Project Document: Geo-location Neural Network
Rectified Linear Undergrads

Dakota Andrews
Gage Cammack
Joseph Seibel
Tong Wu

March 3rd, 2024

Contents

Abstract

GeoGuessr is an online game where the player is given full navigational capabilities of a random location on Google Maps, with the goal of approximating the true location of the given environment. Players learn distinguishing features from country to country, including but not limited to license plate color and size, certain types of telephone poles, and traffic signs. Our goal is to train a deep learning model to be able to recognize these distinguishing features, and then closely approximate the true location of a given game state. If this model proves successful, it would dominate competitive GeoGuessr matches, could be used to identify new distinguishable features not yet found by players, and could even be used by law enforcement if they ever needed to identify a location based solely on a single image.

Our two approaches for this project both involve a Convolutional Neural Network. One of which utilizes geocells to classify the location, and the other is a regression with a transfer learning model that finds the approximate longitude and latitude. Since this project deals heavily with image data, we expect the geocell Convolutional Neural Network to perform better.

In our first attempt, we created a CNN model and tested it on 100,000 images, split using the geocell method. Our results ended subpar, with a Top 1 accuracy of only 15%. We were able to further refine the model with more images and deeper networks to achieve a Top-1 accuracy of 33% and a Mean Distance of 419 miles. After accounting for ecological zones in the creation of geocells we were able to achieve a Top-1 accuracy of over 42% and had a mean distance of 132 miles from the location of the photo. This performed substantially better than the average human and most benchmarks.

Chapter 1

Milestone 1: Project Ideas

1.1 Introduction

When the Rectified Linear Undergrads began thinking of ideas, we were quite stumped on what type of project we wanted. Initially, we had grand ideas, like training an A.I. to beat Bloons Tower Defense 6 on the hardest difficulty [3]. However, we soon realized that may be beyond our scope and the requirements were too constraining. Instead, we settled on ideas more personal to our members; GeoGuessr Bot and Art Generation in Style Bot. The goal of the Geo-guesser bot is to locate the geographic location through a single picture, which has practical and entertaining uses. This idea was posed by Joseph Seibel, who was the only person in the group with a prior Machine Learning background. As for Art Generation in Style Bot, our goal is to generate some drawings via prompts by an artist's unique style.

1.2 Project Idea 1: GeoGuessr Bot

The problem we are solving with this model is correctly predicting location based on an image. Some applications that this would have is winning games in Geoguessr and helping to locate pictures of missing/wanted people based on photos of their locations.

The main approach we are considering using to solve this problem is the classification of images into geocells similar to the PIGEON network that is used to geo-locate images [7]. We would use a convolutional neural network with an input of a panorama image 1.1 and an output of the predicted geocell in which the image was taken. These panoramas would be sourced from the Google Street View API which guarantees a large source of high-quality data for training and validation that is pre-labeled and easily accessible. The resolution of the image returned from the API is 600×600 but training would be done on images of smaller size, depending on the complexity of the model architecture, to keep the computational resources needed to a manageable level. To sample

the dataset in a useful manner we are going to use the population of each geocell to determine the number of images to include, hopefully leading to more accurate predictions in areas where it would have more utility.

Figure 1.1: Google Street View Panorama

We also believe that using ensemble learning techniques such as bagging and boosting will help to better refine our predictions and produce more accurate guesses. Although ensemble learning techniques do require much more computation they can help to increase the accuracy of predictions and reduce variance which we believe will be important for the usefulness of our model [10]

We plan on using Living Atlas to get our geocell data and using the ArcGIS software to plot the distribution of our data and merge cells so that all cells have no more than 25% more points than the median cell.

By separating the classes in this way we believe that it will help our model increase accuracy in areas where more images are taken without greatly increasing the complexity of the model.

1.3 Project Idea 2: Art Generation in Style Bot

Often artists struggle to meet deadlines and drawing a single piece may take days. However, what if an artist could upload their pieces and provide data of their style to generate a desired drawing? That's exactly what this project aims to do: for artists to prompt an idea and generate art in their own artistic style.

This tool requires initial art generation training so it can recognize objects like a coffee mug or a person before any output is given. The second step is to allow users to upload large quantities of their work so the bot can recognize patterns in the artist's style and copy it. Finally, we'd need to train it to take in an input prompt and generate the correct output in said style [11].

The input data for this model would be a collection of works from each artist whose style the model would be expected to emulate and a prompt of what the

user would want the image to look like. The output of the model would be an image that follows the specified prompt for the respective artist.

The process of input for each artist would require at least 20 of their pieces, with the choice of (preferably) uploading more to better understand their art style. For our initial model, we will train it using the same method as Open AI's DALL-E [1] so it can generate images via a prompt. To train individuals, we'll use art from famous artists that can be accessible online and frame the A.I. to understand the artist's niche and patterns within their work. Our goal for each user is that they create their own smaller model using their work, which paired with our initial model, should help recognize and create their desired vision.

Project Idea 2 requires the use of classification, clustering, and generation. To be more specific, we'd be using a Generative Adversarial Network (GAN) as our model [2].

1.4 Conclusions

Although both ideas are interesting learning problems and would provide valuable learning experiences. We believe that idea 1 is much better suited to the time frame and scope of this semester. Problem 2 has some issues that would need to be addressed such as a lack of quality datasets, an extremely complex learning problem, and a lack of resources for training such a complex model. Problem 1 also has some downfalls such as the need to learn how to use GIS software to create and merge geocells, but it is much more feasible within the time frame of this course. Another limitation is that the Google API limits users to 30000 images per month thus will all 4 of our members we will only be able to get 120,000 images to start with until March 1st. Although that is not a small amount, having more data is always better than having less and may cause a few issues if some locations end up with much more sparse data. Additionally, since half of our group members have no prior experience in machine/deep learning, we feel it would be more practical to tackle the simpler of the two problems before trying to do a much more complex problem.

Table 1.1: Contributions by team member for Milestone 1.

Team Member	**Contribution**
Tong Wu	Write up for Idea 2 Introduction
Joseph Seibel	Write up for Idea 1
Dakota Andrews	Find and Create Citations, Edit for Idea 1
Gage Cammack	Write up for Idea 2, Edit for idea 1

Chapter 2

Milestone 2: Project Selection

2.1 Introduction

Due to time constraints and our limited workforce, we have chosen the GeoGuessr Bot as our project. Likewise, we decided to narrow our scope, and we will specifically focus on locating images within the United States. This means training our data on US images and testing it on US images. While we would like to train on images around the world, it is unrealistic due to our time and resource constraints. For the task of US image geolocation, we came up with two solutions utilizing a Convolutional Neural Network: a Geocell Classification Model, and a Regression With Transfer Learning . The geocell model's objective is to classify the image to a specific cell on a map, whereas our regression model is expected to return the longitude and latitude of the image. We'll be training our models using Google Street View panorama images [6] of a location. By careful choice of architecture and the novel use of geocells, we hope to achieve lower error than other attempts at solving this problem. If this model proves highly successful, it may even provide insight and new tactics for professional GeoGuessr players, or be used to locate images with limited metadata.

2.2 Problem Specification

2.2.1 Project Statement

We are trying to predict the location of an image on Geoguessr to maximize our points earned for each image in the United States. We will use Google's Street View panorama images [6] to train our model (our inputs), and our models will return the estimated geographic location (our outputs). Since we are using all labeled data, this is a supervised learning problem.

2.2.2 Motivation

In our approach, we're integrating machine learning techniques with the popular Geoguessr game to improve players' ability to accurately guess locations in the United States. What sets our method apart is its reliance on Google's Street View panorama images for training, enabling us to fine-tune our model for better performance. We believe this approach will be successful because it leverages a vast dataset of real-world images, allowing our model to learn from diverse scenarios.

Our work matters because it enhances the gaming experience for Geoguessr players, providing them with a valuable tool to increase their scores and enjoyment. By successfully predicting locations, players can feel a greater sense of accomplishment and engagement with the game. Moreover, our approach showcases the practical applications of machine learning in recreational activities, demonstrating its versatility and potential beyond traditional domains. Ultimately, our success could inspire further exploration of machine learning in gaming and foster innovation in the intersection of technology and entertainment.

2.2.3 Required Resources

To train our models, we'll need large amounts of latitude-and-longitude-labeled images. Luckily, the Google Street View API [6] can provide us with panoramas with their corresponding latitude and longitude. The API requires setting up a Google Cloud account, and each account can download 28,500 free images each month. Any more images after that is $0.007 per image. To avoid costs, we can create four separate accounts (one per group member) and pull a total of 114,000 panoramas, which satisfies the scale for our project. We also used ArcGIS Pro [4] this software allowed us to create hexagonal Geo cells over the Continental United States totaling 5,662 Geo cells. With it, we can map all the panoramas we got to their corresponding Geo cell based on latitude and longitude data.

2.2.4 References

We'll be looking at three sources and see how other people accomplished similar projects with varying degrees of success. The first one is from Stelath [12], the second one is from nirvan66 [13], and the third and most successful is from a group of Stanford students who created the PIGEON bot [7].

2.3 Related Work

During our research, we discovered three sources and approaches in solving our Geoguessr dilemma with varying degrees of success.

The first approach is by user Stelath, and his project is named geoguessr-ai [12]. His goal is similar to ours, and he explains how he wanted to "reliably

guess a random location in one of five US cities (unnamed) given only a Google Street View image. The idea was inspired by the game GeoGuessr." To accomplish this, he uses a PyTorch implementation of a 3D CNN & RNN model. Similar to us, he chooses to narrow his scope to the United States and uses Google's API [6] as his dataset, but he only pulls 50,000 images for his training set. He formatted his targets by turning the GPS coordinates into multi-class targets in one hot array with numbers ranging from one through ten. Then he tried a wideresnet with 50 layers and 20 epochs, which yielded the best performance in his own words. Stelath believes he can improve the model by adding more layers and training a 3D CNN on an array of images from a location, giving the model more data so it may make accurate predictions.

The second approach is by nirvan66, and his article title is GeoGuessr AI: Image-Based Geo-Location [13]. Similar to the first approach, this project focuses on the United States but has a broader scope that covers the entirety rather than 5 cities. To begin, nirvan66 split the US map into grids of 4 squared units totaling 138.265 squared units with border grids being smaller. This resulted in 243 grids forming the mainland US. For his dataset, he uses three images per location for 40 locations per grid, pulling from the Google Street View static API [6]. Each 3 images represent headings of 0°, 90° and 180°. This resulted in 9720 locations (far more than 5) and 29160 images. As for the models, nirvan66 built and trained two custom CNN/LSTM models using Keras. The images were converted to numpy arrays made up of RGB values and had the shape of (3, 300, 600, 3). Using a soft-maxed output for prediction, the grid numbers corresponding to the given input image vector were converted to one hot vector. For better experimentation and comparison, he decided to train two models; the first had a ResNet CNN whose weights were frozen during training connected to a trainable LSTM, and the second had a trainable CNN connected to an LSTM. Figure 2.1 shows the final accuracy representing the average distance in miles the model was wrong by.

