CIS 4190/5190: Applied Machine Learning Fall 2024

Final Project Report

Team Members: Vineet Pasumarti, Adam Peles, Michael Imevbore *Project: Img2GPS*

1 Summary

In this project, we sought to train a model that could predict the GPS coordinates of an input image of a location in the UPenn Engineering quad. We collected a robust dataset of 741 images capturing as much distribution in potential variables as possible. Building off of the provided baseline model, we chose to stick with the ResNet architecture due to its proficiency in computer vision tasks. After varying the depth of our network as well as various hyperparameters, we observed that a pretrained ResNet-18 with a training batch size of 16, a validation batch size of 32, a number of epochs of 15, a learning rate of 0.001, the Adam optimizer, and a learning rate scheduler with step size of 5 and gamma of 0.1 was the best performing model. We evaluated the performance of the baseline ResNet-18 and the more complex ResNet-50, theorizing that its greater depth would allow it to capture more complex image features and thus perform better. However, we observed high bias in the deeper network, and thus opted for ResNet-18. We also observed better performance with a smaller initial learning rate and a sweet spot training batch size of 16 (as opposed to 32 or 8). Disparities in ResNet depth and learning rate are further detailed in the exploratory questions section. Our best performing model achieved a final Root Mean Squared Error (RMSE) of 68.78 meters on our held out 60 image validation set and a RMSE of 65.969 meters when tested for our leaderboard submission.

2 Core Components

1. **Data Collection and Dataset:**

   The dataset we crafted for this project consists of a large variety of images from the test region (the engineering quad, bounded by Walnut street, Spruce street, 33rd street, and 34th street). Photos were taken from a range of locations in the test region, with photographers ensuring that all angles were accurately captured by rotating and taking a series of photos from each location. Additionally, in order to further diversify our dataset, we took photos on multiple different occasions, ensuring we were able to capture the test region across different times of day, lighting, and weather (in particular, we made sure to photograph on a rainy day). This also helped combat variance in moving pieces that may appear in our images, such as people passing by, cars, and construction work. Essentially, we aimed for immobile significant landmarks to be the only constant in our images. Initially, our dataset consisted of 360 images in total, but open observing subpar performance from our model, we aimed to provide more data for it to be trained on and increased the size of our dataset to 741 images. In terms of post processing, we extracted the EXIF data of all images, pulling GPS location in order to assign training labels to each image. We resized all images to 224 x 224 pixels and normalized latitude and longitude labels. Finally, we chose to create a validation dataset consisting of a random selection of 60 images to evaluate our models performance before submission. Thus, our models were trained on the remaining 681 images.

2. **Model Design:**

   We began by running the baseline ResNet-18 on our dataset, observing solid but improvable model generalization on both our validation set and leaderboard test data. Our initial thinking was to use a deeper model architecture, specifically ResNet-50, for stronger learning of features present in images. We also opted for a more stochastic gradient descent approach experimenting with smaller training batch sizes (initially 8 images, then 16) in order to best escape local minima in the loss landscape. Additionally, we experimented with different (specifically larger) initial learning rates, hoping these would also help us reach global minima rather than getting stuck beforehand. We observed multiple interesting conclusions about the architecture of our best performing model. First of all, in general, the

simpler ResNet-18 outperformed the ResNet-50. We observed higher bias in our ResNet-50 models, the opposite of what we had expected. Thus, we concluded that with the size of the dataset we collected and the simplicity of our task at hand, the less complex model exhibited superior performance. In addition, among these ResNet-18 models, we observed the best performance in the one with the smallest initial learning rate (0.001). We theorize that a larger initial learning rate may be overcorrecting our model weights, and that a learning rate of 0.001 combined with a learning rate scheduler is most sufficient in minimizing loss. Finally, after trying training batch sizes of 8, 16, and 32, we observed the best performance with a training batch size of 16. This batch size is best for optimal loss minimization at a learning rate of 0.001, as we found that a batch size of 8 had less stable training and worse overall performance, while a batch size of 32 did not consistently make enough updates to learn robust weights. Ultimately, we ended up with our best performing model being ResNet-18 with a training batch size of 16, a validation batch size of 32, a number of epochs of 15, a learning rate of 0.001, and employed an Adam optimizer and learning rate scheduler with step size of 5 and gamma of 0.1.

3. **Evaluation and Model Performance:**

   Internally, we evaluated our models by computing the Root Mean Squared Error on a validation dataset of 60 randomly selected images. We chose to submit the model with the lowest validation RMSE after the final training epoch to the leaderboard. In addition, we kept track of the training and validation Mean Squared Error after each epoch. In order to better conceptualize our model's performance, we created scatter plots of the actual GPS coordinates for each validation image vs the model's predicted coordinates, adding in error lines between points related to the same input image.

