Image Geolocation via Adaptive Partitioning with ConvNeXt and Vision Transformers

Branislav Kesic
branisk@gatech.edu

Louis Bellosguardo
lbellosguardo3@gatech.edu

Matthew Freeman
mfreeman74@gatech.edu

Nicolas Gonzalez
ngonzalez62@gatech.edu

Abstract

Image geolocation aims to determine the real-world coordinates about the origins of a photograph. The task has gained wider attention through the game "GeoGuessr," which challenges players to identify the location of street-view images from around the world. It also has practical uses, including GPS augmentation, identifying the origin of illicit content on livestreams and social media, and organizing photos from travel destinations. We focus on a constrained version of this problem: predicting the location of an image within a 2 km by 2 km area centered on Washington, D.C. Our dataset contains roughly 200,000 images. We frame the task as a classification problem by dividing the region into 144 cells using an adaptive partitioning algorithm. We evaluated several neural network architectures including Vision Transformers (ViT) and modern Convolutional Neural Networks (CNN) such as ResNet and ConvNeXt. Our best model reaches an accuracy of more than 86 percent and an average error of 57 meters from the true coordinates.

1. Introduction

1.1. Objective and motivation

"GeoGuessr" is an online game which drops players into random Google Street View locations and asks them to mark where they think they are on a world map. Inspired by the game and computer vision models covered in CS-7643, we set out to build a system that could geolocate images taken anywhere on Earth. As we explain in this paper, limits on compute and dataset availability led us to narrow the task to a 2 km by 2 km region in Washington, D.C. We will present existing approaches and additional context to the motivation behind this decision.

1.2. Existing approaches

Prior to the rise of deep learning, most geolocation systems relied on retrieval-based approaches. A seminal example is *Im2GPS* [4] which extracted handcrafted features such as color histograms and texture descriptors, and then matched a query image to its nearest neighbors in a large reference database. The predicted location was derived directly from those neighbors, making the task essentially a feature-engineering and KNN-style retrieval problem.

The first major shift toward modern approaches came with *PlaNet* [9], which introduced an adaptive partitioning scheme to divide the Earth into uneven but approximately balanced geographic cells, and trained a convolutional neural network to classify an input image into one of these cells. This formulation allowed the model to learn location-relevant cues directly from data rather than relying on handcrafted descriptors.

The current state of the art, *PIGEON* [3], extends this idea into a multi-stage pipeline: it begins with coarse geocell classification similar to PlaNet, then refines its initial prediction by comparing the query image's embedding to densely clustered embeddings within the top-scoring cells. This hierarchical combination of classification and fine-grained retrieval achieves significantly higher resolution than either component alone and represents the dominant paradigm for contemporary geolocation systems.

1.3. Impact

Image geolocation has several potential impacts beyond academic interest. Law enforcement agencies increasingly encounter criminal activity shared through images or videos online, and automated geolocation tools could help authorities identify where such content was created. A successful model may also have commercial applications: social media posts often omit location tags, and users may be curious about where a particular photo was taken. Geolocation also plays a role in recreational settings, as demonstrated by the popularity of GeoGuessr. In the same way that Deep Blue

famously beat Garry Kasparov at chess, and AlphaGo beat Lee Sedol at Go, an image geolocation model could set a baseline for Geoguessr. Finally, vision-based geolocation could complement traditional GPS systems, providing additional localization cues in signal-poor or congested environments by inferring position directly from images of the surrounding area.

1.4. Selected Dataset

We use the *Global Streetscapes* dataset [6], a large-scale street-level imagery collection designed to support urban analytics. We initially chose it for its global coverage and precise geographic labels, which suited our early goal of building a model capable of worldwide geolocation. As our scope narrowed to city classification and eventually to fine-grained localization within a single city, the dataset remained flexible enough to support each stage.

Motivation Global Streetscapes was created to provide a free, large-scale dataset enabling urban science research across many cities worldwide.

