AI GeoGuessr: Determining the Location of Google Street View Imagery Using Deep Learning

Peter Lai
Student# 1005332542
peterl.lai@mail.utoronto.ca

Liza Zoubakina
Student# 1004899425
liza.zoubakina@mail.utoronto.ca

Adrien Mery
Student# 1006905850
adrien.mery@mail.utoronto.ca

Joshua Francis
Student# 1006764510
joshuavd.francis@mail.utoronto.ca

Abstract

Building a CNN model that correctly classifies the geolocation of Google Street View imagery, inspired by the online game of GeoGuessr. This project is completed as part of APS360: Applied Fundamentals of Machine Learning at the University of Toronto. —-Total Pages: 9

1 Introduction

GeoGuessr is web-based online game where the player is provided a random Google Street View panorama anywhere in the world, and using what they see they must pinpoint their location on the map as accurately as possible. Restrictions could be imposed, such as any combination of no moving, no panning, and no zooming, which makes the game more difficult. Top players of the game use a variety of clues specific to a country or region to help them narrow down their guesses, including vegetation, language, architectural style, direction of the sun, road line markings, traffic signs, driving direction, license plates, climate and soil, landforms, electricity poles, and "meta" clues such as the generation of the camera hardware, each of which produces images with a distinct aesthetic. All of this knowledge is learnt by the player through many hours spent playing the game and being exposed to a variety of locations, and recognizing patterns between similar locations.

Machine learning, in general, follows a similar process. This raises an intriguing question of whether a computer could achieve the same feat using deep learning and neural networks, which has been shown to be highly effective at recognizing features and patterns within images and associating these characteristics with other information (e.g., the geographic location, in the case of GeoGuessr). This is exactly what we want to achieve, which makes deep learning a suitable approach to this project.

The goal of this project is to train a deep neural network to estimate the approximate geographic location of any given Google Street View image within the continent of North America. This scope was chosen as it has a wide range of landscapes and climate for the model to distinguish between, but not overly so compared to including more continents or the entire world.

The project is interesting because there is substantial parallelism between how a human plays GeoGuessr and how a computer performs machine learning. Many games in the past that were once believed to require human ingenuity, such as Chess and Go, were solved and conquered by the computer, driven by our intrinsic curiosity to make computers outperform humans at these tasks. The same applies to GeoGuessr. The importance of this project lies in the fact that, in the field of machine learning, we are striving to harness the power of computers to help us better understand the world we live in. Google Street View, with its incredible volume of coverage in every continent, provides this (almost) comprehensive representation of the world as we see it, which provides the foundation for opening up many possibilities of what machine learning could do.

2 Illustration of AI Geoguessr Project

A high-level overview of the proposed AI Geoguessr project is provided in Figure 1.

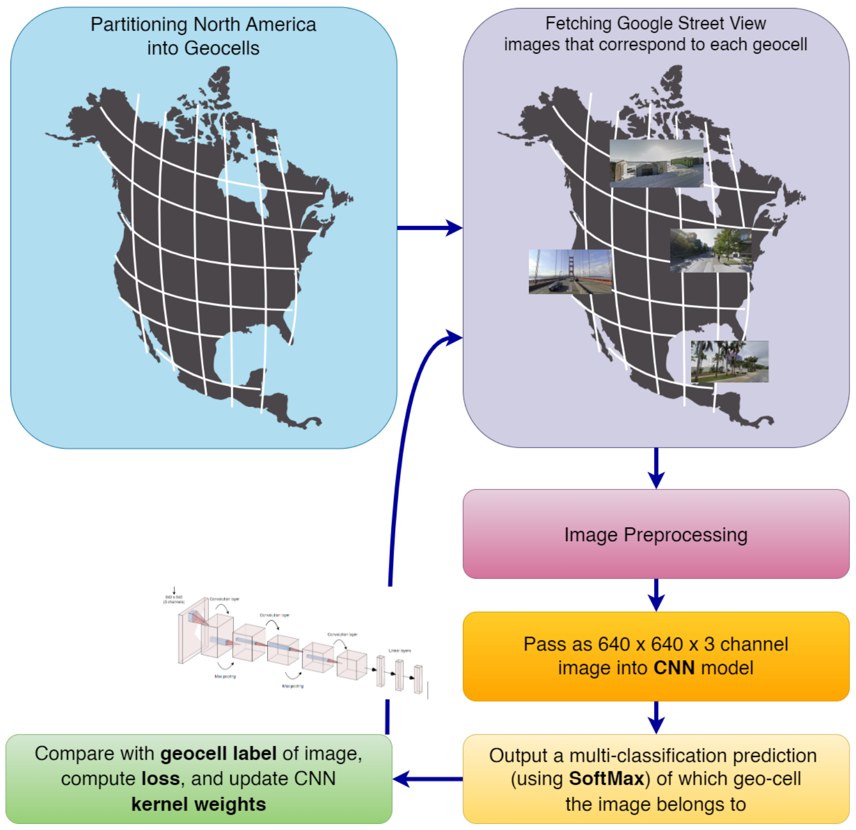
Partitioning North America into Geocells
Fetching Google Street View images that correspond to each geocell
Image Preprocessing
Pass as 640 x 640 x 3 channel image into **CNN** model
Output a multi-classification prediction (using **SoftMax**) of which geo-cell the image belongs to
Compare with **geocell label** of image, compute **loss**, and update CNN **kernel weights**

