EUGuess - EU Countries Geolocalization given Random Images with Several Perspectives

Jorge Garcelán, Luis Pérez and Carlos Pérez
Georgia Institute of Technology, Atlanta, GA, 30332
[jgarcelan3, lperez63, cjimenez41]@gatech.edu

Abstract

Nowadays, image geolocalization is still a challenging task in the field of computer vision. Most of the state-of-the-art approaches divide the earth in partitioning cells with a certain number of images within an interval, so the cells are unbalanced in area but not in number of images. Our proposal goes further and at the same time becomes more complicated. We try to tackle image geolocalization as a classification task using convolutional neural networks, classifying random localizations generated from random coordinates with their given country, within 32 European countries. Additionally, in this paper we only use images' pixels as features with no prior knowledge, therefore being a single trained end-to-end model with no manually engineered pipeline. Furthermore, we also make a comparison between training our networks with 1 or 3 perspectives of the same locations, getting slight differences, especially in test accuracy. EUGuess confronts several handicaps, but the most significant is the size of the imagery. The complexity of the image generation and the lack of high performing computational resources leads to a small dataset of 14000 locations with 42000 images taking into account the 3 perspectives. Despite the inconveniences, EUGuess significantly improves the accuracy of guessing the location's country randomly.

1. Introduction

Computer vision, the field of artificial intelligence that deals with the analysis and interpretation of visual data, has seen significant advancements in recent years. The use of convolutional neural networks (CNNs), a type of deep learning algorithm, has led to breakthroughs in various applications such as object recognition, image classification, and face detection. However, geolocalization is still a pending work in the computer vision field. Geolocating an image gives uncountable information of the image as we know its location in the maps, e.g, we could know the population of the country, climate information, distance to closest gas station, GDP, well-known landmarks, etc. On the other hand, the model could be used, for example, to identify a location of an old picture or video of ancestors, recognize specific zones where an incident has occurred just using some images or video, or know which country an image comes from, in order to travel there, etc.

Figure 1. Random image from Albania.

Most of the approaches use prior knowledge of the images. Without this knowledge, guessing the country from a single random image can be a challenging task since the image can be very ambiguous with infinite variations such as the resolution and quality of the image, different daytimes, the lighting conditions, the similarity of the locations within a country, vegetation, camera settings or roads. Thus, EUGuess infers locations with no geolocation features, only using pixels. One of the most common approaches is to divide images in three scenes: natural, urban or indoor. Natural scenes gather waterfalls, deserts, beaches or mountains. Urban scenes have buildings, streets, signs or landmarks. Indoor scenes present specific architectures, decorations and interior furnishings. In this paper, we mix the three scenes, leading to a very noisy dataset and complex training process. Moreover, the images are generated from random coordinates bringing a dataset with the majority of natural images of roads without characteristic features as urban buildings or landmarks. These input images are generated around the countries of Europe by using the Google Street View API. The data collection has been one of the main challenges of the project and will be discussed in following sections.

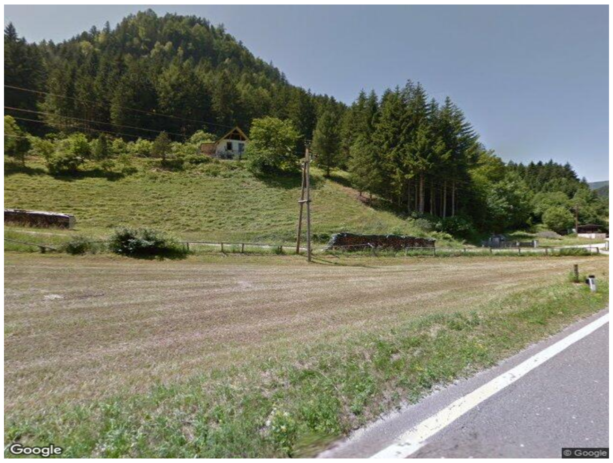


Figure 2. Random image from Austria.

For geolocalization different error thresholds can be used: street level (1km), city level (25km), region level (200km), country level (750km), or continent level (2500km). EUGuess, as one can expect, has an error tolerance at country-scale. Another of the most common techniques used in some of the best geolocalization models such as Im2GPS or PlaNet is the subdivision of the earth into geographical cells with a balanced number of images. As mentioned before, we just divide it into 32 European countries.