Composition It contains 10 million images from 688 cities, sampled from a 2 km $\times$ 2 km area at each city's center. Each image includes over 300 derived attributes in addition to latitude–longitude coordinates.

Collection Process Imagery was sourced from crowdsourced street-view platforms such as Mapillary and KartaView. Cities were selected to ensure broad geographic and demographic coverage (population above 50k; inclusion of national capitals).

Preprocessing and Labeling The dataset creators enriched imagery with 300+ metadata attributes using geospatial operations and computer vision methods.

Distribution The dataset is openly released, with metadata hosted on Hugging Face and accompanying open-source code provided by the authors.

2. Approach

2.1. Problem description

Our original aim was to develop a deep learning model capable of predicting any location on Earth. To support this objective, we selected a globally comprehensive dataset [6].

We initially considered defining the task as a straightforward regression problem with two outputs: latitude and longitude. However, upon investigation we found that other

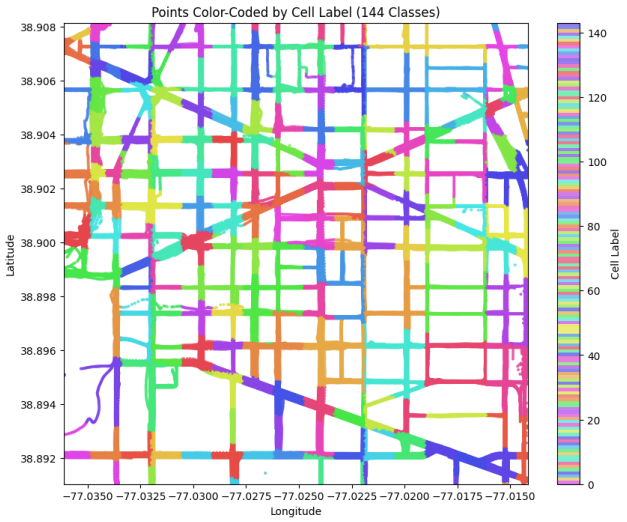

Figure 1. Partition of Washington DC into 144 distinct classes.

works[7][9] had to pivot to classification due to uneven spatial coverage in available datasets (few images in the middle of oceans and deserts, and high density of samples in highly populated regions of the earth). Furthermore, the wrap-around nature of longitudes makes it difficult for regression to understand that longitudes of -179.9 and +179.9 are right next to each other.

These limitations motivated a shift toward classification-based methods. A naive uniform grid over the Earth yields extreme class imbalance, so we surveyed the related literature and identified PlaNet [9] as a promising direction. Its adaptive partitioning algorithm recursively subdivides the globe to produce a more balanced set of geographic cells. Further details can be found in the paper.

We soon realized a few problems with our global ambition. Firstly, computational constraints. The full dataset, with its 10 million images, would have required over 5TB of storage - more than any of our team members had available. More problematic still was the amount of compute that training a model on this much data would have necessitated.

To scale the problem down, we considered picking a subset of cities in our dataset and working on a city-classification problem. However, after achieving 80% on an initial run, we decided that we wanted to go in a more challenging direction, and also wanted to build a model with more real-world value. This is how we arrived on what would become of final problem: image geolocation within a specific city. Based on what our dataset had available, this became image geolocation within the 2km by 2km downtown area of Washington DC.

Using the 207,103 available images within this area, we applied PlaNet's [9] adaptive partitioning algorithm to get 144 semi-balanced cells. Starting with a 6-cell division of

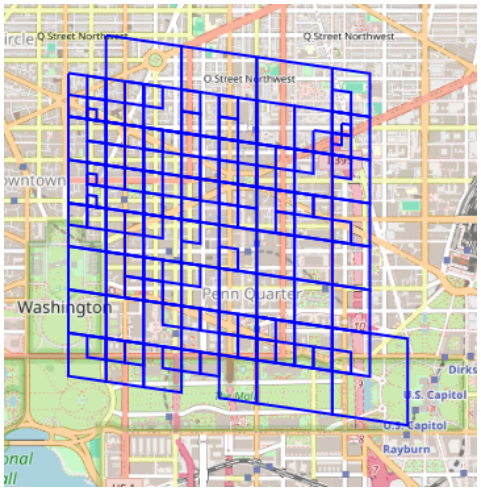

Figure 2. Adaptive partitioning cells overlayed on Washington DC map.