Figure 1: High-level overview of the AI Geoguessr project.

As outlined in Figure 1, the AI Geoguessr project partitions the continent of North America into a number of geo-cells. These are the partitions which will be later used in our multi-classification CNN that aims to predict the correct geo-cell associated with each image. This partitioning process will be detailed in section 4.

3 Background and Related Work

There have been major developments in the field of media geolocation research since 2008, due to the popularity of social-media and location-based services (Hays & Efros, 2015). Here, we will briefly summarize the progress of research so far, from a traditional Machine Learning approach to the recent Deep Learning breakthroughs. The problem to be solved is always: "estimate the geolocation of an arbitrary photo".

3.1 Im2GPS

A foundational research approach is the Im2GPS system, which is based on a Nearest Neighbor or greedy matching method (Hays & Efros, 2015). It is trained on the Im2GPS dataset (mentioned further as "the dataset"), a database of 6.5M geolocated images taken from the Flickr photo platform (Butterfield, 2022). Conveniently, subsequent research in the field all use this dataset to compare their performance against the Im2GPS system.The Im2GPS Nearest Neighbor algorithm matches a given image to geolocated images in the dataset, and returns the location of this nearest image. The matching technique involves extracting 6 different "image descriptor", characteristics of the image that can be compared against the images present in the dataset (Korel, 2010). These descriptors are:

- tiny 16×16 color images
- color histograms, showing the color distribution of the image (Hadai, 2019)
- texton histograms, showing the texture of the image (Zhu et al., 2005)
- line features, giving the line directions, starting and end point on the image
- Gist descriptors, giving gradient information of the image (Oujaoura, 2013)
- geometric context, estimating the 3D geometric structure of the scene on the image

Using these features, it is easier to compare the given image with the images of the dataset. Solved in this way, the geolocation problem is described as a retrieval problem. However, the Im2GPS system is very complex and requires many photos to be trained and achieve good results. It has not reached superhuman performance.

3.2 PlaNet

In 2016, the PlaNet system was created by applying Convolutional Neural Network (CNN) to the same problem. PlaNet transformed the Machine Learning geolocation problem into a Deep Learning Computer Vision problem (Weyand et al., 2016). To do so, the researchers created an adaptive partitioning of the world based on the geographical distribution of photos in the dataset. In this way, there is a consistent number of photos in the dataset for each geo-cell, which is beneficial for training. The system can become very accurate in cities where the concentration of photos is high (and so the geo-cells are small). The CNN outputs a probability distribution over the surface of the Earth, based on its confidence that the photo comes from a certain area. If more than one area is concerned, the model chooses the geo-cell with the highest probability. Solved in this way, the geolocation problem becomes a classification problem. It achieves a 236% improvement at street-level, and a 50% improvement at country level, over Im2GPS (Weyand et al., 2016). It also uses less memory to train as it does not use the many descriptors extracted by Im2GPS, and it is faster since it does not require a Nearest Neighbor search through millions of image features.

3.3 Combining Im2GPS and PlaNet

In 2017, researchers decided to combine the retrieval and classification approaches of Im2GPS and PlaNet to achieve an even better performance (Vo et al., 2017). They used CNN (PlaNet) as a feature extractor, instead of image descriptors (Im2GPS), to find the characteristics of images taken in the same area. They then used a Nearest Neighbor search (Im2GPS) to match the image to a geo-cell on the map, using an adaptive partition (PlaNet). They achieved a 10% accuracy increase over the PlaNet system, while only using 5% of the training data used by PlaNet.