The principal goal of EUGuess is to prove that neural networks can easily surpass random and human guessing the country being trained with a tiny random image repository. In addition, as mentioned earlier we also see the difference in performance on training with 1 or 3 perspectives of the same location. When considering three perspectives EUGuess obtains similar information characteristics while when performing with one perspective it gains a broader range of characteristics from each country.

2. Related works

When thinking about related works, the first thing that comes to mind is Geoguessr [1]. It is the inspiration of our work and we have spent lots of time playing it. This related work consists of a geography game which proposes images from places all over the world and you must indicate the location that image belongs to. This game has several modes, for example predicting only the country or you receive more points as near your prediction is to the real location. In our case, our program will have a unique "game mode" which is to predict the country to which the image belongs. There are many differences between our project and Geoguessr. This web allows the user to navigate to spots nearby, to zoom in and pan, so the variations of the image are infinite. Our project deals only with 1 or 3 perspectives with no motion options. Furthermore, Geoguessr works with every image Google Views offers whereas our model is limited by the small sample size we are able to work with. The comparison is not very fair, meaning that in this case, Geoguessr is just a game and our project a predictive model.

Regarding the geolocalization works we are going to focus on previous papers that consist in approaches that are not restricted to specific albums and urban environments, e.g., cities, landmarks, etc. [2–4]. Some other approaches use satellite aerial images [5] or 3D model cities [6–8]. One can also find works that use Google Maps Street View images [9, 10], however, these two works use more than double images and most of them are cities. In addition, there are several works that focus specifically on natural scenes [11–15].

In this paper, we want to focus on two of the most successful works in the field of image geolocalization: Im2GPS [16] and PlaNet [17]. Both projects work at planet-scale not only restricted to Europe, but the idea is the same, predicting all kinds of image locations. Original Im2GPS uses image retrieval; to do this task, Im2GPS matches a given image query with a reference dataset using k nearest neighbors. Later, a new approach was made using deep learning, to try to outperform PlaNet. On the other hand, PlaNet is way more similar to EUGuess, both tackling image geolocalization problems as a classification task, using deep convolutional neural networks and using only pixels as features. Nevertheless, PlaNet estimates a probability distribution over regions, so some of the predictions are between certain locations instead of just one as EUGuess does.

These two works use similar approaches dividing the region into a set of geographical cells based on image density, whereas EUGuess works with a more simpler approach based on countries' area, this will be explained in deeper detail in following sections. The difference in performance is enormous, Im2GPS and PlaNet use 6M and 126M images respectively, completely incomparable to our dataset with 14k locations, furthermore, PlaNet training process lasted 2.5 months on 200 CPU cores.

Another noticeable difference is the fact that both PlaNet, but above all Im2GPS (all images are taken from Flickr), have many images taken by photographers. These pictures have more significant information compared to our images, most of them just photos of trees and roads. In general, photographers do not take photos of this kind of irrelevant landscape. For instance, Im2GPS is very accurate at recognizing pictures of Galapagos Islands, because all the pictures include animals that are characteristic of this archipelago. Our model would just be trained with pictures of Galapagos' jungle. Anyway, PlaNet uses web scraping to take any type of images applying some filters, in this way, it ends up with a lot of noise as EUGuess does. Generating random street view locations leads to some irrelevant and unrepresentative images as indoor pictures.

As one can check in this related works section, EUGuess is significantly far away from other works. Nevertheless, that is part of the essence of this project. We try to prove how neural networks can be better than random choice just being trained with a very little amount of resources.

3. Data Collection

It is commonly known that 70% of the time of a data scientist's work is spent on data. Preparing, processing, loading and cleaning the data are some of the most important tasks in machine learning. In this project, a great part of the work is related to the creation and implementation of proper datasets with locations along Europe. We needed random images from specific EU countries that were not available publicly. Therefore, the dataset was created by us, and funding was provided by the beginning trial that Google Cloud offers, which was a total of \$300 + \$50 per member of the team.