3 Exploratory Questions

3.1 Question 1: Learning Rate and Batch Size

1. **Question and motivation:**

   How can we exploit the relationship between learning rate and batch size to train a model that generalizes well on an image dataset?

   The loss landscape of high-dimensional feature spaces is bound to have many local minima that deter gradient descent from easily reaching a global minimum. To ensure performance on the test dataset, our model must learn meaningful parameters during training that facilitate effective generalization on unseen data. Therefore, we should enhance the fine-tuning process by exploiting the relationship between learning rate and batch size.

2. **Prior work or course material:**

   To cite Yann LeCun, "Friends don't let friends use batch sizes greater than 32." Our intuition entering this project was to retain a small batch size to boost the stochastic nature of mini-batch gradient descent. In doing so, we hypothesized that the noise associated with using a small batch size will help traverse through saddle points and local minima in the loss landscape. Despite our motivation to increase stochasticity, we avoided pure stochastic gradient descent on the premise that it is too inefficient and will take much longer to train on our GPU.

   Recent theory suggests that the learning rate should be multiplied by k when using large mini-batches of size kN[2]. 3 years prior to this work, Alex Krizhevsky provided evidence that learning rate should actually be multiplied by the square root of k when multiplying batch size by k. By doing so, the variance in the gradient expectation remains constant[3]. We began our fine-tuning process with a batch size of 8 – a quarter of the default batch size (32) provided by the baseline model. This implied that our initial learning rate of 0.001 should be halved to 0.0005. However, we approached this recommendation with caution, given that 0.001 is already considered small. Ultimately, we determined that the existing learning rate scheduler would naturally decrease the learning rate to an adequately small value by the time of convergence, thus justifying our decision to maintain 0.001 as the initial

learning rate.

We also believed that a relatively larger initial learning rate will help overcome saddle points and local minima. Our intuition was that the noise of a small batch size may produce a gradient that will help overcome local minima; however, a very small learning rate will lead to a minuscule step, and therefore would not suffice to escape local minima and saddle points. Although we decided on 0.001 as our initial learning rate, we chose to try larger learning rates like 0.003 and 0.005 in the fine-tuning process.

3. **Methods for investigation:**

We evaluated the learning rates and batch sizes for both ResNet18 and ResNet50 according to the test matrix shown in Table 1. The content of this exploratory question, however, will revolve around ResNet18. Our test domain for learning rate was 0.001, 0.003, and 0.005. Our domain for batch size was 8, 16, and 32. Table 1 displays the RMSE on our validation set in meters and we qualitatively observed the rate of convergence by plotting loss against epochs for each run.

Table 1: Validation RMSE of Learning Rates and Batch Sizes on ResNet18

Learning Rate	Batch Size 8	Batch Size 16	Batch Size 32
0.001	79.89	68.78	70.12
0.003	91.42	91.79	92.63
0.005	-	93.61	93.39

4. **Results and updated beliefs:**

According to our results, the optimal learning rate was 0.001. We disproved our original hypothesis that a relatively larger learning rate would increase our validation accuracy by providing an adequate step-size to overcome local minima and saddle points in the loss landscape. Instead, as our learning rate increased, the validation accuracy decreased, as shown visually in Figure 2 below.