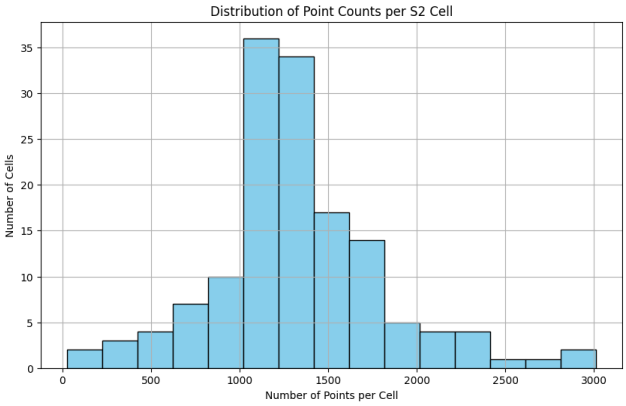

Figure 3. Count of values for each cell after performing adaptive partitioning on our dataset

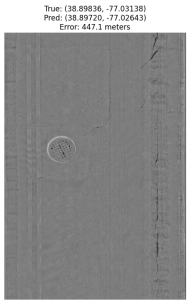

Figure 4. Sample image from our dataset, demonstrating the issues with data quality.

the earth, the algorithm repeatedly subdivides each of those cells into 4, traversing a maximum of 25 ”levels”. The algorithm discards cells with fewer than 1.000 images, and further divides cells with more than 10,000 images. This approach yielded the cells visualized in Figure 1 and Figure 2. As shown in Figure 3, the sample images are pretty well balanced, with most classes having approximately 1200 images in them.

Having set up this classification problem, we downloaded the weights of a vision transformer model (ViT-B/16) and used pytorch to pretrain it with ImageNet-1k. We then replaced the head of the pretrained model with a linear layer using the input size of the original head, and replaced the output dimensions with the number of classes. Finally, we fine-tuned the model on our dataset for 5 epochs using cross-entropy loss and observed 40% test accuracy on 144 classes after 5 epochs. We took this as a clear signal that the model was learning correctly, and gave us confidence that our selected problem had potential. As we will discuss in the Experiments and Results section, we spent the rest of our time trying different approaches at improving our test accuracy. Our approach ranged from tuning hyper-parameters, to trying different base model sizes and architectures (ResNet50[5], ViT[1], and ConvNeXt[8]), to changing the number of classes and even revisiting the idea of regression. Something novel which we attempted was to mix regression and classification, as we will discuss in more detail in later sections.

2.2. Problems anticipated and encountered

As discussed in the previous section, one of the challenges we faced, which ultimately led us to scale down our research problem, was the problem of data size and required compute resource constraints. Beyond scaling down the problem as discussed above, another approach that we had to resort to was using online compute and storage. We spent time setting up RunPod - an online compute platform. This allowed us to rent GPUs of varying capacities by the minute, and ultimately allowed us to tune our hyperparameters - something that would have been impossible using our own hardware.

Another challenge that we faced was data quality. It was impossible for us to look at all 207,103 images in our dataset. However, it only took looking at a few dozen to come across multiple poor quality samples like the one pictured in Figure 4. Considering that the dataset is crowdsourced, this is not surprising. We decided to rely on the resilient nature of deep learning, capable of finding effective weights despite some erroneous samples in the dataset. Indeed, it is clear that our model learned effective representations regardless of the observed issues with data quality. However, we suspect that samples such as Figure 4, may have made it difficult for our model to achieve even better performance than it did.