The process of generating the dataset followed the next steps. First, in order to generate images we use the API of "Google Street View" with certain random coordinates (Longitude, Latitude). This call retrieved metadata of the aforementioned location, so we could acknowledge if it was a valid location. If that coordinates lead to a valid location of Google Street View, it generates a jpg 640x640 pixels image and \$0.007 of the budget is used. If the location is not a valid location in Street View we continue until a satisfactory call results in a valid location. Moreover, when calling Google Street API some parameters can be input, such as size, heading, fov and angle of the camera. As we mentioned previously, this last feature was one of the main approaches of this work. We decided to select 3 different perspectives, taking 0º, 120º and 240º.

In order to generate random coordinates within the range of areas of the different countries, we decided to use shapefiles, which are geospatial vector data for geographic information system (GIS) software. These files were downloaded from the EEA (European Environmental Agency) [18]. Shapefiles basically are geolocated grids of such a country, therefore we needed to establish maximum and minimum bounds for each country, so we can take random numbers for latitude and longitude, and thus, generate a random location within that bounds. For doing such task we make use of the geopandas library [19] .

It is important to comment that latitude and longitude are expressed in ETRS89-extended / LAEA Europe coordinates [20], which correspond to EPSG:3035 [21]. However, Google Street View API only takes coordinates in the WGS84 - World Geodetic System, that corresponds to EPSG:4326 [22]. For this reason, we needed to convert such latitude and longitude coordinates into this reference system.

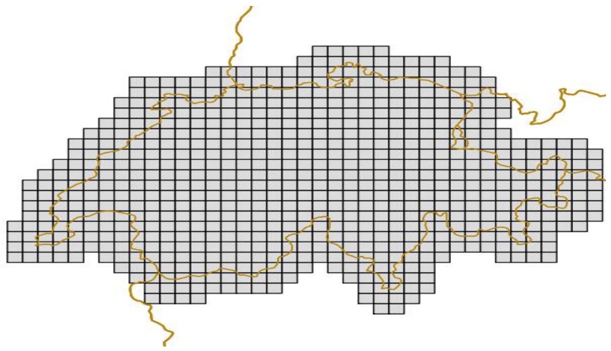
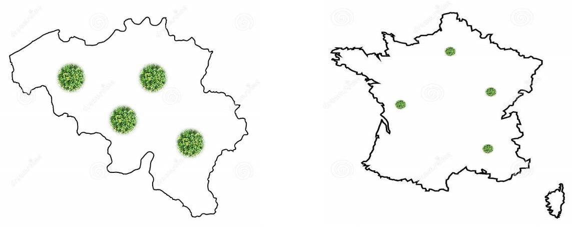


Figure 3. Shapefile of Switzerland.

Following this approach, we have generated 14258 locations and a total of 42774 images taking into account the 3 perspectives. Different countries will have a different number of images, because we consider there are more important countries depending on population, economy, how Google Street View performs in this country, meaning if, for instance, in a mountainous country there are less places where Google Street View has access to, etc.

However, we are specially taking into account the area of the country. In this way, we end up with unbalanced classes, this is because a certain number of images does not represent equally two countries with big differences in surface. For instance, France's area is 248,573 m² whereas Belgium's surface is 11,849 m², meaning that 4 random images are much less representative for France than for Belgium as shown in Figure 4 where the location points are exaggerated.

Therefore, we decided to classify the countries into three



(a) Belgium (b) France

Figure 4. Countries shape with 4 points

categories: small, medium and large. For the first category we have generated 123 locations (369 images) per country, for the second category 369 locations (1107 images) and for the last category 860 locations (2580 images). In Table 1 we present some examples of the number of locations and images per country. By doing this we are creating a dataset that has a number of images proportional to the size of the countries which is reasonable when trying to create a model that predicts the country an image belongs to.

Country	# Locations	# Images
Spain	860	2580
Germany	860	2580
Ireland	369	1107
⋮	⋮	⋮
Croatia	369	1107
Slovenia	123	369
Luxembourg	123	369

Table 1. Locations and Images distribution.

Notice that the code to replicate the data collection can be found in the generateDataset.py file.

4. Technical Approach

In order to test different approaches, we decided to create different folders with different distribution of images. We stored all the images in a main folder continuing with a split of the dataset in different folders for different perspectives. This organization helped us when comparing results and performances.