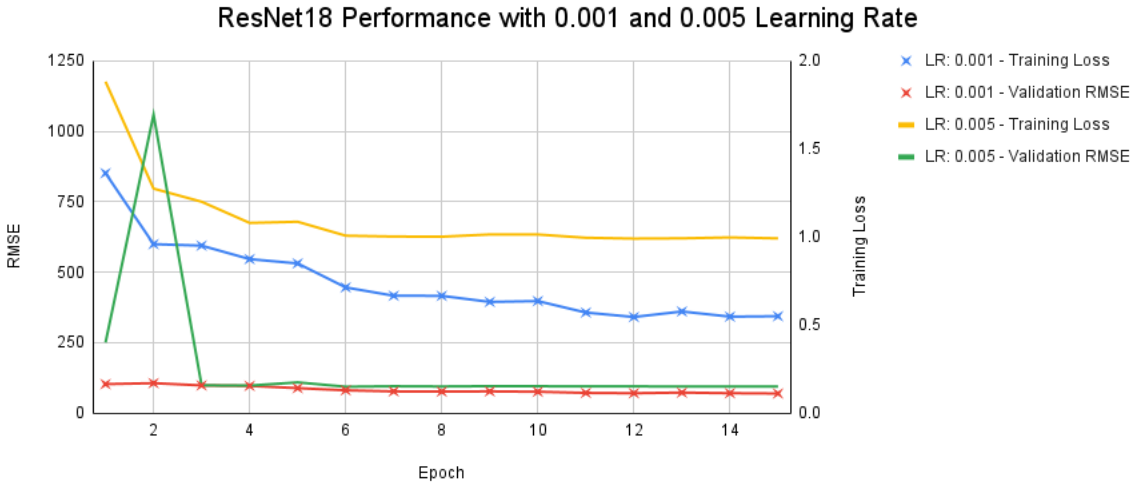

Figure 1: Training Loss and Validation RMSE plotted against epochs for two ResNet18 models trained on two different learning rates (0.001 and 0.005)

As seen in Figure 2, the ResNet18 model trained with a learning rate of 0.001 exhibited superior performance on both the training and validation datasets compared to its counterpart trained with a learning rate of 0.005. Both models underwent training for 15 epochs with a batch size of 16 and a validation batch size of 32, and utilized an Adam optimizer and a learning rate scheduler with a step size of 5 and gamma of 0.1. The model trained with a 0.005 learning rate clearly underfits and struggles to achieve the same level of accuracy on the training data as the model trained with a 0.001 learning rate.

Most importantly, we can visually observe how the training loss with a 0.005 learning rate converged at epoch 6 – far sooner than the training loss with a 0.001 learning rate, which converged at approximately epoch 12. This observation strongly validates our classwork as we observed in our own lectures how learning rates that are too large will converge too early and retain a significant bias. We can also conclude that a learning rate of 0.001 is not too small; otherwise we should observe a much higher initial training loss and eventually a point of intersection between both training loss curves where the too-large learning rate is outperformed by the too-small learning rate. Therefore, we can conclude that 0.001 was a strong selection for our ResNet18 model.

5. **Limitations:** Discuss any limitations of your investigation, and describe what additional steps you might take if you had more time or resources.

   Our biggest limiting factor is in the nature of our testing matrix – holding all but one hyperparameters constant in order to compare model performance is an inefficient method of fine-tuning. Although using a classic test matrix with strict control variables adheres to the scientific method, employing a deep understanding of the learning rate and batch size relationship allows one to make reasonable assumptions and cut down on the number of test cases. With more time and resources, we would extend our experiments on learning rate and batch size to include learning rate decay as well.

3.2 Question 2: ResNet18 vs ResNet50

1. **Question and motivation:**

   Will ResNet50 outperform ResNet18, and why?

   Although both ResNet18 and ResNet50 are used for image classification, we wanted to see which would perform best and be most generalizable on the validation dataset, especially since our validation dataset is meant to emulate the distribution shift in the test dataset used on the leaderboard. Therefore, having an edge on our validation set should also transfer to high performance on the leaderboard.

2. **Prior work or course material:**

   ResNet50 is a significantly deeper neural network with 50 layers compared to ResNet18 with only 18 layers. Our belief was that a larger neural network, like ResNet50, would learn to put more weight on consistent landmarks like trees and buildings over cars or people, and therefore perform better on our validation set. This is because of the inherent nonlinearity of a neural network, so the increase in layers also means an increase in features and parameters; this allows the model to learn more effectively given a complicated input dataset like that of images. Additionally, the bias-variance trade-off suggests that we may overfit the training data due to the complexity of the deeper neural network. When comparing ResNet18, ResNet50, and ResNet101 in existing literature, we see that "deeper ResNets, such as ResNet-50 and ResNet-101, achieve better results compared to their shallower counterparts"[1] for visual geo-localization. Additionally, "ResNet-50 provides a good trade-off between accuracy, FLOPs and model size", while "ResNet-18 allows for much faster and lighter computation, making it the most efficient, lightweight backbone despite performing worse"[1]. Based on this article and our prior knowledge, we would expect ResNet50 to achieve better GPS coordinate prediction accuracy due to its deeper architecture allowing for more complex feature extraction. We expect ResNet18 to show faster training and inference times but potentially higher prediction error

3. **Methods for investigation:**

   To compare the performance of ResNet18 v ResNet50, we will hold all hyperparameters constant and plot the training loss and validation RMSE curves to visually compare the rate of convergence and final loss at convergence. We will also plot latitude vs longitude for predictions and labels to visually see clustering that imply bias. Our hyperparameter selection followed the test matrix shown in Table 1.