The third approach is called PIGEON: Predicting Image Geolocations, created by a group of Stanford students [7]. They created and trained two models: PIGEON which is made to compete in GeoGuessr, and PIGEOTTO which is more generalized for unseen places. For this project, we will focus on the first model, PIGEON, which boasts to have consistently placed over 40% of its guesses within 25 kilometers of a target location globally. It also ranked among the top 0.01% of competitive GeoGuessr players. To accomplish such a feat, PIGEON utilizes a CNN with a Geocell Classification model. Their dataset consisted of 100,000 randomly sampled locations from GeoGuessr and downloaded a set of four images spanning the entire panorama, totaling 400,000 training images. Using their geocells, PIGEON creates caption components for each image based on location, climate, compass direction, season, and traffic. They then utilize a loss equation to aid their predictions. It quickly becomes clear how PIGEON was able to achieve its impressive accuracy.

Our main takeaway from these three sources is to use a Geocell Classification model while creating a map of the US for our model.

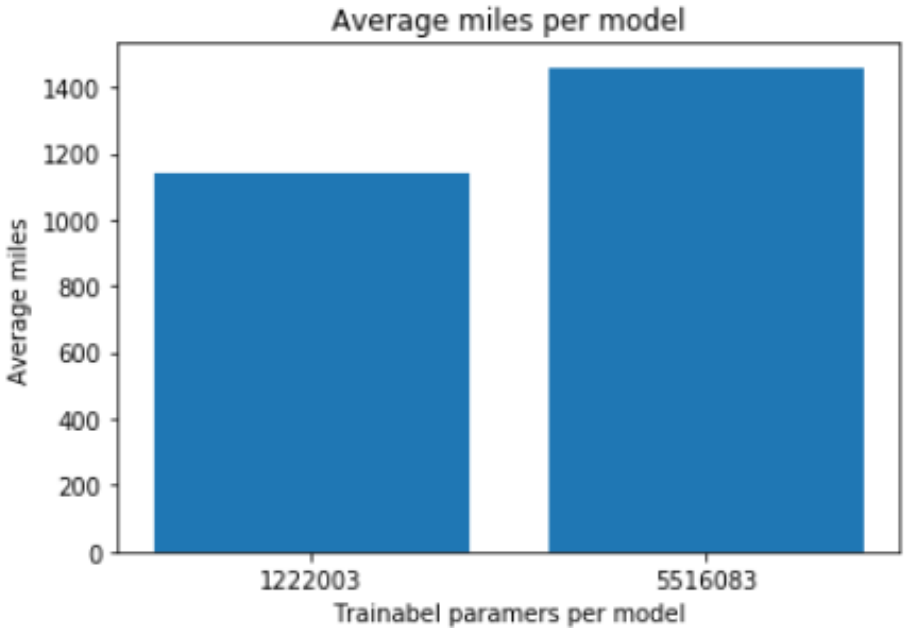

Figure 2.1: Nirvan66 Final Accuracy

2.4 Proposed Method 1: Geocell Classification Model

2.4.1 Method Overview

This method will categorize pieces of the United States into geographic categories or geocells. These geocells will then be used as classes in the classification problem that we will solve with our model. We will limit our classification to 128 classes to make sure the problem does not become overly complex.

2.4.2 Dataset

Our dataset consists of 100,000 latitude-and-longitude-labeled street view panorama images of size 600×600 as seen in Figure 1.1. The images were obtained from the Google Street View API [6] via random sampling of all locations with Google Street View coverage. The images show 90 degrees of view at a random heading between $0°$ and $360°$. 10,000 of these images were randomly chosen for our validation set and another 10,000 for the test set.

2.4.3 Model Architecture

We will be using a sequential convolutional neural network since our input is images.

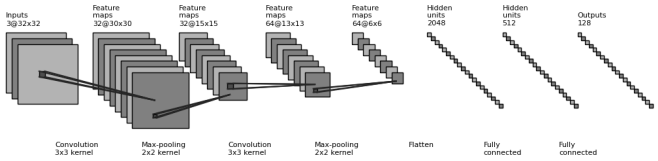

Figure 2.2: Initial Architecture For Geocell Classification

As seen in Figure 2.2 we will be inputting 32×32 images in all 3 channels and doing alternating layers of convolution with 3×3 filters and 2×2 MaxPooling. We chose to do only 2 to keep as much detail as we could without making the computation required to be excessive. We chose small sizes for our convolutional layer kernels because we believe that smaller details such as road lines or light posts will be critical to the geolocation of the image and larger kernels may not extract those small features very well.

We will be using ReLU for all of the activation functions in the hidden layers and soft-max as the activation on the output layer. Cross entropy will be used to calculate the loss during training.

As we do experimentation we may change the filter size on the convolution and max pooling layers to better fit our problem. Additionally, if time and compute power allow we may decide to increase the size of the input image to gather more detail to inference on.

2.4.4 Data Preprocessing

The step of data preprocessing that will need to be done is to plot all of the images' longitude and latitude using ArchGIS so that we can create our geocells as seen in Figure 2.3. We will then merge adjacent cells with the lowest total number of samples repeatedly until 128 cells remain. This approach is similar to what was used in the PIGEON [7] model. The goal is to make classes that have roughly equal amounts of data points to provide better predictions in areas where more images are taken by sacrificing precision in areas with few pictures. This geocell creation also helps to define more realistic boundaries between locations which can help isolate distinct features of the region.

The next step of preprocessing will be labeling the images based on the geocell they appear in, this can be done easily through ArchGis.

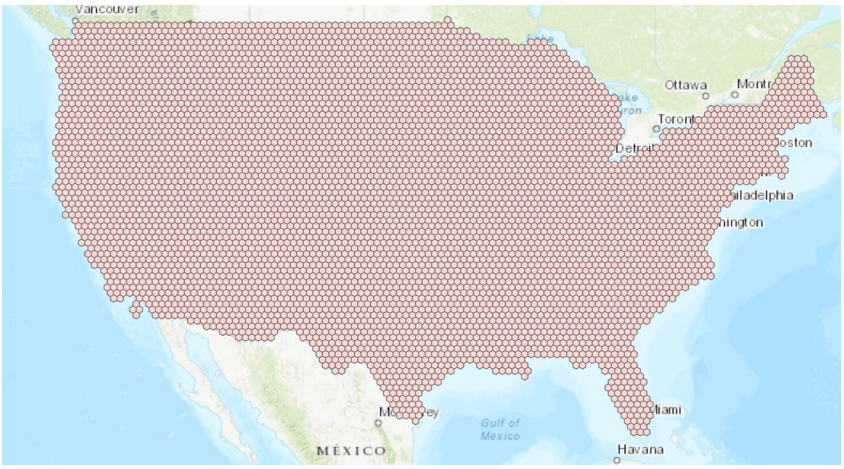

Figure 2.3: Map of all geocells before merging

We will then load the images into Swan in order and preprocess the images by scaling them from the original 600×600 to a more manageable 32×32 so that they are ready to be used for training.

Additionally, the data will need to be split into train, test, and validation sets of $80,000$, $10,000$, and $10,000$ images respectively.

2.4.5 Evaluation and Comparison

There will be two metrics with which we will evaluate our model.

We will use the classification accuracy of the model e.g. the percent of time the correct geocell is chosen for the image. This will be what we use to compare hyper-parameter changes over different training runs.

To compare our model to others in the literature, we will have to use a different metric. Since most of the related works [13] [12] [7] use the Haversine distance between the actual point and the predicted point as their evaluation metric, we will compute the distance between the center of the geocell and the actual point to get a metric that can be compared to the other literature.

In comparison, we must also address the fact that every project model was done over a different scope. [7] covered multiple countries across different continents, [13] was only the United States, and [12] only predicted over 5 cities. This must be taken into account when comparing our results to other projects.

We also want to collect how well it does playing Geogussr where points are awarded given the formula $5000e^{\frac{-x}{2000}}$ where x is the distance from the actual location in kilometers.

2.4.6 Timeline

Since the dataset is already procured we will be able to immediately start preprocessing the data and classifying it into geocells.

The training of the model will be time-consuming as it is a complex task with many data points. We will likely need to adjust the model architecture after seeing preliminary results. Thus the limitation of the project scope the the United States will allow us time to complete the project with the rest of the drawbacks in mind.

We believe that this project under these constraints can be completed within the semester. If there is more time, another region may potentially be added or the number of geocells expanded.

2.4.7 Potential Problems and Risks

Since we limited the number of cells that we will use for classification there may be certain geocells that have little or no data. This would mean that it would not be able to generalize to that area very well. This was one of the issues faced by [13] when developing their model with rigid rectangular geocells.

Additionally, since all images are from Google Street View, it would be difficult to apply this model to images taken in areas with no roads that would not have been mapped by Google.

2.5 Proposed Method 2: Regression with Transfer Learning Model

2.5.1 Method Overview

This method will attempt to solve a regression problem rather than a classification problem. This addresses the problem of only being able to guess in one location within a specific area but adds a much higher degree of complexity to the problem. Importantly, [7] showed evidence that using transfer learning helped them to increase the accuracy of their model. Therefore we would use transfer learning to help with feature extraction, helping to ease the computational resources and time it would take to implement this model. We have chosen to use MobileNet [9] as the base for the transfer learning since Keras has built-in support for it and we believe object detection feature extraction would be useful in finding features that would give hints to the location of an image.

2.5.2 Dataset

The dataset is the same as mentioned in part 2.4.2

2.5.3 Model Architecture

We will be adding two dense layers with 1024, 512, and 128 nodes to the head of the MobileNet base model and locking the other layers to make their parameters untrainable.

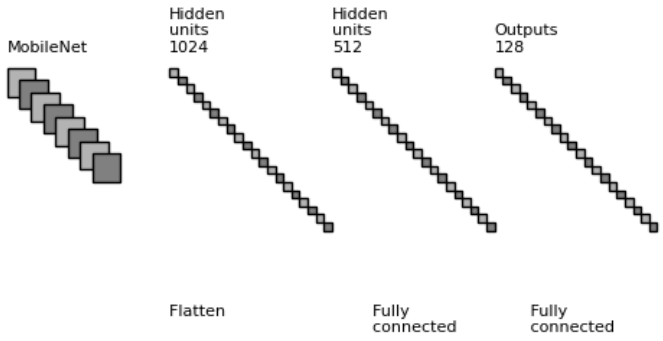

Figure 2.4: Mobile Net Transfer Learning Architecture

The input size of the images for MobileNet is much larger at 224×224 but since we do not have to train as many convolutional layers for feature extraction, we can still use it to improve our accuracy without using large amounts of computing.

The two added hidden layers will use ReLU activation and the output layer will use linear activation since we are treating the problem as regression.