Thus, after generating the dataset, we have decided to resize all the images to 64x64 pixels and 128x128 pixels. The main reason for carrying out this action is the huge size of the images from Google Street API (640x640 pixels), which resulted in problems related to the space of RAM and execution time needed for loading the images and execution of EUGuess.

The initial tests we made were to decide which image size should we consider. In order to make this decision we evaluated different models with images of 64x64 and 128x128 pixels. After comparing different executions of models we concluded that we should select 64x64 pixels images due to the minimal differences in terms of accuracy and the large difference in execution time and memory needed for storing images of 128 pixels.

Once we had the whole dataset of images resized we decided to create a function named *save_tensors* so as to keep a copy of the tensor of every image in a folder, stored as a pickle object. Moreover, we decided to normalize the resulting tensors with mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225). By doing this we avoid using the function load_data which is in charge of reading jpg files and transforming them into tensors which took several hours for loading the whole dataset.

After saving the tensors we created an additional function named *load_tensors* for loading such tensors into Colab. This function performs exceptionally well in terms of execution time compared to *load_data*, allowing us to try our models without having to wait several hours whenever a restart runtime was triggered or a disconnection occurred in Colab.

The next action to conduct was the split of data into 80% training and 20% test. Additionally, we have considered the fact that some countries have even 5 times less images than others. Therefore, we need to be sure that we are also splitting the images of each country with the same proportion, or we could end up with few training images of a country entailing a bad performance in future evaluations. In order to perform the split, we have created the function *split_train_test*, which uses the *torch.utils.data.random_split* from pytorch library and considers the number of perspectives selected.

After the processing of images, tensors and perspectives we have tested a considerable number of models from the torchvision library. EUGuess has focused its training in four main models: Resnet50, EfficientNetB4, GoogleNet and VGG16. Additionally, it has tested a CNN implemented including several Convolution layers with ReLU as activation function, MaxPooling and BatchNormalization between layers and finally fully connected layers.

The next step was to test the models mentioned above considering one or several perspectives. In order to carry out this test, we have considered only 14258 images as the dataset has that exact number of locations if we gather all the countries. The main reason to conduct this action was to train models with the same number of images independently of the number of perspectives.

For training with one perspective we saved in a folder all the tensors associated with images of perspective 0. Initially, we saved images as *Albania_0_0.jpg*, the first number represents the location number and the second number represents the perspective, so we took all images with 0 in the second index and the folder contained 14258 tensors.

In order to train with three perspectives we saved in a folder of the locations of each country including the three perspectives per location. The folder ended up having 14267 images because for the countries with 860 locations we included 287 locations as the 860 was not a multiple of 3. Therefore, for countries with 123 locations we took the first 41 locations, for countries with 369 locations we took first 123 locations and for the rest of the countries the first 287 locations.

Then, we compared the model's accuracies between both sets of images using the same parameters and splitting values. We considered a split of 80% for training 20 for the test, 200 as batch_size, varying learning rate depending on the models executed and momentum of 0.9. In addition to testing several models, we made use of different optimizers such as Adam or SGD and a "CosineAnnealingWarmRestarts" scheduler that adjust the learning rate in each epoch in order to select the one that provided better accuracy and considered categorical entropy loss as the loss function of EUGuess.

Our final step was to test the models we have considered for the project with the whole dataset of images. For that we used the folder that included the tensors for all the images of 64x64 pixels, in total there were 42774 tensors. After loading the tensors we performed the split of training and test and tuned the hyperparameters used in the models in order to increase the accuracy.

5. Experiments and Results

5.1. Dataset

The first decision that we needed to make was deciding between using 64 or 128 pixels images for EUGuess to train with. Therefore, some tests were performed resulting in selecting images of 64x64 pixels. This decision was made due to the similar accuracies obtained between the models comparing images of 64 and 128 pixels, and the high memory and execution time of 128 pixels in comparison to smaller images. The high memory needed was a result of storing the 128 pixels tensors which increased significantly the RAM used during executions. Furthermore, execution time raised due to introducing larger tensors to the different convolution layers each model performed.