3. Experiments and Results

Having set up the problem and proved its viability, we experimented with a variety of approaches to achieve higher

levels of accuracy and minimize the Euclidean distance between predicted values and their ground truth label. In the following sections, we go over further details on experimentation with various model architectures, hyperparameter decisions, and exploration between regression vs classification and their effects on model output. All models were loaded with pre-trained weights which can be found in B.

3.1. Hyperparameter Tuning

Taking our baseline performance of 40% with the ViT-B/16 model, we modify the hyperparameters for model training in search of meaningful performance improvements. Starting with the number of epochs, which alone plays a large role in the large performance jump. After 5 epochs, the base model's learning curves were not yet showing any signs of convergence, indicating that there were still many features and patterns to be learned by the model given more time. With 152,000 samples in the training set, 144 classes, and 86.57 million trainable parameters, 5 epochs is insufficient for the model to adequately learn, resulting in an under-trained model which showed fair but non-optimal generalization capabilities. By increasing training epochs to 20, we afford the model more iterations to continue learning, and observe as a result the learning curves flattening at epochs 13-15. The feature representations used to geolocate images become better fleshed out with later epochs, as class-specific patterns are uncovered by the model.

We next look at the learning rate, which we lowered from the base value of 1e-3 down to 1e-4. The higher base rate was too aggressive to effectively locate minima within the loss landscape, a problem amplified by the low epoch count discussed earlier. Given a higher epoch count, we have the ability to lower the rate to a more conservative value, resulting in fine-grained updates to the model's weights following each iteration. Useful representations of certain features are more likely to be preserved as training furthers, while making more precise adjustments to avoid overshooting newly discovered local minima. This last point is aided by the implementation of a OneCycleLR scheduler with a 10% warm up, which was absent from the baseline model. The use of this scheduler is beneficial when using a pretrained model as we did, as it prevents strongly-weighted updates early which would destabilize the model, and allows the pretrained rates to gradually update based on the learnings of the model during the training process. The scheduler then allows the learning rate to reach the max learning rate of 1e-4, before "annealing" the learning rate down to enable the convergence of the model once it has learned sufficient features.

An additional change was the use of label smoothing with a value of 0.1. Using label smoothing helps the model avoid overconfidence in its estimates, particularly when many of the classes may be similar. This is particularly useful in our case, as images from neighboring zones may have shared features or landmarks visible in both instances, and it would not make sense to penalize a mistake here as heavily as predicting a class on the other side of the map. By lowering such penalties for similar classes, we reduce the rigidity of the decision boundary while also avoiding significant dilution of any signals with a relatively low smoothing value of 0.1.

Model	Accuracy	CE Loss	L2 Norm
ResNet-50	80.20%	1.598	99.22
ViT-B/16	83.79%	1.442	78.706
ViT-B/32	77.67%	1.676	119.659
ViT-L/16	82.37%	1.586	89.183
ViT-L/32	78.26%	1.724	117.123
ConvNeXt_base	84.64%	**1.377**	66.860
ConvNeXt_large	**86.52%**	1.381	**56.971**
Small-Data Regime			
ViT-B/16	61.52%	2.392	215.882
ConvNeXt_large	**69.38%**	**2.020**	**150.183**

Table 1. Model results. L2 Norm is the average euclidean distance in meters between the center coordinate of the predicted cells and their corresponding targets. Each model was trained with 144 classes and identical hyperparameters over 20 epochs. The *Small-Data Regime* results are for models trained on only 25 percent of the total data.

3.2. Model architectures

The ViT-B/16 model achieved strong performance due to its use of self-attention, allowing the model to identify features from the foreground and background and combine patches together to determine a classification. Each patch is of size 16x16, which strikes a balance between fine-level details in an image without being overly sensitive to noisy pixels, and contains 86.5 million parameters available for feature representation. In order to better learn spatial priors, the transformer architecture could benefit from having more data than our dataset size, which may have been a limiting factor for generalization relative to its peers.