Mean squared error (MSE) was chosen as the loss function since one of the main motivations of this project is for the bot to play Geogussr and Geogussr punishes outliers more heavily. MSE models similar behavior by making larger errors much more impactful than smaller ones. We may also use a custom Haversine distance loss formula based on results from our initial training.

We may unlock layers to train in the MobileNet base model if preliminary results show that it would be beneficial to do so.

2.5.4 Data Preprocessing

This approach requires much less data preprocessing than Method 1.

The images are already labeled by longitude and latitude so no further action on the labels is needed. The pixels of the images need to be normalized to the range [0,1] and the images need to be resized to 224×224. The resulting dataset can then be stored so the preprocessing step only needs to be done once.

Additionally, the data will need to be split into train, test, and validation sets of $80,000$, $10,000$, and $10,000$ images respectively.

2.5.5 Evaluation and Comparison

The main form of evaluation used in this model would be the average Haversine distance between the predicted point and the actual point. As mentioned in Section 2.4.5 most of the other literature on this problem uses this metric to evaluate the performance of their model.

We also want to collect how well it does playing Geogussr where points are awarded given the formula $5000e^{\frac{-x}{2000}}$ where x is the distance from the actual location in kilometers.

In similarity to Method 1, each of the related projects [7] [13] [12] all have separate scopes, and thus comparing the results must be done in context.

2.5.6 Timeline

Since the data needs less preprocessing in this method compared to method 1, the timeline is more doable within the semester. In addition, using transfer learning will reduce training time in comparison to training all the parameters since the new head is much less complex than the architecture in the previous method.

Although the time and compute cost is most likely less than that of method 1, training and testing the model on $100,000$ images will still be time and compute-heavy and require good time planning.

2.5.7 Problems and Risks

The main problem with this method is that changing the method to regression greatly increases the complexity of the model. It also introduces more chances for the model to get stuck in local minimums such as returning the average of all labels.

Another concern is that none of the pieces of literature we read used only a regression model to generate their predictions. The closest was [7] PIGEON who used a regression model after they located the predicted geocell to help predict an even more precise location.

2.6 Conclusions

In conclusion, our project aims to enhance the accuracy of GeoGuessr players' location predictions within the United States using a Convolutional Neural Network applied to Google Street View imagery [6]. After careful consideration and analysis of the two proposed methods, our group is inclined toward method 1, which utilizes Geocell classification to categorize locations into geographic cells and determine an image's location.

Method 1 offers several advantages. By categorizing locations into geocells, we simply the problem into a classification task, which is far more straightforward to implement. Furthermore, geocells allow us to handle areas with varying densities of Street View imagery effectively. However, we understand the potential challenge of sparse data in certain cells and the need for careful preprocessing and merging of cells.

Additionally, we are considering incorporating Transfer Learning into method 1 to leverage pre-trained models such as MobileNetV2 for feature extraction, which could potentially enhance the model's performance.

On the other hand, method 2, which employs regression with transfer learning, presents a more nuanced approach aiming for pinpoint accuracy in predicting longitude and latitude directly. While this model may potentially achieve higher precision, it also introduces greater computational complexity and requires more extensive data preprocessing and model fine-tuning.

Method 1 simply offers a balanced approach that addresses the project's objectives while considering the constraints of time and resources. By focusing on geocell classification and potentially integrating Transfer Learning, we may develop a robust model.

Overall, our group is leaning towards method 1, which better aligns with our project goals and constraints. With careful implementation and experimentation, we hope to create a robust model with a high degree of accuracy.

Table 2.1: Contributions by team member for Milestone 2.

Team Member	**Contribution**
Dakota Andrews	Worked on Abstract and introduction
Gage Cammack	Created our graphs and references
Joseph Seibel	Developed and wrote our proposed models
Tong Wu	Document write-up for Problem Specification, Related Works, and Conclusion

Chapter 3

Milestone 3: Progress Report 1

3.1 Introduction

After careful consideration of how to approach our GeoGuessr Bot, our team decided to proceed with the Geocell Classification Model from Section 2.4 utilizing a convolutional neural network (CNN). For our first experiment, this method offers a balanced approach that addresses our time constraints while giving us potentially decent results.

To accomplish this, we first pulled our images from Google Street View [6], totaling 100,000 images. Then we partitioned our data and created 3 different geocell maps with 97, 50, and 25 unique zones (See Figures 3.1, 3.2, and 3.3, respectively). Following this, we created our first CNN model utilizing 5 hidden layers with the use of max pooling, normalization, and dropout. After training and testing our model on the three maps, we yielded Top-1 validation accuracies of 4%, 8%, and 15%, respectively.

3.2 Related Work

As discussed in Section 2.3, our approach follows PIGEON's [7] method of classifying geocells. We believe PIGEON's approach to be the most effective method for our problem as they yielded the highest accuracy from our research. We are likewise creating a CNN model but our main difference in approach lies in how PIGEON captioned their geocells and their scope. PIGEON captioned components for each image based on location, climate, compass direction, season, and traffic, which creates a stronger model. However, with our limited time and scope, we can only label the longitude and latitude of our data. The second difference is the scope of the project. PIGEON created geocells globally,

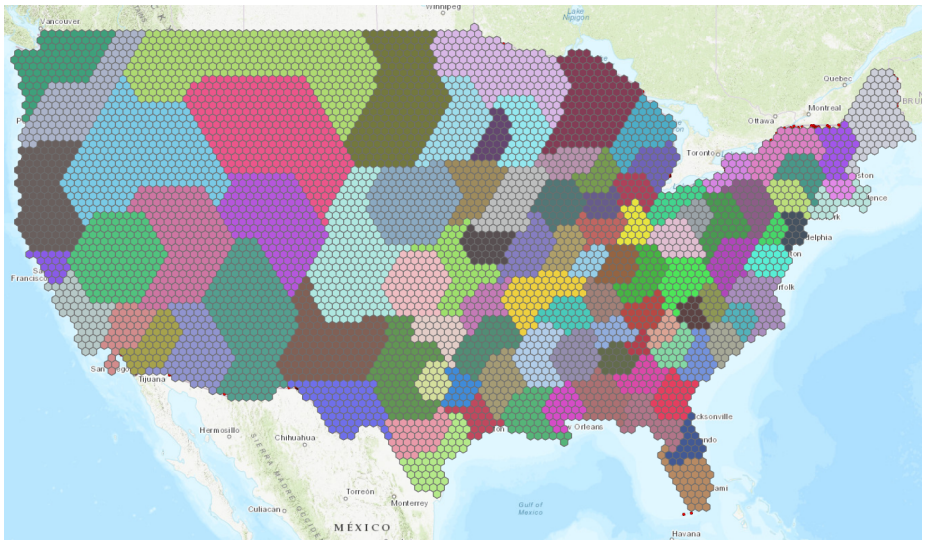

Figure 3.1: Geocell map with 97 unique zones

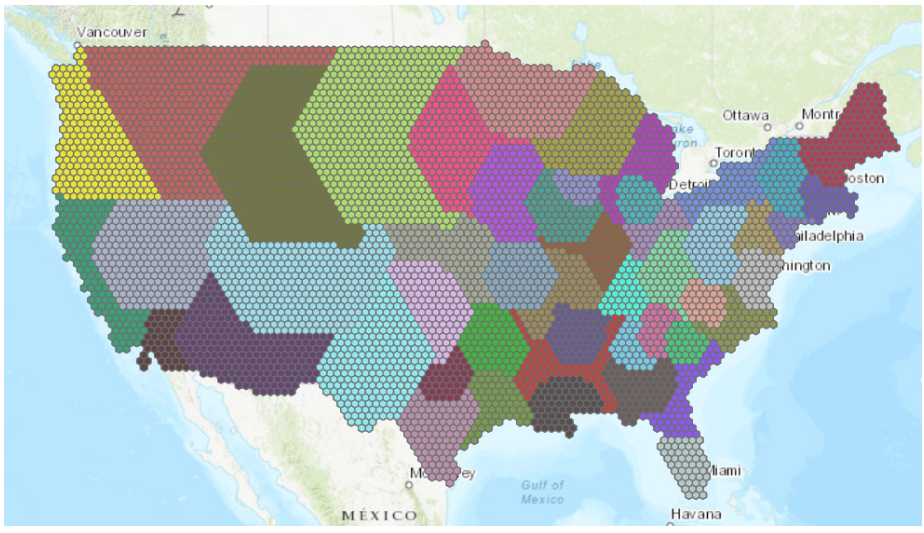

Figure 3.2: Geocell map with 50 unique zones

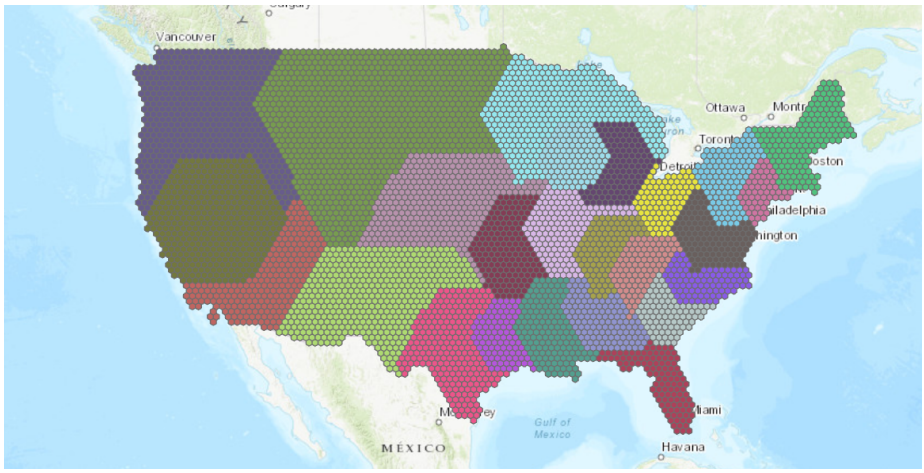

Figure 3.3: Geocell map with 25 unique zones

whereas we are focusing solely on the United States. In that sense, our scope follows nirvan66's project of staying in the United States.

3.3 Experimental Setup

As a team, we each pulled 25,000 unique panorama images from the Google Street View [6]. This totals our random dataset of 100,000 images of sizes 640×640. Images pulled also have their longitude and latitude coordinates recorded representing our data's labels. Furthermore, we partitioned 80,000 images as training, 10,000 images as testing, and 10,000 images as validation.

To repeat our process of gathering data, we first created a Google Cloud account [5] and created a Google Street View API key to randomly pull images with a script. After downloading the images, we uploaded the data onto Jupyter so we could partition our data and create our geocells using ArcGIS PRO [4]. The maps in Figures 3.1, 3.2, and 3.3 are made using ArcGIS Pro.