The problem of image geolocation in the research summarized above is the similar to the problem to be solved in the context of this project. The difference is the source of the data (c.f. Section 4). This brief review confirms our choice of using CNN to solve the problem, detailed in Section 5.

4 Data Processing

4.1 General Approach

Before presenting the specifics of data processing, it is important to note the general approach we are taking for this project. The general form of the data will be a set of Google Street View images each paired with its geographic coordinates (latitude and longitude), which naturally may be suited for a regression approach to be taken by the machine learning model. However, we will be approaching this as a **classification** problem, by partitioning the area within our scope (i.e., North America) into equally-sized geo-cells, generating a unique ID for each geo-cell, and using that ID as the class label, instead of using the lat/lon coordinates as the label value directly. This geo-partitioning approach was used and validated in the PlaNet project (Weyand et al., 2016) as mentioned in Section 3, for the reason that it gives the model a means to express uncertainty as reflected in the output confidence probabilities in each of the geo-cells. It also allows us to abstract and simplify the task for the neural network. To obtain the final estimated location output in lat/lon coordinates, we can simply take the coordinates of the center of the geo-cell that the model predicts with the highest probability.

4.2 Creating the geo-cells and Selecting Image Locations

Based on this approach, the first step of data processing is to create the geo-cells by partitioning North America, which we are defining as the combination of Canada, the contiguous United States, Alaska, and Mexico. The specific number of geo-cells and the size of each geo-cell will need to be experimented with. Each geo-cell is to be assigned a unique integer ID from 1 to n, where n is the total number of geo-cells. Each geo-cell is rectangular, and the boundaries of each geo-cell is determined by the lat/lon coordinates of the four vertices.

Within each geo-cell, we will sample random locations (lat/lon coordinates) to collect images from. Each geo-cell will have the same number of images in the dataset to keep the distribution balanced. In comparison, simply randomly sampling the images over the entire project scope area is more prone to the data being skewed to more urban locations (corresponding to those associated geo-cells) due to the inherently higher density of available Street View coverage (and roadways themselves) in urban areas.

4.3 Obtaining the Images

With a complete list of lat/lon coordinates to collect images from as well as the ID of the geo-cell each one belongs to, the images will then be collected using the Google Maps Street View Static API (Google, 2022), using Python code to automate the process. The collection of each image consists of two steps.

4.3.1 Checking the imagery metadata

Before requesting the actual image, we first check its metadata. A metadata request will be sent to the API with the lat/lon coordinates of the desired location. The server will search for the closest location with available Street View imagery within a set radius, and return information about the resulting imagery including the copyright owner, date, actual location coordinates, and a unique panorama ID. If there is no image found, it will return an error. We want to make sure that the copyright owner is always Google instead of a third party, in order to ensure consistency in the camera and image characteristics. In the event the image is not found or the owner is not Google, we will scrap the location and try a different random location within the same geo-cell.

4.3.2 Requesting the Image File

If the image is valid from the metadata request, we will request the image itself from the API using the unique panorama ID, and the API will return the image as a file. Other parameters to be specified in the request include:

- Size (dimensions, i.e., width $\times$ height, of the image) - We will choose to download at the maximum allowed resolution of 640 $\times$ 640. This is likely higher resolution than what may be used in actual training, but since the usage of the API has an associated cost, it is better to get the images once, and then downscale and crop locally later if needed.
- Heading (compass heading between 0 and 359 degrees, with 0 being north, 90 being east, etc.) - A random value will be chosen.
- FOV (field of view in degrees, or essentially the zoom level) - We will keep the default of 90 degrees, which is relatively wide as the maximum is 120 degrees. Too wide of an angle could make important features harder to distinguish, too close of an angle could exclude those important features altogether, and either extreme is undesirable.
- Pitch (the amount of tilt upwards or downwards for the camera) - We will always keep it to be zero, meaning no camera pitch and looking directly forward, which will provide the most amount of useful information for the model to learn, especially with a relatively wide FOV.
- Source of the image - We will enforce this to select outdoor images only.

Depending on the design of the final neural network model, the image files may undergo further processing such as downscaling the resolution and cropping.