Building on the successes of tuning the ViT-B/16 model, we contrast its performance to two competing CNN models, Resnet-50, and ConvNeXt. The former dates from the pre-transformer era, and is touted for its excellent image classification abilities [5]. The Resnet-50 architecture involves the use of 3x3 kernels, 50 layers, and about 25 million parameters. Though its pre-trained ImageNet weights helped the Resnet-50 model to achieve performance metrics similar to the other transformer-based models, some architectural limitations including compression of feature dimensions, smaller kernel size, and texture biases (such as classifying based on building or road textures rather than geographic layout) inherent to the model may have caused

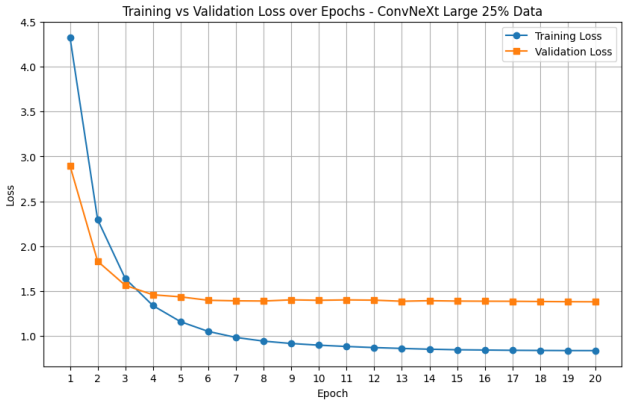

Figure 5. Learning Curve for ConvNeXt Large Model.

performance to lag behind other models.

The ConvNeXt model is a CNN designed to mimic the Transformer architecture, but using convolutions rather than attention to do so. Instead of 3x3 kernels as with Resnet-50, it uses 7x7 kernels instead, which enables it to view a larger context window [8]. It is able to benefit from the locality bias of CNNs while also taking advantage of large receptive fields, which allow the model to better learn and understand different geographic layouts. The ConvNeXt_base model we used updates 88.6 million parameters, which is even more than ViT-B/16, and offers larger representational capability to distinguish between the high number of classes in our problem. Results can be found in Table 1.

Across the board, ConvNeXt outperforms the traditional Resnet architecture and the ViT architecture, confirming the results from [8]. We also confirmed small-data regime results from [1] by using 25 percent of the total dataset. These results can be found in 1 below the row labeled *Small-Data Regime*. For completeness we include the learning curve for the ConvNeXt-Large model in Figure 5. The learning curves for all models had similar characteristics.

In Figure 6 and Figure 7, we visualize the gradients of the image inputs with respect to the loss between both ViT and ConvNeXt. We can see the potential limitations of the image patches utilized by ViT, as ConvNeXt has the ability to capture more fine-grained features with individualized representations. We see that ViT tends to emphasize the importance of vegetation in general, where ConvNeXt focuses on structures adjacent to the streets. Both models tend to omit the sky as well as dynamic objects such as cars.

3.3. Regression vs Classification

In an earlier section of this paper, we discussed the fact that we chose to pursue classification instead of regression due to regression's likely struggle with large swaths of earth with no data points, as well as the wrap-around nature of longitudes. However, that was when we were thinking of geolocating images across the entire earth. After scaling down the problem to a 2km by 2km area with high density of samples, we realized that those issues no longer played a big factor. Furthermore, we saw regression as having the potential to alleviate the following issue: we were concerned that cross entropy loss does not distinguish between a predicted cell next to the correct one, and a prediction that is totally off. We feared that classification with cross entropy loss may be unreasonably harsh on samples drawn from the boundary between two cells.

