CIS 4190/5190: Applied Machine Learning Fall 2024

ViTs for Visual Place Recogition

Team Members: Samuel Wang, Franci Branda-Chen, Seth Sukboontip *Project: Img2GPS*

1 Summary

In this project, we trained machine learning models for visual place recognition. Specifically, we collected data points in the Penn Engineering Quadrangle and trained models to predict the latitude and longitude of the images from only visual features. We tested several model architectures, such as ResNets, ViTs, and VLMs. Moreover, we used data augmentation and semantic segmentation techniques to improved the model's performance and robustness. We achieved an error of 34.6 meters on the held-out test set.

2 Core Components

2.1 Data Collection and Dataset

Each of our team members took charge of the data collection at different times of day and weather conditions from overcast to sunny days. Furthermore, we received information that all of them were at eye level and only on the walkways. This prior knowledge greatly helped with the data collection; we took around eight photos of our surroundings with each step through the engineering quadrangle. Moreover, after the first round of image cleaning, we realized it would be easier if we were to take all of the photos in a native square ratio straight from our phones rather than cropping in the post-processing step. This way, we could focus on the identifiable objects in the frame and receive real-time feedback on what the model will be analyzing.

In terms of processing the data, we utilized the given Google Colab file to easily turn our images into huggingface datasets; we cropped the images into a one-to-one ratio, reduced the resolution to 224x224 pixels, and extracted the GPS metadata. We collected 3,600 images across several days and covered the entire region of the engineering quadrangle. A visualization of the train distribution is plotted alongside the provided validation set in the appendix.

2.2 Model Design

To solve this problem, we need a supervised learning model that takes in images and outputs the predicted GPS location. In our case, the "feature variables" are the images and the labels would be the GPS location. In this project, we tested different models to beat the provided ResNet baseline.

The linear regression model is the most basic form of supervised learning. As expected, the model performed extremely poorly, overfitting the training data and failing to extend to the validation images. Even with a lowered learning rate, L2 regularization, and more aggressive data augmentation, the model was still performing worse than the baseline RMSE. Another hypothesis we had was that the model was overfitting due to the high dimensional nature of the input data — the samples contained over 150,000 features. To combat this issue, we tried employing PCA to reduce the high dimension of the data; we lowered the number of variables to 200 features. However, after training with the modified data processing method, the linear regression model performed worse than all of the previous attempts.

Hypothetically, there may be a "sweet spot" in terms of dimensionality, but we decided to transition to a more complex model family. We believe the input data and complexity of the problem transcends a simple linear relationship. This matches what we were taught in class, as linear regression fails to capture the translation invariance and equivariance needed to properly analyze images. We will expand on these design choices in our Exploratory Questions section.

After some research, we decided to focus on the Vision Transformer (ViT) model [**?**], which builds upon the transformer model from the Natural Language Processing lecture. The model adapts the architecture for images instead of text processing by splitting each image into small patches. Moreover, to maintain the spatial information of the image, we also pass in the position encodings of each patch, signifying their relative distances. Furthermore, ViTs rely on the same transformer model with self-attention layers.

ViTs provide the benefits of understanding the global context of the image through the sequences and relationships between the input patches rather than isolated areas with convolution filters. However, the model inherently requires a lot of training data. We used a training dataset of over three thousand images; it is possible for the model to perform even better with more sample data. Running the ViT model on our training dataset resulted in promising results. However, the model quickly became computationally expensive, a downside of the model family. To alleviate this, we use open-sourced weights for the model that were trained on ImageNet 21k [**?**]. We then do transfer learning by attaching a two-head fully connected network to the end of the ViT.

2.3 Evaluation and Model Performance

We used Mean Squared Error (MSE) as the loss function. The model tried to minimize the MSE value between the true and predicted GPS locations. Since the GPS location consists of the latitude and longitude values, the MSE equation becomes

$$MSE = \frac{1}{2}((lat_{true} - lat_{pred})^2 + (long_{true} - long_{pred})^2)$$

After training the model, we also evaluated its performance via geodesic distance during the validation phase. The distance value is simply the distance in meters between the true and predicted GPS locations as the error value. Moreover, we can apply RMSE to these error values such that $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(distance_i)^2}$ to calculate the distance in meters.