4.4 Cost of Data Collection

Nominally, there are costs associated with the usage of the Google Maps Street View API. The metadata requests to the API (Section 4.3.1) is free of charge, but requesting the actual images (Section 4.3.2) does incur a charge of $0.007 USD each. However, Google Maps Platform provides a $200 free monthly credit, so effectively, the number of images that could be obtained for free from one API account is around 28,500, which is already a sizeable dataset. In addition, Google Cloud Platform (which Google Maps Platform belongs under) also offers a one-time $300 free credit for new users, so the maximum free credits our team could obtain is $1700 (one member of the team already has an account, so they cannot get the $300 sign-up credit again), which translates to around 242,850 images.

We plan to reserve a portion (e.g., 30%) of the free credit for contingency purposes (see Section 9), so the size of the actual working dataset will be a reduced number (e.g., 170,000). This will be further randomly divided into training, validation, and test datasets in a 70%:20%:10% ratio, while ensuring that each subset has equal distributions across all geo-cells.

5 Architecture

Given that we are classifying the geolocation of images, we will be using the Convolutional Neural Network (CNN) architecture that will be applied to our collection of Street View images.

Some of the inspiration behind our chosen architecture is sourced from Weyand et al. (2016), a paper which describes PlaNet, a machine learning system which uses CNNs to geolocate images (c.f. Section 3). An overview of our chosen CNN architecture can be seen in Figure 2.

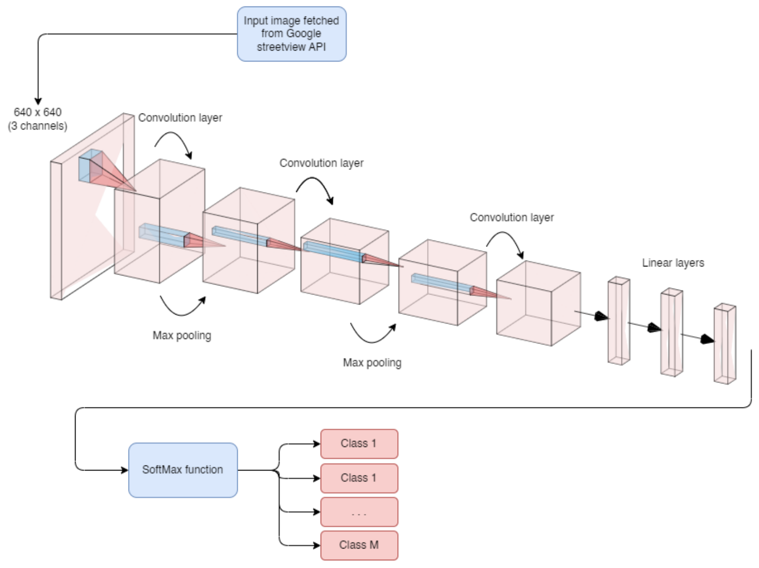

Figure 2: High-level overview of the chosen CNN network.

Our CNN will be based on the Inception architecture using batch normalization and dropout (Szegedy et al., 2014) (Ioffe & Szegedy, 2015). Inception architecture is especially useful in the context of localization and object detection, and its implementation is freely available online by Yang.

As shown in Figure 2, this CNN will consist of several convolutional layers, pooling layers, and conclude with a final fully connected linear layer. Each convolutional layer will have its own kernel that will be specified with a certain size, stride, and padding. As we progress through the layers, the width and height of the feature map will progressively decrease.

At the end of the CNN, we will be using the function SoftMax for the output layer for multi-classification. To train the CNN, kernel weights are initialized to random values and we will make use of mini-batch gradient descent to gain the feedback advantages of stochastic gradient descent without compromising the high levels of computation complexity.

6 Baseline Model

6.1 Human Baseline

The goal of the project is to create a Deep Learning algorithm that will be better at GeoGuessr than the average human. In a game, the locations of 5 Google Street View images must be guessed, and the score of the guess is based on the distance from the guessed location to the actual location, and the time taken to guess. The average human score is 2500 points per round, which gives an average score of 12500 points per game (McBride, 2016). As a first baseline, all members of the group will play the game three times, and we will verify that we get an average score close to 12500 by averaging the three scores of each team member. Since our team has varying levels of experience with the game, we will consider that we can collectively represent an average player. The average score obtained will be our first metric to measure the performance of our algorithm.