Alpha	Accuracy	L2 Norm (CLS)	L2 Norm (REG)
1	**80.26%**	**88.258**	**141.604**
5	79.09%	90.714	126.469
10	73.85%	107.423	146.901

Table 2. Results for multi-task regression and classification trained with identical hyperparameters and 144 classes. Alpha represents the value multiplied to the regression loss. L2 Norm is the average euclidean distance in meters between either the center coordinate of the predicted cells or the regression prediction and their corresponding targets.

With this idea in mind, we replaced the linear classification layer attached at the end of the pretrained model with a small regression head consisting of two fully connected layers that directly predicted a latitude and a longitude that predicted a latitude and a longitude.

In order to compare the performance of the regression model with the prior classification model, we introduced a new metric: average Euclidean distance between prediction and ground truth. For the classification setting, we approximated this as the Euclidean distance between the center of the predicted and ground truth cell. While our best classification model (at this point) had achieved average eucledian distance of 89m, the equivalent regression model plateaued out at about 120m. We attribute this to the regression model's difficulty in learning a stable, globally consistent mapping from images to continuous coordinates. Classification benefits from discrete cell boundaries that provide a strong inductive prior, while regression must fit a highly non-linear function in which similar-looking scenes can correspond to very different locations. As a result, the regression head tended to overfit and generalize poorly, leading to higher positional error. This gap motivated our later exploration of hybrid models that combine the strengths of both approaches.

3.4. Multi-tasking

In an attempt to further refine results, we also experimented with a combined multitasking approach similar to the methods introduced by Fast R-CNN[2]. Fast R-CNN realized improvements ranging from +0.8 to +1.1 mAP points using the multi-task training approach, providing a promising route for improvements within our models. Although