3 Exploratory Questions

3.1 In-context Learning for Foundation Models

3.1.1 Question and motivation

In this exploratory direction, we explored if foundation models can be used to predict locations directly from the image. We chose to run our experiments with the Gemini Vision Language Model because it offered a long context window and a free tier of API usage.

3.1.2 Prior work

Specifically, we sought to leverage the ability of VLMs to do in-context learning — where given a few examples, the VLM can achieve better performance on tasks that are farther out of its distribution by having examples of correctly executed tasks in its long context windows. This is similar to transfer learning, except rather than needing resources to fine-tune the model, in-context offers the promise of quick performance gains in foundation models. In-context learning has been shown to work for tasks in a wide range of domains. For example, in recent works, with in-context learning, VLMs have been shown to have the ability to generate value functions for unlabeled robot videos [**?**] to drug binding prediction from molecular images [**?**].

We hypothesized that while the model would have some success, it would not ultimately be the best model due to the specific task of identifying the nuances between scenery in a small region. Additionally, while we were excited about Gemini's ability to recognize objects in a scene, we had concerns about its performance due to its struggles with spatial reasoning and numerical understanding.

3.1.3 Methods of investigation

To measure the performance of the VLM, we evaluated a few prompting techniques. Across all prompts, we used K-Means to cluster the images from the training dataset into K clusters. We also instructed the VLM to pay attention to details such as landmarks and the types of materials used in the buildings. We then used the centroids of each of these clusters as the in-context examples in the prompt to ensure that a diverse set of locations was in the context of the VLM. To assess the VLM's ability to generalize, it was always run on the given 100-image validation set.

To verify that the VLM could even do this task, we first began with a basic test where it was given the validation set as its in-context examples and asked to predict the locations of the same data points. Despite being fed the exact dataset it was tested on, this preliminary iteration had an error rate of 10 meters.

3.1.4 Results

Ultimately, we found that across the different prompts we tried, which varied the number of in-context examples (20, 50, 100), and the output format (normalized, unnormalized), we found the best-performing prompt had 100 in-context examples and normalized the latitude and longitude. However this pipeline achieved roughly the same performance as the baseline of predicting the mean of the training dataset with an error rate of 102 meters, while taking around 5 seconds per image.

Overall, this method performed somewhat worse than we expected; we predicted it would at least beat the baseline of directly predicting the mean. However, it is still impressive that the VLM was able to do this task with little direct training on it. A further study would test other VLMs and potentially involve a multi-step pipeline that uses the VLM to break down the scene into constituent objects.

3.2 Object-Centric Location Prediction with Segmentation

3.2.1 Question and motivation

In this exploratory direction, we explored if segmentation models can be used to improve the performance of the models by removing distractor objects. We expected this to improve the model performance because some of the objects in the images were not relevant to classifying the image. If there was a red car around the outside of Levine when we took the photos, it may be in a large proportion of the photos, leading to spurious correlations when the model runs inference. This direction also maps better to what we do intuitively as humans — we identify the location by looking at landmarks in the scene such as buildings and other details.

3.2.2 Prior work

Some works in the visual place recognition literature had tried similar techniques and were successful as discussed in this survey on deep visual place recognition [?]. However, it is worth noting that these datasets, like CMU CityScapes, were much more focused on classifying scenes from across a city, rather than identifying a specific location in a city block, so we were unsure if the same findings would transfer.

3.2.3 Methods for investigation

To test integrating segmentation models into the training pipeline, we planned to compare the performance of the ViT architecture from the previous section with and without segmentation masks. Additionally, we used post-hoc explanation methods to analyze where the model fixated on to make its predictions to see if segmentation models would help focus it on task-relevant regions. Specifically, we used EigenCam [?] for this project, which projects the activation model of the map to 1 dimension using PCA to extract the most salient regions in the image. We initially sought to use the Segment Anything Model [?], specifically prompting it using bounding boxes from Grounding DINO, as done in the Grounded SAM pipeline. However, we ultimately found that while this pipeline gave us very high-quality segmentations, it took over 5 seconds to run inference for a single image, making it impractical to train a model for sufficiently long on our limited computational budget. Thus, we transitioned to using a panoptic segmentation model, specifically, NVIDIA's SegFormer [?] that was finetuned on CityScapes [?]. This version had much quicker inference, while still providing labels to differentiate between the different semantic labels for each of the masks.

