test them against the training data and find even larger gaps between the models. In this case, the cluster model performs 23.94% better than the state model and 22.10% better than the grid model.



Discussion

To briefly summarize these numbers and accuracy scores, the crux of our findings is that the cluster-based labels yield a significantly more accurate model than the other two methods of classification do. This model is also less affected than the others when we switch the dataset from training to testing, meaning it is less prone to underfitting and is able to generalize patterns more efficiently than the state-based and grid-based models.

Thus, we have answered our question: It turns out that, under the conditions we simulated, cluster-based data labels are the best method of classification for AI geolocation. In light of this, let's try to understand why exactly the cluster model performs comparatively well. Unlike the other two methods, K-means clusters are based on the distribution of Google Maps images rather than just geographical areas. Since the K-means algorithm defines these clusters around geometric centroids, areas with more images will have more clusters. Compare this to grid-based classification, where areas with more *land* will have more squares, or to state-based classification, where boundaries are defined by culture and politics. Since images are distributed equally across clusters, the samples in a given cluster are more likely to resemble one another, and therefore yield more reliable patterns for the model. That is, while two images in California could look completely different (desert, forest, coastline, etc.), two images in a given cluster are less likely to vary so drastically.

Additionally, since any two clusters have the same number of images, they convey roughly equal "information" (i.e. they narrow down the range of possibilities equally). In contrast, a state or grid cell with 1000 images conveys much less information than one with only 100 images. Here's an example of this using simple probability theory: Imagine we only had two different regions—one with 2 possible locations and one with 8 possible locations. Since images are selected with equal probability, we get the first region 20% of the time and the second 80% of the time. If we always knew the correct region, we would narrow our options down to 6.8 locations on average. Now imagine we split the images evenly so that each region has 5 images and is selected with 50% probability. This time, if we always knew the correct region, we would narrow our options down to 5 locations, which is over 25% better. Keep in mind our assumption of always knowing the correct region; this argument would not hold if the first pair of regions were easier to differentiate than the second pair was. However, since our tests have revealed that each of the three models—state-based, grid-based, and cluster-based—identify the correct region with the same accuracy (roughly 77%), this probabilistic example is consistent with our results.

Conclusion

With all this in mind, we can safely say that in our study—which deliberately excluded the time- and labor-intensive process of manually selecting geocells—K-means clusters were the most effective way to classify the United States for a machine learning model. It's worth noting that since this outcome is based on image data from the United States, one might find that states or grid cells work better in a different country. For example, if there existed a country in which each state mandated a particular house color for its residents (e.g. State 1 has red houses, State 2 has blue houses, etc.), state-based classification would likely perform best. Or, if a country's Google Maps images were equally spread out across its land, maybe it would be more computationally efficient to use grid-based classification. One could also compare different clustering algorithms or shapes for grid cells, or even invent an entirely new method of classification from the three we have discussed. For example, it may be interesting to define regions based on what the model *predicts* for each state, so that "Alaska" is simply all the locations that the bot initially labels "Alaska," regardless of whether it is correct or incorrect. This could group together images that have more meaningful patterns, and potentially help inform what patterns we look for when defining geocells manually. If we continue to answer these questions and gradually fine-tune our algorithms for AI geolocation, it will not only enhance its existing applications, but also open the door for it to be used more commonly and effectively in other fields.

References

- Patnaik, Ishan GitHub repository for this project *github.com*
- Alwer, Saleh GeoGuessr-Inspired Exploration of CNNs *medium.com*
- Alwer, Saleh GitHub repository for above article *github.com*
- Haas, Lukas PIGEON: Predicting Image Geolocation *Stanford University*
- Claburn, Thomas PIGEON AI review article *theregister.com*
- Rainbolt, Trevor PIGEON AI vs. Professional GeoGuessr Player *youtube.com*
- Berton, Gabriele Deep Visual Geo-localization Benchmark *Politecnico di Torino*
- Ernst, Douglas Geolocation in Intelligence Agencies *washingtontimes.com*
- OpenAI OpenAI Clip Model *openai.com*
- PyTorch IMAGENET1K_V1 pre-trained model *pytorch.org*
- Education Ecosystem K-means clustering algorithm *Towards Data Science*
- Metin, Ferhat Map Visualization with Folium *medium.com*
- ScienceBase USA Spatial Data by State *sciencebase.gov*
- Google Maps Street View Static API *developers.google.com*
- GeoJSON map editor: geojson.io
- GeoJSON grid creator: cityofaustin.github.io/geojson-grid
- GeoGuessr map generator: map-generator.vercel.app
- GeoGuessr: geoguessr.com