We then created a CNN model designed for image classification. The model consists of convolutional layers with rectified linear unit (ReLU) activation, batch normalization, and max-pooling layers. The architecture begins with a convolutional layer with 32 filters of size 3×3, followed by batch normalization and max-pooling layers. Subsequently, another convolutional layer with 64 filters and identical activation and normalization functions is used. Following max-pooling, the feature maps are flattened and passed through a dense layer of 512 neurons with ReLU activation and batch normalization. Dropout regularization is employed to mitigate overfitting. The final dense layer uses softmax activation, yielding class probabilities. See Figure 3.4 for a visualization of our model.

For our first set of runs we used categorical cross entropy as our loss function and for our second set we used categorical cross entropy with a spatial penalty to punish wrong guesses that are farther away. The spatial penalty was calculated via multiplying by the Haversine distance between the centers of the two geocells. The formula for Haversine distance is

$$h = \sin^2(\phi_1 - \phi_2) + \cos(\phi_1)\cos(\phi_2)\sin^2(\lambda_1 - \lambda_2)$$

where $(\phi_1, \lambda_1), (\phi_2, \lambda_2)$ are longitude and latitude points. Both models used Adam as the optimiser.

We used two sets of hyper-parameters in our runs. After noticing that there was extreme overfitting in our first run we greatly increased the dropout rate and regularization in order to address the issue.

Table 3.1: Hyperparameters for Neural Network Models

Hyperparameter	**Set 1**	**Set 2**
Learning Rate	0.001	0.0001
Dropout Rate	0.5	0.7
Regularization Rate	0.001	0.01
Epochs	10	20

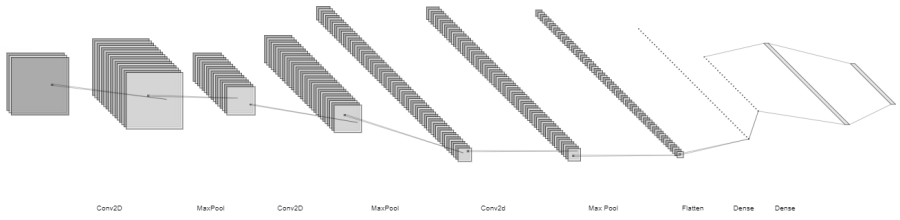

Figure 3.4: First CNN Model Attempt

3.4 Experimental Results

As seen in Figure 3.6 and Figure 3.5 the results of our first set of parameters was poor. The accuracy was only about 3 times that of guessing at 14 percent top 1 and about 40 percent top 5. It was very obviously overfitting the data as the training loss and accuracy became disconnected from that of the validation. Figurine 3.6 shows large bands in the confusion matrix indicating it was guessing a small number of classes more often than all the rest. We added

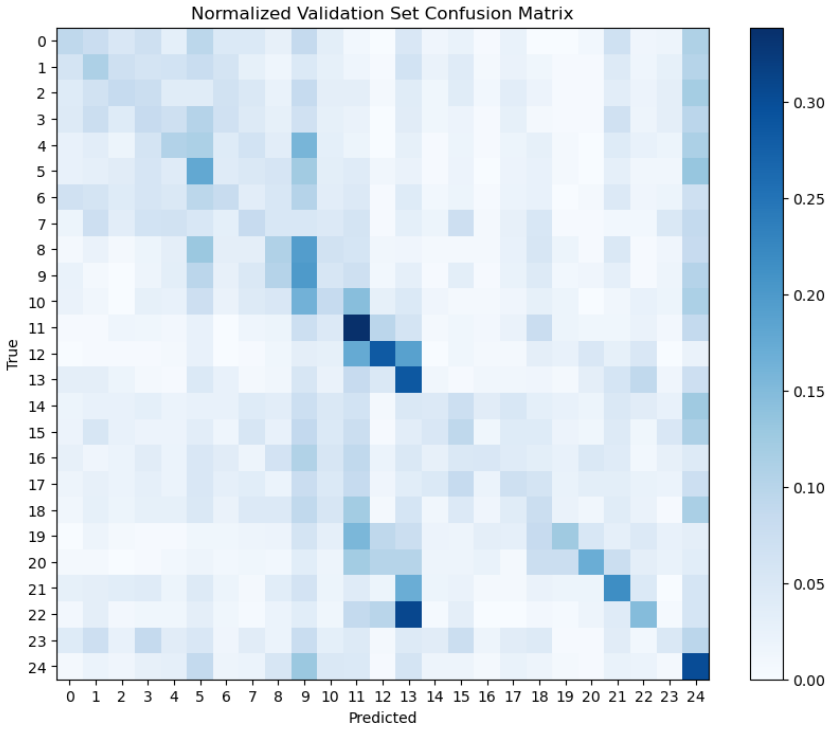

Figure 3.5: Confusion Matrix First Attempt for 25 Geocells

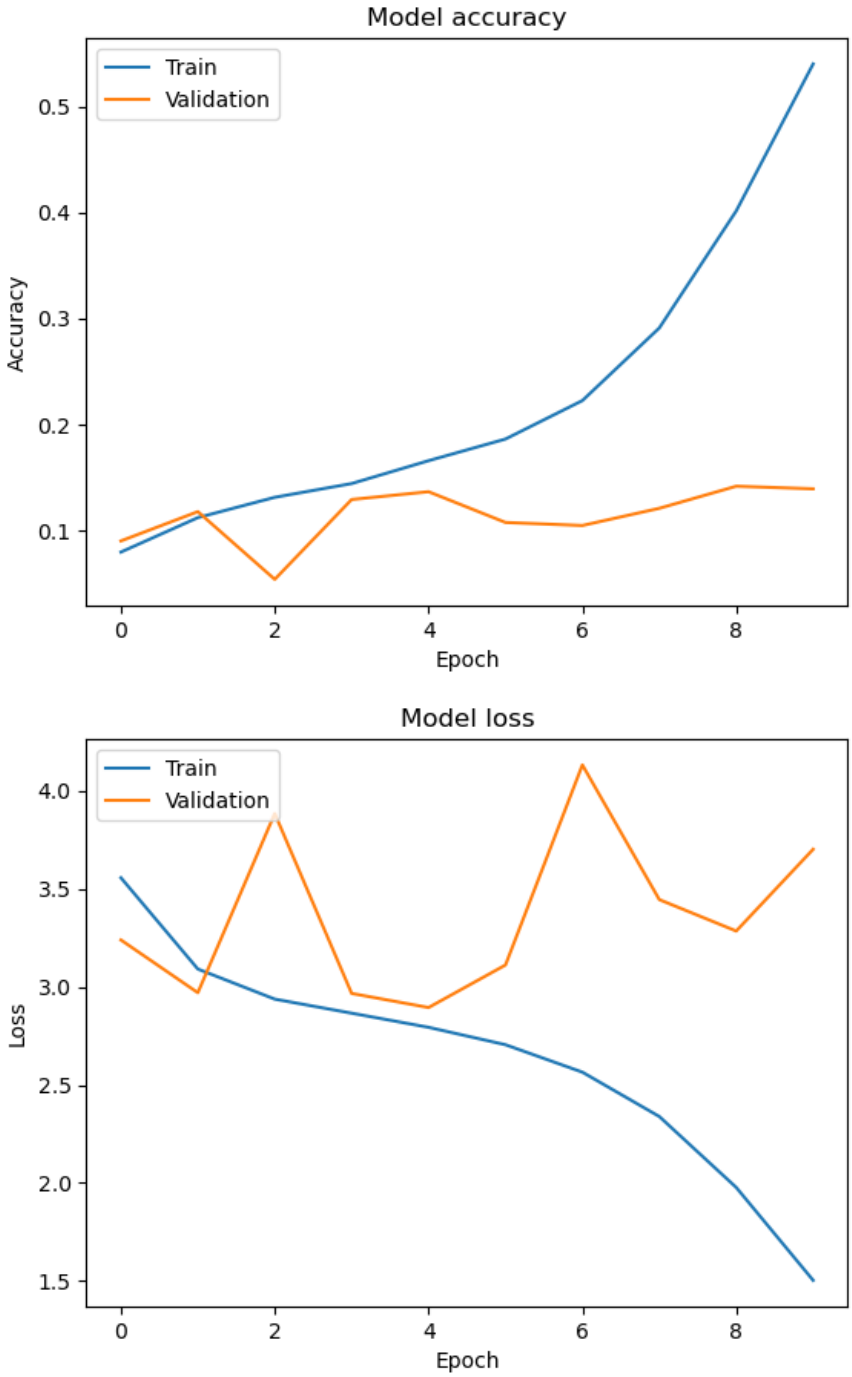

Figure 3.6: Accuracy and Loss Graph First Attempt for 25 Geocells

more regularization and dropout to attempt to address the overfitting in the next run.

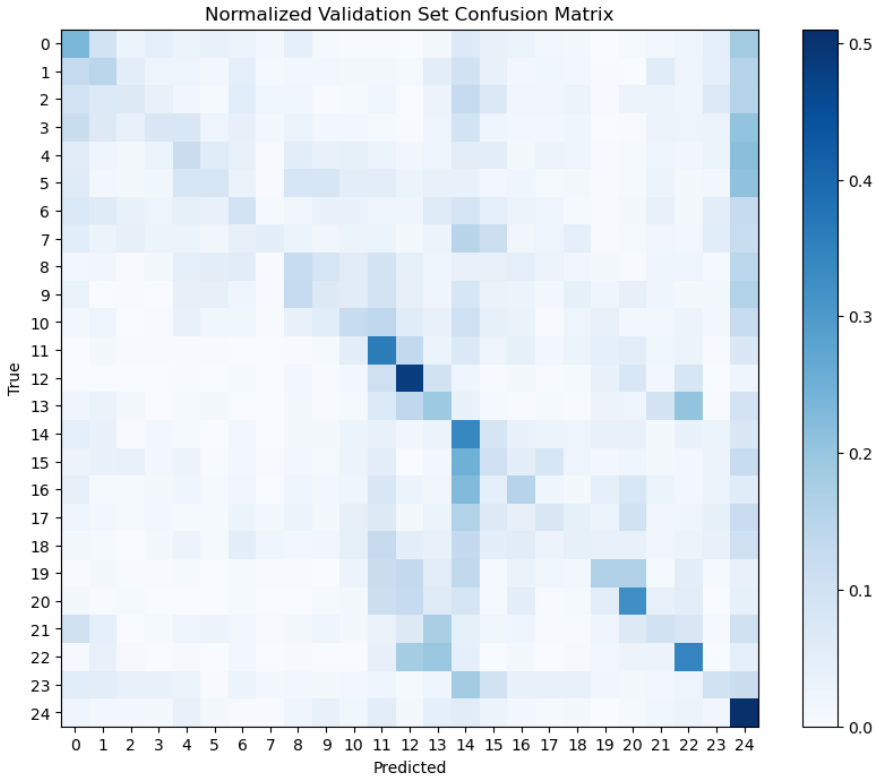

Figure 3.7: Confusion Matrix Second Attempt for 25 Geocells

The results of the next attempt seen in Figure 3.8 and Figure 3.7 are better than the first run. We achieved an 18.5 percent top 1 accuracy and an over 55 percent top 5 accuracy. Although this is still subpar it was an improvement from the previous model and the problem with overfitting seemed to decrease.