6.2 Nearest Neighbor Algorithm baseline

As a second metric, we will use a simple Python algorithm which follows the logic of the Im2GPS algorithm (c.f. Section 3). When the algorithm receives an image, it extracts features and finds the Nearest Neighbor to the image. We will implement this simple algorithm and apply it to our dataset of Google Street View photos (c.f. Section 8). We will use the Python library Scikit (2022) to extract features from images easily. This decision is inspired by our Background and Related work, because the Im2GPS Nearest Neighbor approach is the baseline method against which most recent CNN approaches to the image geolocation problem have been compared (Hays & Efros, 2015; Weyand et al., 2016).

7 Ethical Considerations

Though the simplified model GeoGuessr AI produced during the project poses limited risk to users aside from in-game cheating, this concept can have lethal implications in the world of national intelligence and personal security. The development of large-scale image geolocalization software - one which accepts visual and audio inputs to track a target - can be used by authoritarian governments to covertly track and eliminate political dissidents and humanitarian activists. Furthermore, faulty intelligence rendered by such programs can lead to the misidentification of civilians as hostile targets, resulting in misdirected assault of innocents via drone strikes (BBC, 2021) (US, 2019).

In the wrong hands, this program can give everyday individuals the ability to commit targeted acts of violence via a hacked camera or even an inane social media post (Bekiempis, 2022) - jeopardizing the safety of confidential informants, members of witness protection programs and everyday people. Going forward, we must weigh the ethical consideration of our code being misappropriated and used as an accomplice to stalking, human trafficking, kidnapping, and targeted executions.

Fortunately for our model, its training data is restricted to a relatively small number of Google Street View images (on the order of 100,000 - 200,000) spanning North America and will likely be impractical for location tracking. Given the surface area coverage and the limited number of training examples, it will likely offer relatively low-precision geolocation data - at best, pinpointing large probable geographical areas on the scale of states and provinces.

8 Project Plan

Outlined below in Table 1 are the project milestones and our preliminary schedule to meet these targets; slack will be included in the due dates of all tasks and sub-tasks to account for unforeseen complications and delays.

Table 1: Project Timeline

TASK	SUB TASK	DUE DATE	ASSIGNED
Writing the Proposal	Background and Baseline	Fri, Oct. 14th	Adrien
	Intro and Data Processing		Peter
	Figure and Architecture		Liza
	Ethical Considerations and Project Plan		Joshua
	Risk Register		Everyone
Data Acquisition and Preprocessing	Geocell Partitioning	Sat, Oct. 22nd	Joshua
	Image Selection and Collection	Fri, Oct. 28th	Peter
	Further Image Preprocessing	Sun, Oct. 30th	Liza
Baseline Model	Setting up Im2GPS infrastructure to accept 640 x 640 x 3 images	Mon, Oct. 17th	Adrien
	Using Im2GPS to train on train data and test on the test data set	Mon, Oct. 31st	Adrien
Model Architecture	Setting Up the CNN Model	Mon, Oct. 31st with iterations	Liza
	Training the CNN Model while tuning the hyperparameters	Start from Mon, Oct. 31st End on Sat, Nov. 19th	Everyone
Presentation and Report	Editing the Presentation Video	Friday Nov. 25th	Adrien, Peter
	Writing the Final Report	Fri, Dec. 2nd	Everyone

The team has elected to meet every Wednesday at noon to discuss individual progress, troubleshoot errors, retool internal deadlines and the semester plan. Nearing major deadlines, or as seen necessary, our team or any member reserves the right to set up an ad hoc meeting via Zoom. Our primary communication channel is our Instagram group chat, ”APS360 Team,” where brief project-related questions and discussions are held. In addition, the team has created and will maintain an APS360 Google Drive folder, housing our trials of project code and document drafts and a shared Overleaf for documenting deliverables in LaTeX. To ensure group members' code is not overwritten, code version control via GitHub will be employed.

9 Risk Register

We have outlined a set of risks for our project and how we will best address them.

One potential risk is in data processing. There is a finite number of images we can obtain from the API before we run out of the free credit, so we need to account for the possibility that the images may not turn out to work very well and we may need to get new images using a different set of locations or a fixed camera heading. One mitigation strategy is to do a small test batch where we try to explore the possible edge cases (e.g. extremely rural areas with almost no Street View coverage) and address them, before moving on to getting the rest of the dataset. Also, we should always reserve a portion of the free credit unused in case we need to spill over (meaning the practical dataset size is the not the max number of images we can get for free but under a factor of safety).