The second main result obtained was the transformation of images into tensors for loading the data needed to run the different Deep Learning models. Instead of using function load_data for reading jpg files whenever Colab was restarted we created functions *save_tensors* and *load_tensors* for doing the same operation. With these two functions we were able to first transform images into tensors and save results into folders using pickle format, and finally load tensors in a much faster execution time compared to *load_data*. The load of the whole dataset of jpg files took between 2 and 3 hours each time the function was executed. The function *save_tensors* took between 10 and 15 minutes to complete the whole dataset and the loading function finished in less than 5 minutes. As you can observe, an important optimization step was achieved that helped us to save time and implement it in testing models.

5.2. Models

The third and main experiment of our project was to test different Deep Learning models with the whole dataset of images. The four models we have considered due to the accuracies obtained when testing around 10 different models are: GoogleNet, EfficientNetB4, VGG16 and Resnet50. After deciding which models we were going to train we followed an exhaustive search of different hyperparameters, optimizers and schedulers in order to improve EUGuess accuracy.

After tuning hyperparameters, we have chosen a batch_size of 1000 images, a split proportion of 0.8 for train and 0.2 for test, a learning rate between 0.005 and 0.0005 depending on the model, 10 epochs and a momentum of 0.9. In addition, the parameters selected for the *CosineAnnealingWarmRestarts* scheduler were: $T\_0 = 10$, $T\_mult = 1$, $eta\_min = 0.001$ and $last\_epoch = -1$. The training and test models outputs are represented in Figures 5, 6 and 7 where we can observe the evolution of accuracies and losses through the different epochs of the training and the test accuracies obtained.

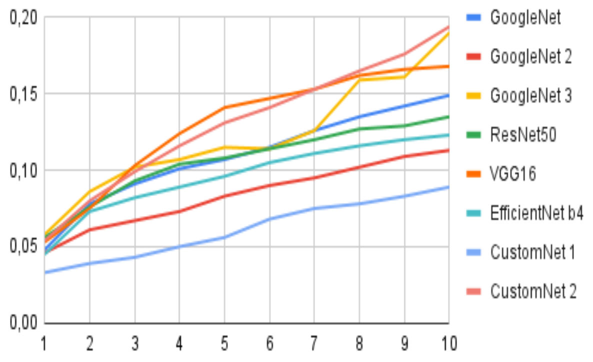


Figure 5. Training Accuracy for 3 perspectives

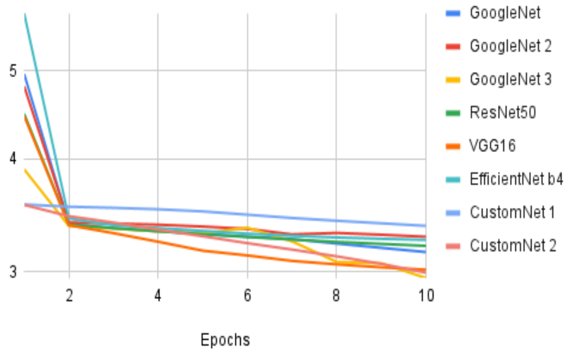


Figure 6. Log Training Loss for 3 perspectives

The main information extracted from these plots is the better performance of GoogleNet when hyperparameters are well tuned as it is the case of GoogleNet3. Due to these results we decided to explore deep into this model looking for a higher accuracy. In order to achieve this goal we tested several schedulers with different parameters and tuning the hyperparameters models.

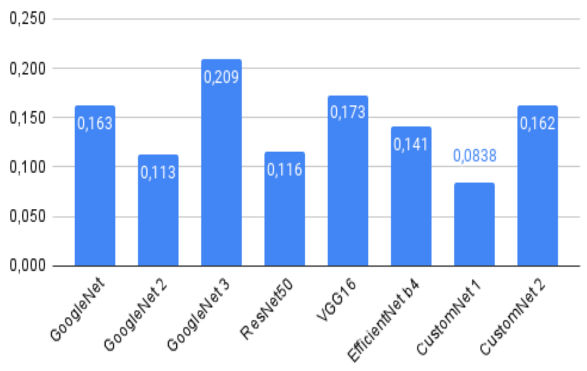


Figure 7. Test Accuracy for 3 perspectives

Our best result is included in Table 2 where accuracies are compared to related works mentioned in this paper. As you can observe, our accuracy is slightly higher than Im2GPS and PlaNet projects if we compare it to the original model of Im2GPS or PlaNet with 900k images. Our accuracy is lower when those projects include a huge dataset of images compared to our project but we also have to notice that this works predict countries all around the world and EUGuess is limited to European countries. We also have to take into account that EUGuess uses 3 perspectives. Furthermore, Im2GPS and PlaNet are world-scale, meaning that these frameworks predict countries throughout the whole globe. Hence, the comparison we make in Table 2 is not measuring equal metrics and definitive conclusions cannot be extracted.