3.5 Discussion

As seen in 5.2 our first attempt at this problem yielded subpar results. We knew this classification problem was going to be difficult because hard-to-detect features would have to be used to determine the location of the image. We believe that part of the reason for the lack of performance is that we do not have enough data. Even though we have 100,000 images of the United States, there are many gaps in our data or locations with only single pictures to base the predictions on. To address this issue we plan on getting an additional 100,000 images as soon as the monthly quota rolls over on Google Cloud.

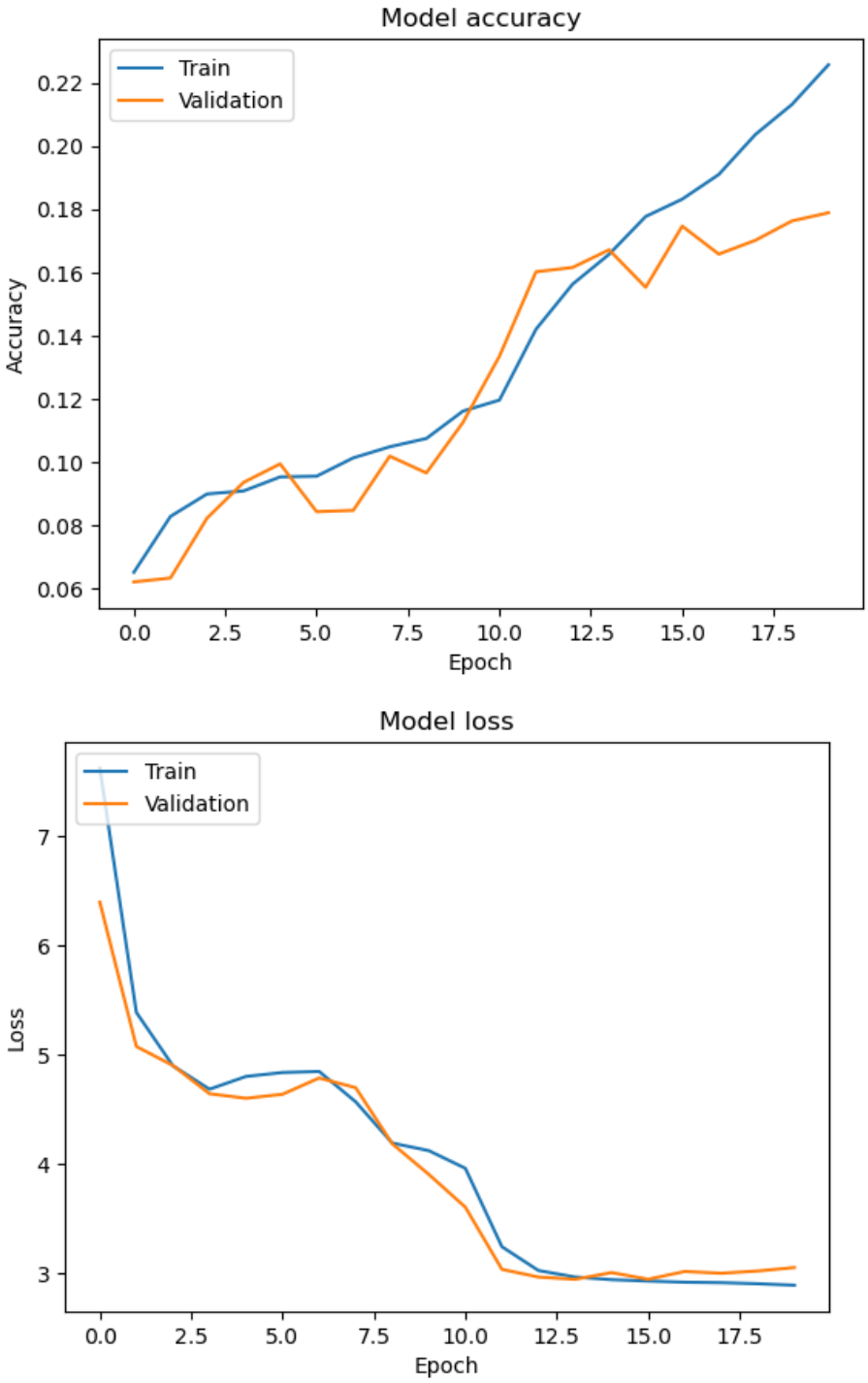

Figure 3.8: Accuracy and Loss Graph Second Attempt for 25 Geocells

Table 3.2: Accuracy Results for Neural Network Models

Model	Metric	Model 1	Model 2
Top-1 Accuracy (%)	Training	53.42	22.54
	Validation	14.83	18.92
Top-5 Accuracy (%)	Training	87.54	61.84
	Validation	38.31	59.13

It may also be the case that the model is not complex enough to pick up on the features in the images. So using transfer learning may be a good improvement on our current solution.

We may also want to revisit our geocell creation method. As it stands we use the built-in region builder in ArcGIS Pro that creates cells with no context. It may be useful to use already existing borders to help with the creation of these cells to avoid oddly shaped cells that split boundaries. This may help improve performance and allow us to create smaller subdivisions to classify our images into.

3.6 Work Plan

From our analysis, we may need to adjust our model by adding an extra layer or implementing transfer learning. This may prevent our model from overfitting and improve its learning. Likewise, as the new month approaches, we can pull another 100,000 images using the method from Section 3.3. This gives us more data to work with and allows for a greater generalization across the United States.

Furthermore, it may potentially help us to re-adjust the geocell creation method by adding some form of context. Currently, we are only creating the geocells, and our data only contains the longitude and latitude of the image with no context, unlike PIGEON's approach [7]. If we choose to add some context, it may improve our model's performance.

We are also going to attempt to use a regression model to solve this problem and compare it to this model by calculating the average distance from the correct point by using the center of the geocells in the classification models and the outputted prediction in the regression model.

Our timeline for the rest of the semester looks like the following:

1. Preset - April 8th: Scrape more data and start building/training regression model

2. Week of April 9th: Test several regression model architectures and test more geocell architectures with more data and refined geocell partitioning.

3. Week of April 16th: Continue to refine architecture on best preforming model type.

4. April 19th: Milestone 4 due

5. Post Milestone 4: Continue to refine model and start working on presentation and chrome extension to let model play Geogessr matches if time allows.

3.7 Conclusion

In conclusion, our first attempt at developing a GeoGuessr Bot using the Geocell Classification Model with Convolutional Neural Network (CNN) ended subpar and leaves much room for improvement. We initially decided on this method due to PIGEON's [7] promising results. However, we failed somewhere in our implementation and yielded modest results with validation accuracies of 4%, 8%, and 18%.

We attribute this outcome to several potential factors, including nuanced features in the images and the limited scope of our data labeling compared to PIGEON.

Our next step is to address the gaps in our current dataset. We plan to acquire another 100,000 images for our next attempt and then update our model, and add transfer learning to capture image features. Furthermore, we intend to refine our geocell creation method by considering borders to improve cell shape and performance. These adjustments, moving forward, will hopefully provide us with better results.

Finally, although our initial results were disappointing, they serve as a basis and a first attempt of our GeoGuessr Bot. Through refinement in our next attempt, we aim to enhance our model's performance and achieve a higher validation accuracy.

Table 3.3: Contributions by team member for Milestone 3.

Team Member	**Contribution**
Dakota Andrews	Worked on the writeup, and trained the model.
Gage Cammack	Organized the data, trained the models, and created visual graphs.
Joseph Seibel	Created the models, organized the data, and trained the models.
Tong Wu	Majority document write-up and organization.

Chapter 4

Milestone 4: Progress Report 2

4.1 Introduction

Our goal is to construct a Geoguessr Bot utilizing a convolutional neural network and employing the Geocell Classification tactic outlined in 2.4. For the first iteration of our model, we partitioned the maps into 97, 50, and 25 unique zones (see Figures 3.1, 3.2, and 3.3). After training and testing the initial model on the three different maps, we yielded Top-1 accuracies of 4%, 8%, and 15% respectively.

For the second iteration of our model, a few changes were made to the structure of the network. We deepened our model from the previous milestone and increased the size of the images. We also attempted a transfer learning approach that yielded promising results. As a result of these changes, we yielded Top-1 accuracies of just over 24% on the model trained from scratch and over 34% on the transfer learning model.

4.2 Related Work

After analyzing the results from our last milestone, we realized that the geocells we had created were sub-optimal. We once again reviewed the PIGEON paper [7] to try and replicate their geocell creation exactly. Unfortunately, the topic is only briefly touched on in the paper and we could not reproduce their method.

4.3 Experimental Setup

As seen in Table 4.1 we used different sets of hyperparameters for the model that used transfer learning and the one that we did from scratch. When training

the transfer learning model we noticed that it was much more prone to overfitting than the model trained from scratch so we used increased dropout and regularization in the dense layers to mitigate it.

In the previous milestone, we suspected that we were not training the models for long enough so we increased the early stopping patience from 5 to 15 and the max number of epochs to 100. Additionally, we increased the size of the images we were training on from 64×64 to 224×224. This combined with making our original model deeper greatly increased training times.

We also increased our dataset size since the last milestone. We obtained roughly 97,000 more images on top of our original 100,000 images to train test and validate.

Additionally, we used the ArchGIS clustering tool to try and get more compact geocells but it had very little effect on the outcome. As seen in Figure 4.3 there are still some very irregular cells.

For the transfer learning model for the base, we used MobileNet from the Keras package. We locked all the layers on the base model and only trained on the dense head that we added. We used the custom Haversine loss function mentioned in Section 3.3 on both models. The loss punishes wrong classifications based on the location of the centers and performed better on previous models. All activations were ReLu on all trainable layers.