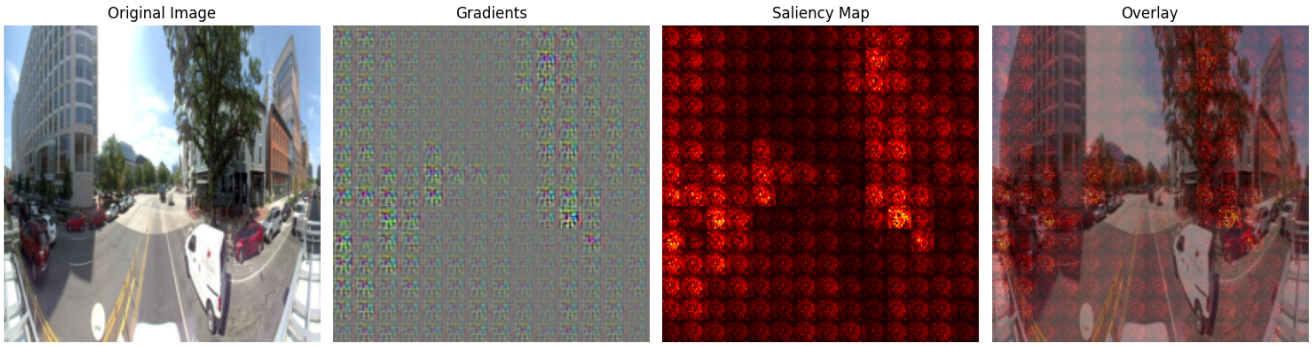

Figure 6. Saliency map for ViT-16/B

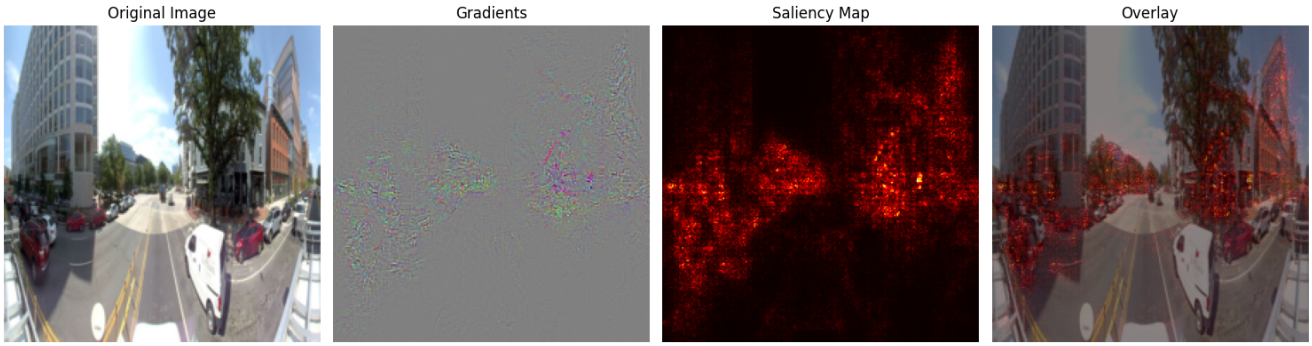

Figure 7. Saliency map for ConvNeXt Large

Fast R-CNN applies its results to object detection, we figured it might be a worthwhile route to explore for our distinct computer vision task.

We implemented the multi-tasking approach by replacing the previous class prediction with a multi-tasking head consisting of 2 fully connected layers. The first fully connected layer tackles the classification task with an output size of 144, whereas the second fully connected layer has an output dimensionality of 2 corresponding to the latitude and longitude of the targets. These outputs were compared against the targets separately using cross-entropy loss and mean squared error respectively, and combined into a single loss score. Furthermore, we multiplied a parameter alpha against the regression loss to observe the impact of the regression loss on training.

To combat exploding and vanishing gradients, we normalized the latitude and longitude values for each validation set in the regression task. To measure our success, we used the standard accuracy based on the classification layer output, as well as individually measuring the mean euclidean distance in meters between the target and both regression layer outputs and the center coordinate for the predicted class cells.

Results can be found in Table 2. It can be observed that all models performed worse than the standard ViT-B/16 model using classification only. We can see accuracy decrease while average euclidean distance increases as we raise the value of alpha. The regression loss was more sensitive to changes in the value of alpha, and followed a less stable trend than the classification distances. It is worth noting that although we achieved overall worse performing results compared to only classification, a lower accuracy compared to ViT-L/16 shows a decrease in distances between the classification L2 Norm. This suggests that the regression loss may actually be assisting in guiding the classification to be closer to the correct values, rather than predicting classes that are further away. More experimentation is required to solidify these results. Results can be found in Table 2.

4. Conclusion

Our study shows that fine-grained geolocation within a dense urban area is feasible using modern vision models and adaptive spatial partitioning. Among the architectures tested, classification-based approaches—especially ConvNeXt—proved most effective, while pure regression and multi-task variants struggled to match their performance.

Future work could explore hierarchical cell structures to reduce boundary errors, apply automated filtering to improve data quality, or incorporate retrieval-based refinement for higher spatial precision. Extending the approach to multiple cities would also provide insight into how well these

models generalize beyond a single urban environment.

5. Work Division

See 3 in Appendix 5 for details on how the work was divided.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[2] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[3] Lukas Haas, Michal Skreta, Silas Alberti, and Chelsea Finn. Pigeon: Predicting image geolocations, 2024.

[4] James Hays and Alexei A. Efros. im2gps: estimating geographic information from a single image. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] Yujun Hou, Matias Quintana, Maxim Khomiakov, Winston Yap, Jiani Ouyang, Koichi Ito, Zeyu Wang, Tianhong Zhao, and Filip Biljecki. Global streetscapes – a comprehensive dataset of 10 million street-level images across 688 cities for urban science and analytics. *ISPRS Journal of Photogrammetry and Remote Sensing*, 215:216–238, 2024.

[7] Sho Kiami and Zach Chapman. Geoknowr: A lightweight geoguessr ai. `https://github.com/shokiami/GeoKnowr`, 2022.