The hyperparameters used for obtaining this result were: $batch\_size = 1000$, $learning\_rate = 0.0005$, $momentum = 0.9$, $epochs = 20$, and the scheduler *CosineAnnealingWarmRestarts* with $T\_0 = 10$, $T\_mult = 2$, $eta\_min = 0.01$ and $last\_epoch = -1$.

Method	Country Accuracy
Im2GPS (orig)	23 %
Im2GPS (new)	35.4 %
PlaNet (900k)	21.6 %
PlaNet (6.2M)	45.6 %
PlaNet (91M)	53.6 %
EUGuess - GoogleNet (42k)	25.1 %
EUGuess - CustomNet (42k)	16.2 %

Table 2. Model Accuracy Comparison.

Last but not least, we have developed several tests to check how models perform with one perspective in comparison to three perspectives. As we have mentioned in the approach we have considered the same number of images

One Perspective		
GoogleNet	**ResNet50**	**EfficientNet B4**
0,148	0,140	0,1196

Table 3. One Perspective Accuracies

Three Perspectives		
GoogleNet	**ResNet50**	**EfficientNet B4**
0,126	0,128	0,123

Table 4. Three Perspective Accuracies

for both cases, including for the test of one perspective all the locations of the dataset but only first perspective and for several perspectives including $\frac{1}{3}$ of the locations per country considering every perspective. Both tests included around 14260 images which is a significantly smaller dataset compared to the previous tests. In tables 3 and 4, we can perceive the different test accuracies obtained with three different models: EfficientNetB4, GoogleNet and Resnet50.

Taking into consideration the previous tables we have concluded that models are performing similarly in both cases due to the following reasons. Whenever we test the one perspective dataset the model is able to recognize a greater number of characteristics per country as it has three times the number of locations compared to the other test. Furthermore, with three perspectives the model can also perform well as images from the same location will be quite similar. This could lead to a model training two of the three perspectives and when testing the third perspective being able to accurately predict the country due to the high similarities between pictures.

6. Conclusions

In the end, we have gotten very decent results. EUGuess reaches unexpected results compared to previous works, this is because of the use of 3 perspectives. We have learnt how image geolocalization is tackled in the computer vision field and added our new perspective, literally. Additionally, our presented work proves that completely random locations from Google Street View can serve to build a model able to guess the country of $\frac{1}{4}$ of these images, just being trained with no more than 42k photos, 14k locations in total. Being honest, we think that this good results come from the fact that in test set we can find some of the perspectives that have obviously similar features to the other perspectives of the same location used when training.

As a curious fact, in early stages of the project, we were unintentionally shuffling the training data, but not its labels, leading to an unmatched train-label set. Despite the mistake, even the struggles we were facing, the accuracy achieved was double of random guesses. EUGuess deals with 32 different countries, meaning that, the chance of guessing randomly a country is around 3%. Likewise, EU-Guess was reaching 6% accuracy. Investigating why this happens, even though the labels seem to not correspond, could be a good idea for future researchers, as well as generalizating the model for countries all around the world.

References

[1] GEOGUESSR, *https://en.wikipedia.org/wiki/GeoGuessr* 2

[2] CNN ARCHITECTURE FOR WEAKLY SUPERVISED PLACE RECOGNITION., *Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: Netvlad. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 5297–5307. IEEE (2016)* 2

[3] LANDMARK CLASSIFICATION IN LARGE-SCALE IMAGE COLLECTIONS., *Li, Y., Crandall, D.J., Huttenlocher, D.P. In: International Conference on Computer Vision. pp. 1957–1964. IEEE (2009)* 2