Table 4.1: Hyperparameters

Hyperparameter	**From Scratch Model**	**Transfer Learning Model**
Learning Rate	0.001	0.0001
Dropout Rate	0.3	0.5
Regularization Rate	0.001	0.01
Epochs	100	100
Early Stopping Patience	15	15

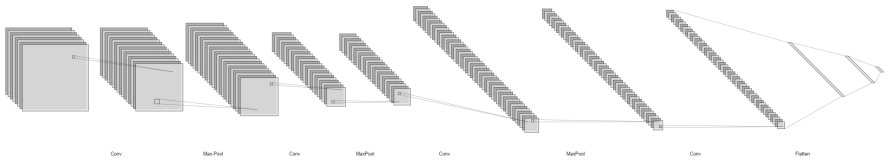

Figure 4.1: Nontransfer Model Architecture

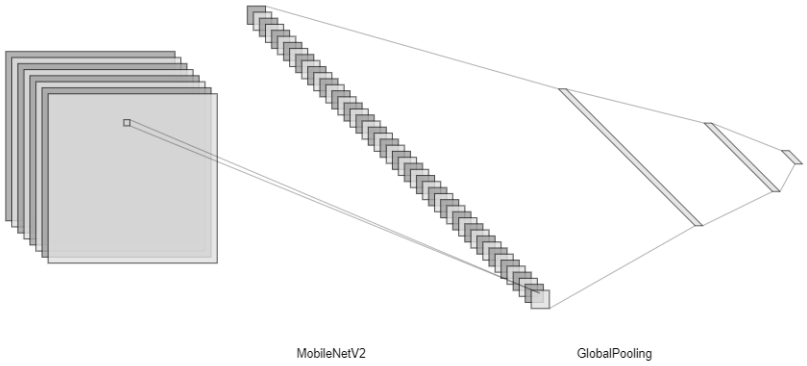

Figure 4.2: Transfer Model Architecture

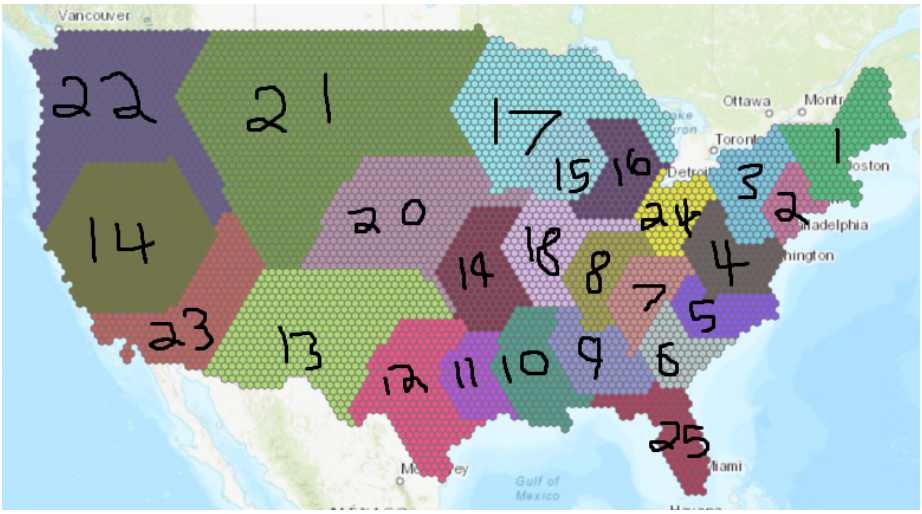

Figure 4.3: Labled Geocell Map

4.4 Experimental Results

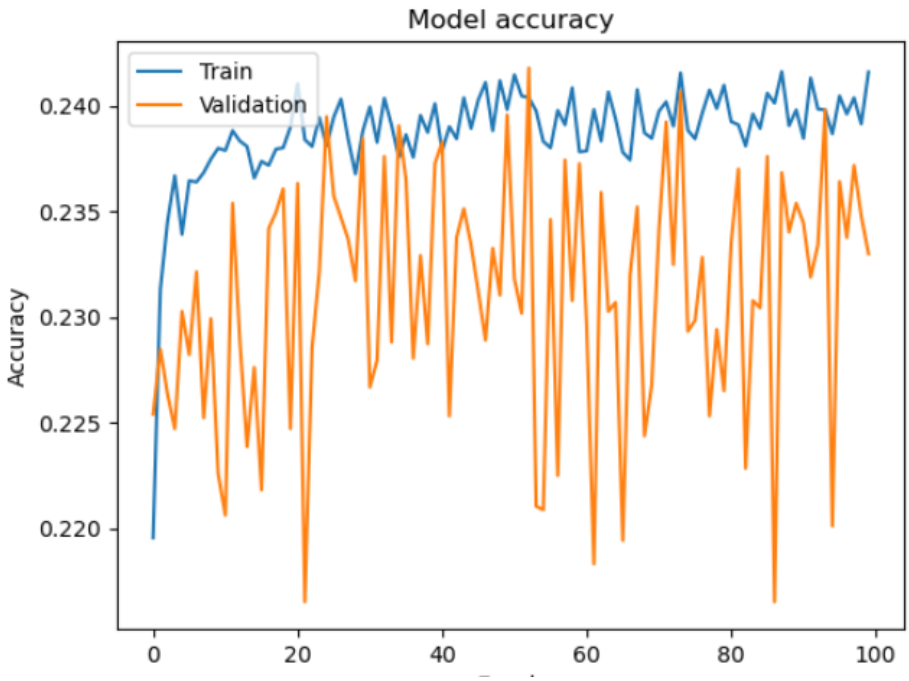

Figure 4.4: Nontransfer Architecture Accuracy Graph

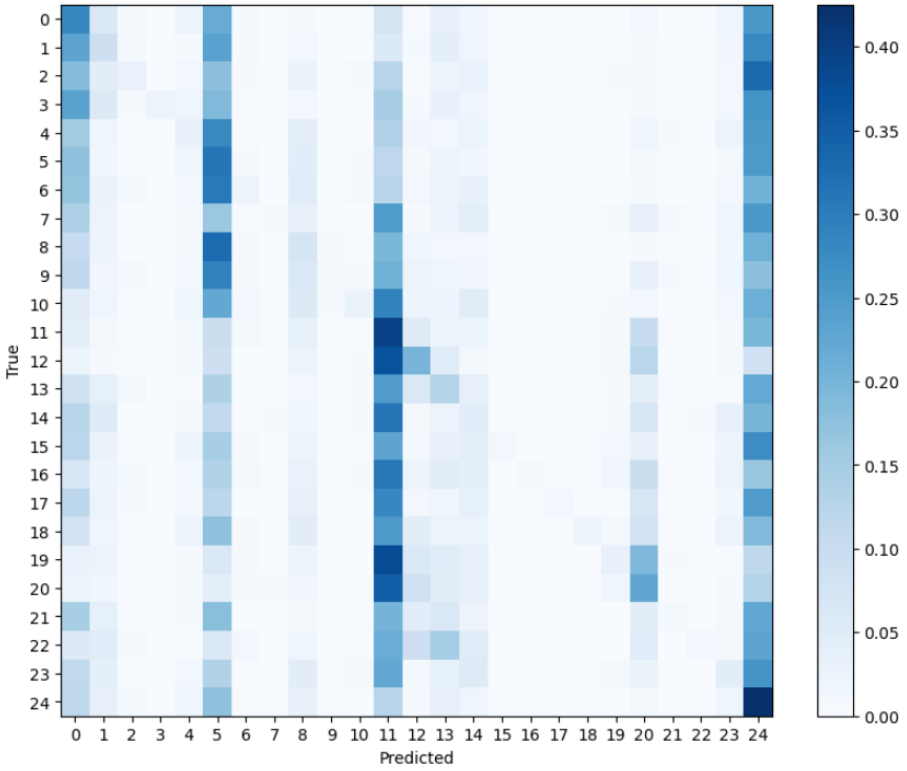

Figure 4.5: Nontransfer Architecture Confusion Matrix

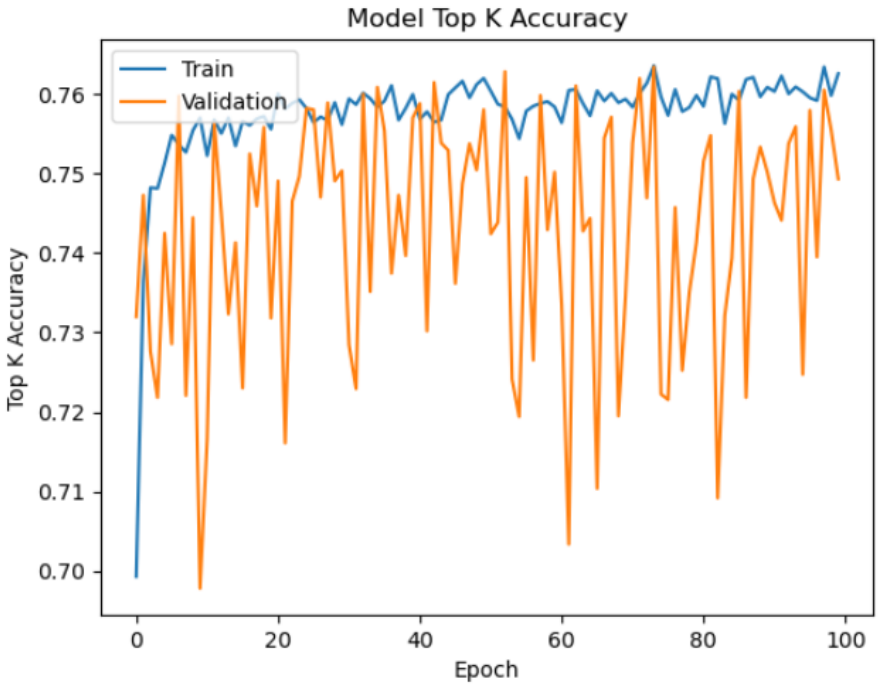

Figure 4.6: Nontransfer Architecture Top 5 Accuracy Graph

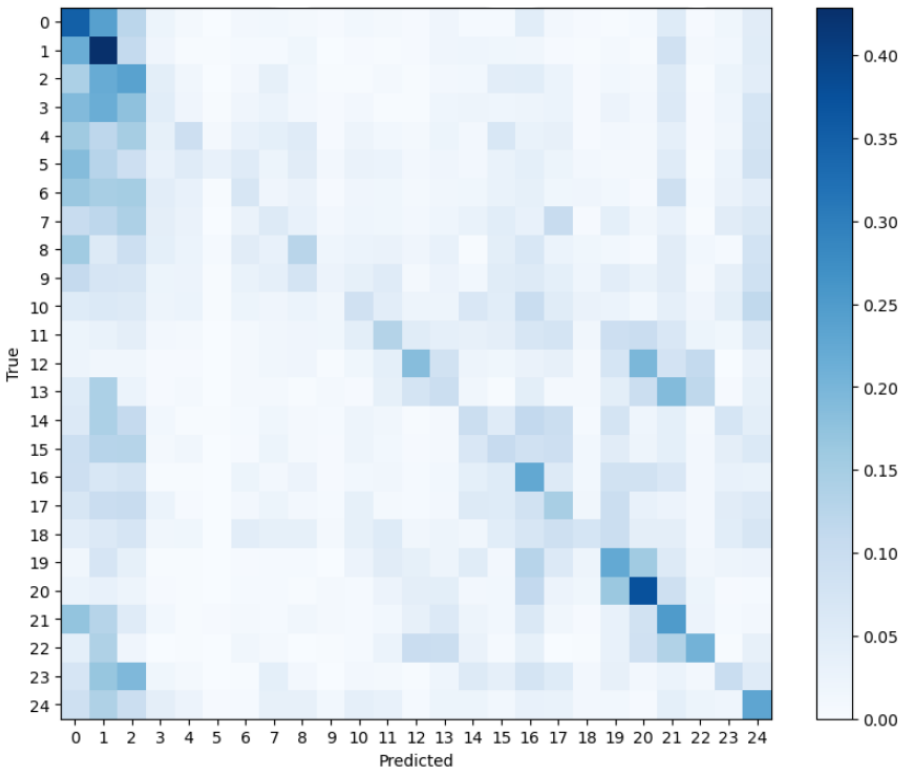

Figure 4.7: Transfer Architecture Confusion Matrix

Due to a mistake made in the Python file that was submitted to the GPU job that went unnoticed, we failed to get the graph of the accuracy history for the transfer learning model. The mean distance metric was obtained for the models by treating the weighted center of all points in a geocell as the predicted point for a particular sample.