[8] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022.

[9] Tobias Weyand, Ilya Kostrikov, and James Philbin. Planet - photo geolocation with convolutional neural networks. *CoRR*, abs/1602.05314, 2016.

A. Work Division

Student Name	Contributed Aspects	Details
Nicolas	• Remote compute setup • Removed data download blockers • Trained proof of concept models • Optimized GPU utilization • Regression experimentation	• Configured environment on RunPod (online GPU-computing platform) and shared instructions with the team. • Wrote a download script that worked around an issue with the dataset's default downloader, enabling us to obtain lat/lon coordinates for 190k points in Washington DC instead of 50k. • Trained the proof-of-concept model with 144 classes, which later evolved into our final model. • Found some optimizations to boost GPU utilization and reduce training times • Contributed in the problem definition phase, executing initial training runs on candidate problems (e.g. city classification) • Experimented with different geographic partitioning algorithms for data preprocessing and created visuals used in the final report. • Experimented with a regression approach instead of classification.
Branislav	• Contributed dataset downloading scripts • Developed baseline model and training procedure • Developed adaptive partitioning algorithm • Developed euclidean distance metric for center-of-cells • Developed visualizations for inference • Experimented with multitasking approach	• Discovered and contributed download scripts to experiment with multiple cities and concatenated multiple download scripts into a single runnable pipeline • Wrote the code for the baseline ViT model for training and evaluation • Wrote a recursive algorithm to partition any set of latitude and longitude values into a grid of semi-balanced cells • Wrote a script to measure the average distance between the center of predicted classification cells and their actual targets • Created model visualizations including saliency map, adaptive partitioning map, and image patches • Experimented with augmenting our classification approach to combine losses for classification and regression • Contributed optimizations to reduce training such as lowering calculation precision and pre-compiling the model during training
Louis	• Tuned and experimented with hyperparameters to improve the performance of our baseline ViT-B/16 model • Comparison of CNN vs Transformer performance	• Experimented with different learning rate, weight decay, scheduler, warmups, epochs, batch size, and label smoothing values to allow the model to fully learn features and patterns from the training data and greatly improve generalization • Generation of learning curves, evaluation of over/underfitting and learning convergence. • Took the tuned hyperparameters of ViT-16/B and trained Resnet-50 and ConvNeXt models to evaluate their performance against the vision transformer architecture.
Matt	• Helper script for remote setup • Setup and trained multiple model architectures • Trained models on multiple partition setups	• Wrote a small script to enable faster RunPod usage. • Expanded training for the initial ViT Base 16 model to include ViT Base 32, ViT Large 16 and ViT Large 32. Also trained the ConvNeXt-Large model to establish a more robust baseline for comparison. • Setup multiple RunPods to enable parallel training along with local training. • Ran experiments on different partition sizes. • Ran experiments to confirm model performance in a small-data regime. • Refactored partitioning script to be more cli friendly.

Table 3. Contributions of team members.

B. Pre-Trained Model Weights

Model	Weights	URL
ViT-B/16	`ViT_B_16_Weights.IMAGENET1K_V1`	ViT-B/16 Weights
ViT-L/16	`ViT_L_16_Weights.IMAGENET1K_V1`	ViT-L/16 Weights
ViT-B/32	`ViT_B_32_Weights.IMAGENET1K_V1`	ViT-B/32 Weights
ViT-L/32	`ViT_L_32_Weights.IMAGENET1K_V1`	ViT-L/32 Weights
ResNet50	`ResNet50_Weights.IMAGENET1K_V2`	ResNet50 Weights
ConvNeXt-Base	`ConvNeXt_Base_Weights.IMAGENET1K_V1`	ConvNeXt Base Weights
ConvNeXt-Large	`ConvNeXt_Large_Weights.IMAGENET1K_V1`	ConvNeXt Large Weights

C. Code Repository

https://github.com/branisk/Geo-Guessrs