4. **Results and updated beliefs:**

According to our data, ResNet18 actually performed better than ResNet50 in correctly predicting GPS locations. Both training loss and validation RMSE for ResNet18 were lower in all of our hyperparameter combinations. Additionally, ResNet18 consistently saw less bias in latitude and longitude plots compared to ResNet50. This defied our initial beliefs in two major ways. One, we expected ResNet50 to outperform ResNet18 on the validation dataset. Two, we expected ResNet50 to potentially overfit the training data and induce variance – not more bias. As seen Figure 2 below, the training loss curve for ResNet50 is higher than that of ResNet18 despite reaching convergence around the same time. This is a clear indication of underfitting the training data and therefore increasing bias. We can assert that our ResNet50 model did not adequately learn effective parameters and we believe the culprit is insufficient data.

ResNet50 is a deeper neural network and requires a larger dataset in order to train effectively. Our training set of 681 images, although useful for ResNet18, is not enough data for ResNet50 to generalize over. This argument is also intuitive – for ResNet50 to combat distribution shift by learning features like landmarks and ignoring features like people and cars, it will need a tremendous amount of data that repeatedly showcases important features. In our case, the deeper network has too many parameters to effectively learn from our limited dataset, and therefore failed to capture underlying patterns in the feature space.

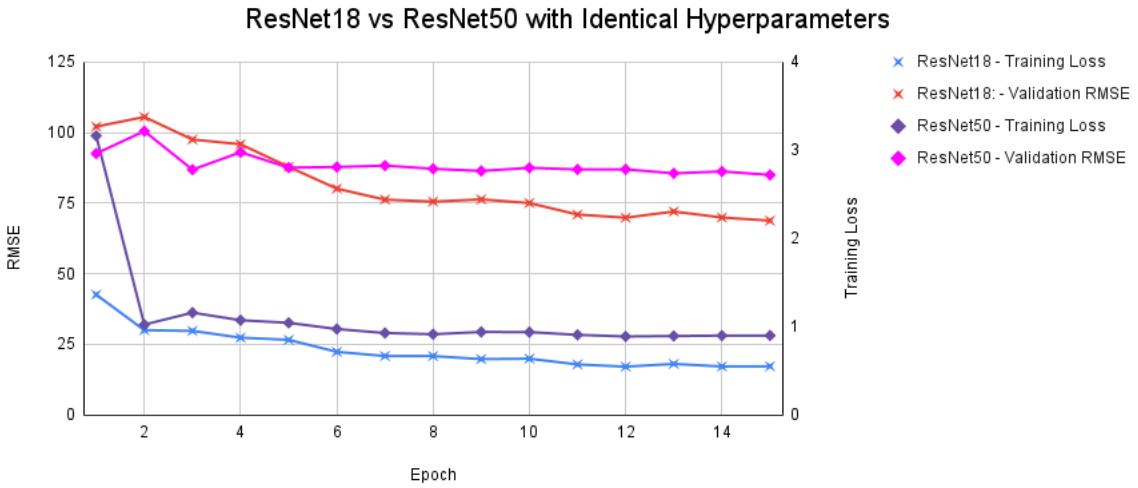

Figure 2: Both ResNet18 model and ResNet50 model are trained for 15 epochs with batch size of 16, validation batch size of 32, learning rate of 0.001 with Adam optimizer, and learning rate scheduler with step size of 5 and gamma of 0.1.

5. **Limitations:**

   As aforementioned, our major limitation was size of dataset in the case of training ResNet50. Although training on 681 images works well for ResNet18, effectively using a larger neural network like ResNet50 will require substantially more images. Medical literature indicated a major increase in performance when training ResNet50 on 291,000 images compared to the initial 20,000 images[4]. We could revisit ResNet50 after capturing closer to ten thousand images.

4 Team Contributions

Vineet Pasumarti: Data collection, model design and training, and wrote the exploratory questions section with Michael.
Adam Peles: Data collection, wrote the summary and core components.
Michael: Data collection and wrote the exploratory questions section with Vineet.
All collectively discussed exploratory directions.

HuggingFace Repository: https://huggingface.co/Imagetogps/ImageToGPSproject_base_resnet18/tree/main

References

[1] Gabriele Berton, Riccardo Mereu, Gabriele Trivigno, Carlo Masone, Gabriela Csurka, Torsten Sattler, and Barbara Caputo. Deep visual geo-localization benchmark, 2023.

[2] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018.

[3] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks, 2014.

[4] Yee Liang Thian, Dian Wen Ng, Jia Tian Pei Daphne Huynh, Priyanka Jagmohan, Shao Ying Soh, Joycelyn Siew Ai Mak, Qing Sheng Tan, and Mengling Feng. Effect of training data volume on performance of convolutional neural network pneumothorax classifiers. *Journal of Digital Imaging*, 35(6):1667–1678, 2022.