Table 4.2: Accuracy Results for Neural Network Models

Model	**Metric**	**Non-Transfer Learning**	**Transfer Learning**
Top-1 Accuracy (%)	Training	24.44	43.68
	Validation	23.38	32.89
Top-5 Accuracy (%)	Training	76.89	95.75
	Validation	74.74	85.34
Mean Distance (miles)	Validation	542.13	419.83

Table 4.3: Comparison to Benchmarks

Model	**Our Model**	**PIGEON**	**GeoGuessr AI**	**Human**
Mean Distance (Miles)	419.83	25.19	1134.56	≈ 600
Scope	USA	World	USA	USA

4.5 Discussion

Our models improved by a noninsignificant margin and the transfer learning approach is proving to be very promising. There are still issues that need to be addressed such as the tendency for the model to classify all samples into only a small subset of the possible output classes. This trend can be seen very obviously in 4.5 where cells 1, 6, 12, 21, and 25 are predicted for the majority of the samples. As seen in 4.3 These cells are in different regions of the United States which suggests that the model is detecting and learning some differences between the images but has found it optimal to classify samples into only one cell in each region. This trend is not as apparent when using the transfer learning model which outputs the matrix 4.7 but it can still be seen.

We believe that our model is picking up and learning the relevant features but the issue with the low accuracy is due to the irregular cells. We have been unable to replicate the success seen by the PIGEON [7] network using its geocells but have been able to surpass GeoGussr AI [8] and human performance according to the average score on the USA-specific map on GeoGussr. There is also the consideration of scope when it comes to evaluating the results. PIDGIN having the entire world (that is covered by Street View) as the scope brings in advantages and disadvantages. The USA is a difficult country to score highly in due to the large stretches of land that is indistinguishable, thus being able to predict in places that have very distinctive features is an advantage. On the other hand, there is a much higher margin for error on guesses and the learning problem is much larger. These have to be taken into consideration when comparing PIGEON to other Geogessur models.

We hope to have more analysis on our transfer learning model after tweaking the hyper-parameters but want to focus heavily on the effect of the geocells on Mean Distance since it is a more important metric than accuracy in terms of the bot being able to win GeoGessur matches. Additionally, we do not wish to make the problem trivial by tailoring the previous results. Since the training times for our models have begun to surpass days due to the large amount of data and scaling up the image sizes we need to be very mindful of what changes we intend to make to our models since the semester is coming to an end.

Overall the model where we used transfer learning yielded much better results then the from scratch model so for future iterations we will most likely keep using transfer learning.

4.6 Work Plan

Since we suspect that the way we have our geocells layed out is negatively impacting the performance of the model we are shifting our focus to trying out other variations of these cells such as normal rectangle cells and cells based on environmental data which can be easily accomplished via ArchGis tools.

1. Week of April 21 - Experiment with new geocell layouts and fine tune hyper-parameters, start Presentation

2. Week of April 29- Final tweaks of model, continue working on presentation.
3. May 10 - Present results/Milestone 5
4. Post Milestone 5 - Continue to refine model and build chrome extension

4.7 Conclusion

Overall we made good progress on the classification problem. Although it has not had the same success as PIGEON, our model has surpassed average human performance and made a large jump in both Top 1 and Top 5 accuracy over the previous two weeks. We are hopeful that by finding a more optimal geocell layout we will be able to find a solution that minimizes the distance between the actual and predicted points and after the course has ended build a Chrome extension so that we can play against our model.

Table 4.4: Contributions by team member for Milestone 4.

Team Member	**Contribution**
Dakota Andrews	Worked on the writeup, and trained the model.
Gage Cammack	Organized the data, trained the models, and created visual graphs.
Joseph Seibel	Created the models, organized the data, and trained the models.
Tong Wu	Majority document write-up and organization.

Chapter 5

Milestone 5: Final Report

5.1 Introduction

Our initial goal was to create a Model that was capable of accurately predicting the location of a photo from Geoguessr. To achieve this we used Convolutional neural networks and Geocell Classification that we outlined in section **??**2.4. For our first iteration of the model, we partitioned the United States into 97, 50, and 25 unique Geocells (Figures 3.1, 3.2, and 3.3). After training and testing were completed the first model on the different maps Yielded a Top-1 accuracy of 4%, 8%, and 15% respectively.

For the second iteration of the model, we made a few changes to the structure of the network. We choose to deepen our model from Milestone Three3.5 and we choose to increase the size of the image. Most importantly we attempted to use transfer learning which yielded promising results for our model. After our changes were made to the model we achieved a Top-1 accuracy of 24% on a model trained from scratch and on the model that employed transfer learning we got a Top-1 of just over 34%.

For the last iteration of Milestone Five, we chose to keep the same model structure but change the Geocell allocation based on more data. We created a new geocell distribution that accounted for both the distribution of data points and ecological zones in the United States giving us 28 unique zones. We decided to downsample our data to create an equal distribution of data for each zone. Additionally, we unlocked the last two convolutional layers to fine-tune our model to our problem. The result of training and testing on the new model gave us a Top-1 accuracy of over 42%.

5.2 Related Work

As discussed in previous milestones, we realized that the geocells used for our model were not partitioned well enough. After having reviewed the PIGEON paper [7] we couldn't learn the method they used to create their geocells so we

chose to implement our new ideas. We created our new geocells based partially on ecological data; this is something that is not accounted for in the PIGEON paper.

5.3 Experimental Setup

As seen in Table 5.1 we choose to keep most hyperparameters of our transfer learning model from milestone 4 (see Table 4.1) because it performed the best out of all the model architectures we tested. Although the model still tends to underfit the training set we tried to mitigate this by unlocking the last two convolutional layers of the model to allow for some fine-tuning. Also, we choose to increase the dropout rate of our model to help combat the problem of overfitting with fine-tuning.

We decided to keep the current early stopping patience at 15 and make the number of epochs at 100 because It worked well for us in previous milestones. We chose to scale our photos down from $224x224$ down to $196x196$ to help make the training of our model go faster without losing as much data as we did when the images were $64x64$.

Additionally, we used ArcGIS's clustering tool to create a new map of zones that would help better group data together. Unlike previous Zones this time the tool also took into account the ecological zones of the United States. With this, we can see the tool-built zones that have more similar ecological features if you look at zones 21 and 12 in Figure 5.2 which follow the coastline that can be observed in Figure 5.3.

For our dataset, we decided to downsample as the new Geocells that were created didn't have an even distribution of data points across all zones. So to keep the even distribution of data across all Geocells we down-sampled to 4000 data points for every zone.

Table 5.1: Hyperparameters

Hyperparameter	**Ecological Zones Model**
Learning Rate	0.0002
Dropout Rate	0.75
Regularization Rate	0.1
Epochs	100
Early Stopping Patience	15

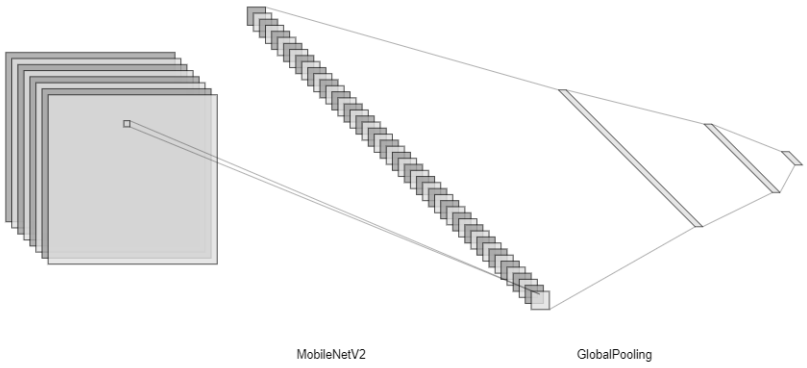

Figure 5.1: Model Architecture

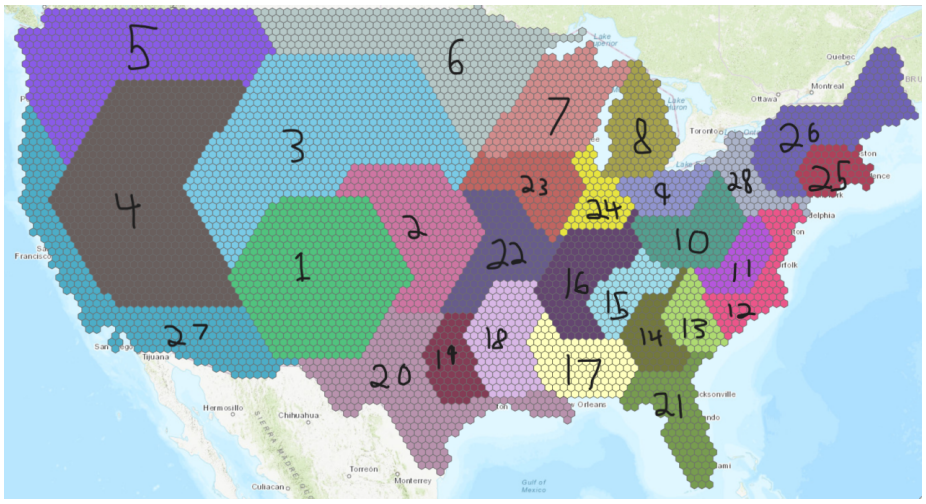

Figure 5.2: Labled Ecological Geocell Map

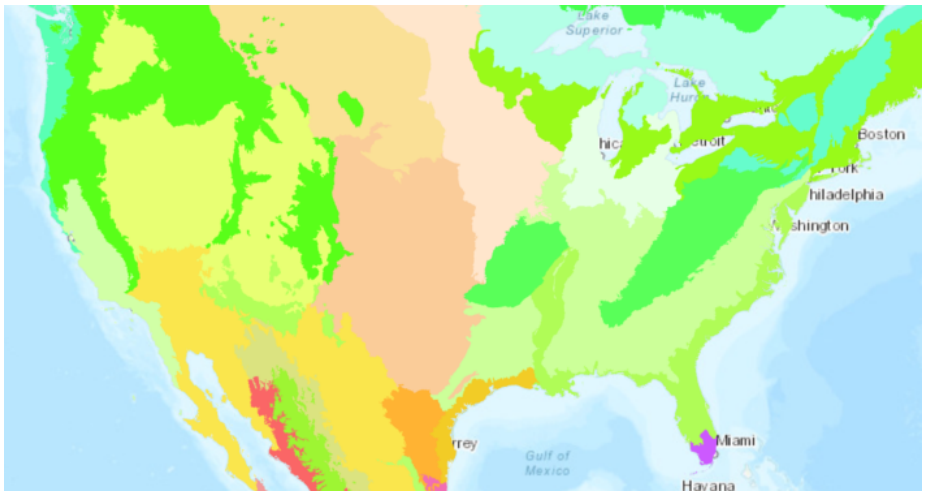

Figure 5.3: Ecological Zones Map form EPA

5.4 Experimental Results

As you can see in Figure 5.4 our model is overfitting but we are seeing a much smoother rate of improvement of our model than we saw in Figure 4.4. In addition, the confusion matrix that we see for our most recent model (Figure 5.5) does not show the bands that we see in Figure 4.5. From Figures 5.6, 5.7, and 5.8, we can see that the model is focusing most on the vegetation in the region such as grass and trees.