3.2.4 Results and updated beliefs

We tested training the segmentation-based model with and without filters. Without filters, it performed better than the ViT baseline. However, when training with filters, it had higher validation error.

Ultimately, the masks from this pipeline were not as fine-grained as something like SAM. For example, if a label like *Foliage* was filtered out, then the buildings would be removed from many images too where trees are in front, which meant there were less removable irrelevant objects than expected. However, it was good to see that the segmentation model did lead to improved grad-cams as expected. The model focused more on task-relevant objects like buildings and learned to ignore cars. Visualizations are in the appendix.

In the future, we could work on tuning the architecture of the model to become more interpretable, rather than changing the input and using post hoc explanations. Rudin et al. have laid out a convincing argument that post-hoc explanations like GradCams and EigenAms are questionable as they can also be used to explain conclusions that don't make sense. [**?**]. For example, one direction we are interested in is a recently introduced "20-questions" style pipeline that automatically generates a set of questions answered by a VQA model to interpretably classify the image [**?**].

3.3 Photo Filters (Color Space Augmentation)

3.3.1 Question and motivation

Do color space transformations for data augmentation improve the performance of the model (ViT)? This question was relevant to our project due to our dataset consists of thousands of images, but we were not able to capture every instance of the day. The idea was to train the model with photos with different types of lighting to simulate different times of the day and weather conditions.

3.3.2 Prior work

According to this paper [**?**] on image data augmentation, augmentations in color channels were practical to implement and one of the earliest effective methods for data augmentation. This improvement stemmed from the fact that RGB values of an image can be easily manipulated to change the brightness of the image. However, this method did not come without its cost; it increased the memory used and the training time significantly. Despite these drawbacks, the expected outcome was that the filters (color space enhancement) would, in fact, improve the model's training to make more accurate coordinate predictions.

3.3.3 Methods for investigation

We used 3 different filters to simulate different times of day and weather conditions.

- Morning filter: We set the brightness between 0.7 - 1.0 of the original photo, which simulated early morning lighting. Contrast was set between 0.8 - 1.0 to exaggerate the difference between light and dark regions of the photo, so we could see shadows in the photo more clearly. Saturation was set between 0.8 - 1.0 to increase the intensity of the photos' color to further simulate early morning photos (in case the photos were taken in the late afternoon). The hue was set between -0.1 - 0.0 to shift the color wavelength towards the red/orange colors because mornings tend to have those colors due to the sunrise.

- Afternoon filter: We set the brightness between 0.8 and 1.2 which simulated afternoon lighting when the sun is past noon and bright. Contrast was set between 0.8 - 1.1, for shadows were generally darker in the afternoon, unlike in the morning. Saturation was set between 0.9 and 1.1 to increase the intensity of the photos' color to be slightly more saturated than the early morning filter. Hue was set between -0.05 - 0.05 to simulate the afternoon type of lighting.

- Sunny filter: We set the brightness between 1.1 and 1.3 to simulate a bright and sunny day, making a rainy and cloudy day look sunny after it finished raining. Contrast was set between 1.0 - 1.2 to increase the contrast between shaded and bright areas. Saturation was set between 1.0 and 1.2 to increase the intensity of the photos' color to make it more vivid. Hue was set between -0.05 - 0.05 to simulate the sunny type of lighting.

The experiment consisted of using all the original dataset photos as the control group of the training, and gradually increasing the amount of photos per filter (photos were chosen randomly). At each attempt, we were starting a fresh new ViT model that was pre-trained from 'google/vit-base-patch16-224-in21k'.

3.3.4 Results and updated beliefs - Refer to Appendix A for results

Based on the experiment (Attempts 1-3), we used our first dataset consisting of 1427 photos. As we increased the batch size for each filter, we can observe that the validation loss decreased. That is, as more photos with filters were added, the model would predict closer to the actual coordinates of the validation dataset.

Our second and most updated dataset consisted of 3638 photos. Once again, we started a new fresh ViT model from 'google/vit-base-patch16-224-in21k.' With our new dataset (consisting of the old 1427 photos and 2211 new photos), we can observe that the validation loss dropped dramatically [Attempt 4]. We must note that our training dataset size was n=5243, compared to n=5703 for the previous third attempt, so the new training dataset we used was almost 500 samples less. We could also see that we used 535 photos, which is less than both of the filter photo quantities on the 2nd and 3rd attempts. However, despite having fewer training images with and without filters, we outperformed significantly our previous two attempts. It must be noted that our 2nd dataset included new photos that were more closely aligned to the photos in the validation dataset provided by the TAs, which can explain the significant drop in loss.