Another risk to consider is the computational complexity of our chosen machine learning architecture and the potential lack of GPU resources (limited to 1 GPU on Google Collab). Some cited CNN architectures have taken up to 2 months to converge on 100 CPUs (Weyand et al., 2016). To mitigate this risk, we will work towards modifying existing architectures such that we can utilize less computational resources (using larger batches, increasing the learning rate, decreasing the number of layers). However, this also comes at the risk of achieving a lower prediction accuracy.

Additionally, our chosen machine learning architecture may prove to be not well suited to our geolocation problem. If after training the CNN, we are unable to converge to an optimal solution or our prediction error is too high, then we may need to consider other alternative NN architectures. Some alternatives to consider are to use Residual Neural Networks (RNN) such as ResNet 50, ResNet 100, or Wide ResNet 50 which have also been proven to be successful at image classification (Zagoruyko & Komodakis, 2016)

In the event of a member's departure from the group due to bereavement, illness/injury, or course withdrawal, tasks allocated to that individual will be equally distributed amongst the remaining members.

10 Link to GitHub or Colab Notebook

The public GitHub repository is linked here: `https://github.com/NOBODIDI/APS360_GeoGuessr`. This will contain data processing code and Colab notebooks.

References

Drones reports reveal problems, Sep 2019. URL `https://www.fcnl.org/updates/2016-09/drones-reports-reveal-problems`.

Deadly us drone strike in kabul did not break law, pentagon says, Nov 2021. URL `https://www.bbc.com/news/world-us-canada-59157089`.

Victoria Bekiempis. Everything we know about pnb rock's murder so far, Sep 2022. URL `https://www.vulture.com/2022/09/pnb-rock-murder-case-details.html`.

Stewart Butterfield. Find your inspiration. — flickr, 2022. URL `https://www.flickr.com/`.

Google. Streetview request and response, 2022. URL `https://developers.google.com/maps/documentation/streetview/request-streetview`.

Vandanni Hadai. Histograms: How to read them and use them to take better photos, 2019. URL `https://phlearn.com/magazine/histograms-better-photos/`.

James Hays and Alexei A. Efros. Large-scale image geolocalization. *Multimodal Location Estimation of Videos and Images*, pp. 41–62, 1 2015. doi: 10.1007/978-3-319-09861-6_3/FIGURES/12. URL `https://link.springer.com/chapter/10.1007/978-3-319-09861-6_3`.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL `https://arxiv.org/abs/1502.03167`.

Basia Korel. Final project writeup: Im2gps, 2010. URL `https://cs.brown.edu/courses/csci1950-g/results/final/bkorel/`.

Austin McBride. 5000 points is a good score... - map - geoguessr, 2016. URL `https://www.geoguessr.com/maps/57e06a83a52b274b5cb04da5`.

Mustapha Oujaoura. Block diagram for computation and extraction of gist descriptor. — download scientific diagram, 2013. URL `https://www.researchgate.net/figure/Block-diagram-for-computation-and-extraction-of-GIST-Descriptor_fig5_262954738`.

Scikit. 6.2. feature extraction — scikit-learn 1.1.2 documentation, 2022. URL `https://scikit-learn.org/stable/modules/feature_extraction.html`.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014. URL `https://arxiv.org/abs/1409.4842`.

Nam Vo, Georgia Tech, Nathan Jacobs, and James Hays. Revisiting im2gps in the deep learning era, 2017.

Tobias Weyand, Ilya Kostrikov, and James Philbin. PlaNet - photo geolocation with convolutional neural networks. In *Computer Vision – ECCV 2016*, pp. 37–55. Springer International Publishing, 2016. doi: 10.1007/978-3-319-46484-8_3. URL `https://doi.org/10.1007%2F978-3-319-46484-8_3`.

Tony Yang. Going deeper with convolutions (cvpr 2015). URL `https://worksheets.codalab.org/worksheets/0xbcd424d2bf544c4786efcc0063759b1a`.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016. URL `https://arxiv.org/abs/1605.07146`.

Song Chun Zhu, Cheng En Guo, Yizhou Wang, and Zijian Xu. What are textons? *International Journal of Computer Vision 2005 62:1*, 62:121–143, 4 2005. ISSN 1573-1405. doi: 10.1023/B:VISI.0000046592.70770.61. URL `https://link.springer.com/article/10.1023/B:VISI.0000046592.70770.61`.

Acknowledgments

We would like to thank the Prof and TAs for teaching us ML concepts.