Table 5.2: Results for Neural Network Model with Ecological Zones

Model	**Metric**	**Ecological Zones**
Top-1 Accuracy (%)	Training	43.68
	Validation	42.78
Top-5 Accuracy (%)	Training	95.75
	Validation	91.56
Mean Distance From Correct Location (Miles)	Validation	132.49

Table 5.3: Comparison to Benchmarks

Model	**Our Model**	**PIGEON**	**GeoGuessr AI**	**Human**
Mean Distance from Correct Location (Miles)	132.49	25.19	1134.56	≈ 600
Scope	USA	World	USA	USA

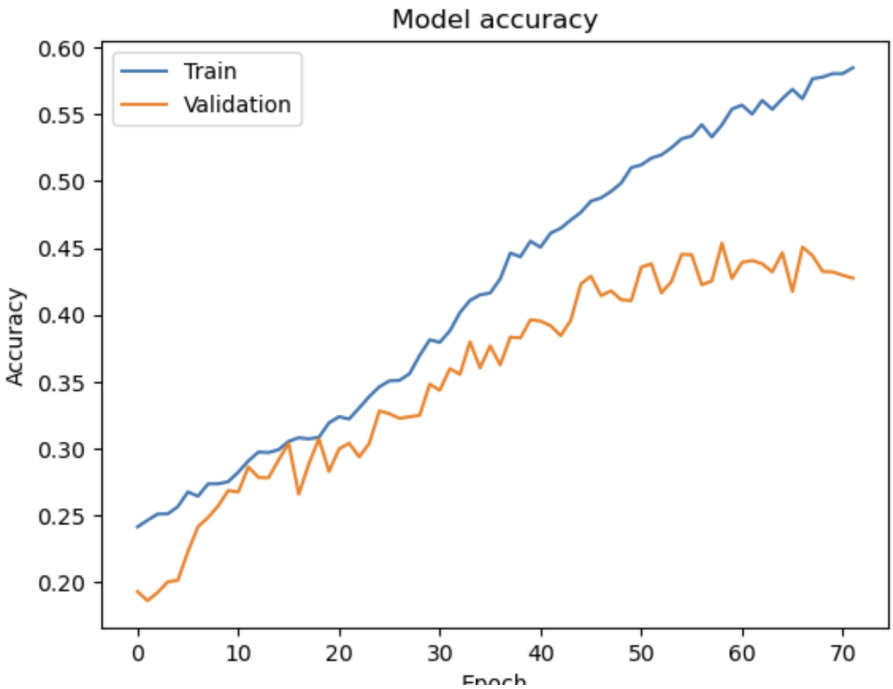

Figure 5.4: Ecological Zones Top 1 Accuracy Graph

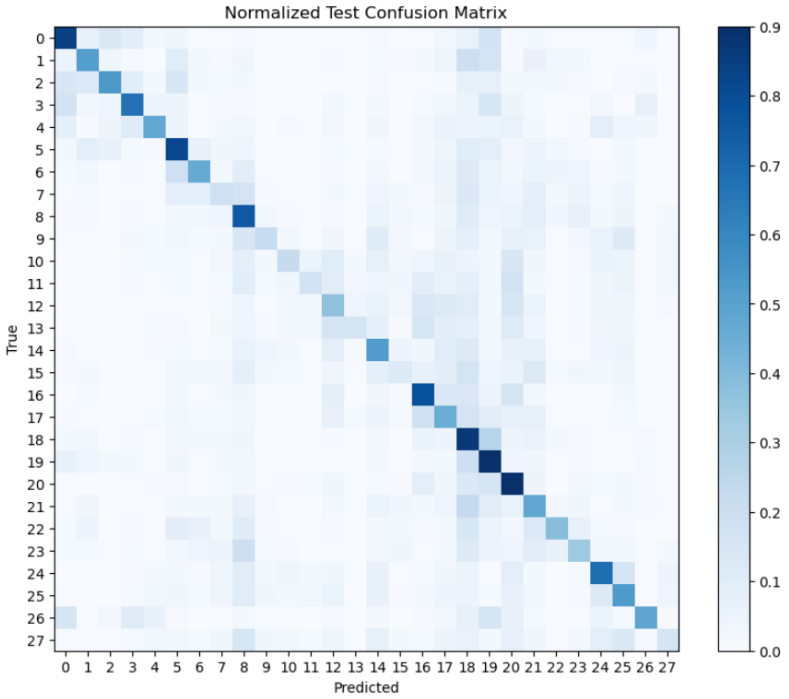

Figure 5.5: Ecological Zones Confusion Matrix

Figure 5.6: Gradient Camera Image 1

Figure 5.7: Gradient Camera Image 2

Figure 5.8: Gradient Camera Image 3

5.5 Discussion

With the newly added ecological data, our model achieved a Top-1 Accuracy of 42.78%, a Top-5 Accuracy of 91.56%, and a Mean Distance from the correct location of 132.49 miles. This means our model accurately guessed the correct geocell 2 out of 5 times based on our Top-1. Our model, even when incorrect, predicts one to two adjacent cells consistently, which is supported by the Top-5 Accuracy.

Based on our confusion matrix 5.5, we can see significant improvement and accuracy rates around the top left and the bottom right corners. This is expected and shows that our model correctly identifies iconic areas around the coastlines, and our model struggles more with the geocells in the middle. This makes sense as the geocells in zones 10-15 show homogeneous patterns and are difficult to distinguish from one another. For example, areas like New Mexico may appear the same as Nevada. Another example is the similarity of suburbs in the United States.

As seen in the first GradCam image 5.6 the model fixated heavily on the shape and color of the grass and trees both up close and in the distance. This is not surprising since the environmental factors were something that we expected to be a large portion of classifying images correctly. In the second GradCam image 5.7 we once again see more activation around the bushes and trees but interestingly it activates around the shadow of the Google car, meaning it might have used the shadow to figure out its location. We searched for other images where the shadow was visible but did not find any others where this was the case in the few hundred images we searched through. Finally, in GradCam image 5.8 we can see that the grass next to the road has heavy activity, reinforcing our idea that things that vary between environments will have a large impact on the model classifying the images.

5.6 Conclusion

Our model ended with a Top-1 Accuracy of 42% and achieved a mean distance of 132.49 miles. While that number may initially seem low, remember there are 28 unique geocell options. Supplementing our geocells with ecological data proved to be fruitful, and improved our model's accuracy from the previous milestone. If we had more time, we could have further broken down ecological zones to refine our model.

However, overall we believe the model achieved our initial goal of accurately predicting a location given a GeoGuessr image. We acknowledge that the learning problem was difficult but we are satisfied with our outcome. If we were to continue working on this project with an indefinite timeline, we would train models with full panoramas instead of a limited 90-degree FOV photo. By doing this, our model can learn more and analyze details that may determine the location. It may also help prevent overfitting and give us a more generalized model of the United States. Another improvement is to expand our project's

scope to all of North America and potentially globally. Finally, we would hope to create a Chrome extension so we may utilize our bot within the game.

Table 5.4: Contributions by team member for Milestone 5.

Team Member	**Contribution**
Dakota Andrews	Worked on the writeup, and trained the model.
Gage Cammack	Organized the data, trained the models, and created visual graphs.
Joseph Seibel	Created the models, organized the data, and trained the models.
Tong Wu	Majority document write-up and organization.

Bibliography

[1] Zain Ul Abideen. *How OpenAI's DALL-E works?* accessed: 2024-02-12. URL: https://medium.com/@zaiinn440/how-openais-dall-e-works-da24ac6c12fa#:~:text=To%20learn%20the%20prior%20distribution,vectors%20in%20an%20autoregressive%20fashion..

[2] Amazon. *What is GAN?* accessed: 2024-02-09. URL: https://aws.amazon.com/what-is/gan/#:~:text=A%20generative%20adversarial%20network%20(GAN,from%20a%20database%20of%20songs..

[3] b2studios. *Beating BTD6 with AI*. accessed: 2024-02-10. URL: https://www.youtube.com/watch?v=QoEMoSbGvbM.

[4] Eirs. *ArcGIS Pro*. accessed: 2024-02-19. URL: https://pro.arcgis.com/en/pro-app/latest/get-started/download-arcgis-pro.htm.

[5] Google. *Google Cloud Documentation*. accessed: 2024-03-27. URL: https://cloud.google.com/docs/.

[6] Google. *Google Street View static API*. 2024. URL: https://developers.google.com/maps/documentation/streetview.

[7] Lukas Haas et al. *PIGEON: Predicting Image Geolocations*. 2023. arXiv: 2307.05845 [cs.CV].

[8] M. J. Kearns. "Computational Complexity of Machine Learning". PhD thesis. Department of Computer Science, Harvard University, 1989.

[9] Keras. *MobileNet, MobileNetV2, and MobileNetV3*. accessed: 2024-03-03. URL: https://keras.io/api/applications/mobilenet/.

[10] Author links open overlay panelAmmar Mohammed and AbstractIn machine learning. *A comprehensive review on Ensemble Deep Learning: Opportunities and challenges*. Feb. 2023. URL: https://www.sciencedirect.com/science/article/pii/S1319157823000228#:~:text=Hence%2C%20ensemble%20deep%20learning%20methods,Mohammed%20and%20Kora%2C%202021)..

[11] OpenAI. *Image Generation*. accessed: 2024-02-09. 2018. URL: https://platform.openai.com/docs/guides/images/image-generation?context=node.

[12] Stelath. *geoguessr-ai*. Jan. 2024. URL: https://github.com/Stelath/geoguessr-ai.

[13] unknown. *GeoGuessr AI: Image Based Geo-Location*. 2020. URL: https://nirvan66.github.io/geoguessr.html.