For our last attempt (Attempt 5), we saved the previous attempt's model and ran it for five more epochs but used a smaller training dataset. As we can see, our model improved significantly. This time we used the same amount of original photos as the number of filtered images. That way, we can attribute the new validation loss drop more heavily to the photos with filters, rather than the original photos.

Based on our results, we can see the pattern that as more photos with filters were added, the validation loss dropped significantly. This told us that adding filters did, in fact, help improve the prediction accuracy. Even from attempts 1-3 (before we increased the dataset), we can observe the same pattern. Thus, the implementations of color filters matched our expectations.

3.3.5 Limitations

- Training the model took longer for each attempt. Simple mistakes in syntax had a high cost in model progression since we had to restart the whole training process.
- Loading the dataset would also lead to valuable time loss due to the significantly higher number of photos.

Additional steps: Given more time, we would train the model solely with original photos without filters, to fully evaluate the impacts of the filters. Also, we would take photos on the side of the street at different distances from the building's wall to create an even higher variance in training data.

4 Team Contributions

Franci Branda-Chen: Taking images, designing data augmentations to simulate different conditions, and writing the report.
Seth Sukboontip: Taking images, implementing & training baseline models (linear regression, PCA, ResNet, and ViT), and writing the report.
Sam Wang: Designing exploratory questions, foundation model experiments, segmentation model & explainability experiments, and writing the report.

HuggingFace: https://huggingface.co/Latitude-Attitude/vit-gps-coordinates-predictor-with-filter

5 Appendix A - Results for Section 3.3 Photo Filters (Color Space Augmentation)

Attempt 1

[1427 original photos + 350 of each filter], n = 2477

Training Loss

Epoch	**Training Loss**
1	0.3741
2	0.0430
3	0.0131
4	0.0049
5	0.0033
Validation Loss	0.7937

Table 1: Attempt 1

Validation Batch Results

Batch	**Avg Distance (m)**
1	39.51
2	55.32
3	66.74
4	66.09
Validation Loss (m)	66.09

Table 2: Attempt 1

Attempt 2

[1427 original photos + 600 of each filter], n = 3227

Training Loss

Epoch	**Training Loss**
1	0.3437
2	0.0448
3	0.0169
4	0.0092
5	0.0074
Validation Loss	0.7448

Table 3: Attempt 2

Validation Batch Results

Batch	**Avg Distance (m)**
1	43.80
2	57.17
3	63.63
4	63.55
Validation Loss (m)	63.55

Table 4: Attempt 2

Attempt 3

[1427 original photos + 1427 of each filter], n = 5708

Training Loss

Epoch	**Training Loss**
1	0.2191
2	0.0153
3	0.0073
4	0.0063
5	0.0087
Validation Loss	0.6567

Table 5: Attempt 3

Validation Batch Results

Batch	**Avg Distance (m)**
1	43.54
2	57.18
3	59.21
4	57.81
Validation Loss (m)	57.81

Table 6: Attempt 3

Attempt 4

[3638 original photos + 535 of each filter], n = 5243

Training Loss

Epoch	**Training Loss**
1	0.2635
2	0.0334
3	0.0121
4	0.0070
5	0.0056
Validation Loss	0.3623

Table 7: Attempt 4

Validation Batch Results

Batch	**Avg Distance (m)**
1	33.99
2	39.38
3	41.96
4	41.37
Validation Loss (m)	41.37

Table 8: Attempt 4

Attempt 5

[535 original photos + 535 of each filter], n = 2140

Training Loss

Epoch	**Training Loss**
1	0.0891
2	0.0510
3	0.0193
4	0.0094
5	0.0076
Validation Loss	0.3051

Table 9: Attempt 5

Validation Batch Results

Batch	**Avg Distance (m)**
1	32.14
2	38.83
3	36.11
4	35.56
Validation Loss (m)	35.56

Table 10: Attempt 5

6 Appendix B - Visualizations for Segmentation and EigenCAMs

segmentation.png

7 Appendix C - Dataset Distribution

Distribution.png

References