[4] WORLD-SCALE MINING OF OBJECTS AND EVENTS FROM COMMUNITY PHOTO COLLECTIONS., *Quack, T., Leibe, B., Van Gool, L. In: International Conference on Content-based Image and Video Retrieval. pp. 47–56. ACM (2008)* 2

[5] LOCALIZING AND ORIENTING STREET VIEWS USING OVERHEAD IMAGERY., *Vo, N.N., Hays, J. In: European Conference on Computer Vision. pp. 494–509. Springer (2016)* 2

[6] WORLDWIDE POSE ESTIMATION USING 3D POINT CLOUDS., *Li, Y., Snavely, N., Huttenlocher, D.P., Fua, P. In: Large-Scale Visual Geo-Localization, pp. 147–163. Springer (2016)* 2

[7] EFFICIENT GLOBAL 2D-3D MATCHING FOR CAMERA LOCALIZATION IN A LARGE-SCALE 3D MAP., *Liu, L., Li, H., Dai, Y. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 2391–2400. IEEE (2017)* 2

[8] SKYLINE2GPS: LOCALIZATION IN URBAN CANYONS USING OMNI-SKYLINES., *Ramalingam, S., Bouaziz, S., Sturm, P., Brand, M. In: IROS (2010)* 2

[9] ACCURATE IMAGE LOCALIZATION BASED ON GOOGLE MAPS STREET VIEW., *Zamir, A.R., Shah, M. In: European Conference on Computer Vision. pp. 255–268. Springer (2010)* 2

[10] IMAGE GEO-LOCALIZATION BASED ON MULTIPLE NEAREST NEIGHBOR FEATURE MATCHING USING GENERALIZED GRAPHS., *Zamir, A.R., Shah, M. IEEE Transactions on Pattern Analysis and Machine Intelligence 36(8), 1546–1558 (2014)* 2

[11] BLUEFINDER: ESTIMATE WHERE A BEACH PHOTO WAS TAKEN., *Cao, L., Smith, J.R., Wen, Z., Yin, Z., Jin, X., Han, J. In: International Conference on World Wide Web. pp. 469–470. ACM (2012)* 2

[12] LARGE SCALE VISUAL GEO-LOCALIZATION OF IMAGES IN MOUNTAINOUS TERRAIN., *Baatz, G., Saurer, O., K¨oser, K., Pollefeys, M. In: European Conference on Computer Vision. pp. 517–530. Springer (2012)* 2

[13] IMAGE BASED GEO-LOCALIZATION IN THE ALPS, *Saurer, O., Baatz, G., K¨oser, K., Pollefeys, M. International Journal of Computer Vision 116(3), 213–225 (2016)* 2

[14] DISCOVERING LATENT CLUSTERS FROM GEO-TAGGED BEACH IMAGES., *Wang, Y., Cao, L. In: International Conference on Advances in Multimedia Modeling. pp. 133–142. Springer (2013)* 2

[15] USER-DRIVEN GEOLOCATION OF UNTAGGED DESERT IMAGERY USING DIGITAL ELEVATION MODELS., *Tzeng, E., Zhai, A., Clements, M., Townshend, R., Zakhor, A. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 237–244. IEEE (2013)* 2

[16] REVISITING IM2GPS IN THE DEEP LEARNING ERA., *Vo, N., Jacobs, N., Hays, J. arXiv preprint arXiv:1705.04838 (2017)* 2

[17] PLANET-PHOTO GEOLOCATION WITH CONVOLUTIONAL NEURAL NETWORKS., *Weyand, T., Kostrikov, I., Philbin, J. In: European Conference on Computer Vision. pp. 37–55. Springer (2016)* 2

[18] EEA, EUROPEAN ENVIRONMENTAL AGENCY, *https://www.eea.europa.eu/data-and-maps/data/eea-reference-grids-2/gis-files/folder$_listing$* 3

[19] GEOPANDAS, *https://geopandas.org/en/stable/* 3

[20] EPSG 3035, *https://epsg.io/3035* 3

[21] MAP PROJECTIONS FOR EUROPE, EUROPEAN COMMISSION JOINT RESEARCH CENTRE, REFERENCE EUR 20120 EN, 2003., *A. Annoni et al.* 3

[22] EPSG 4326, *https://epsg.io/4326* 3

Team Contributions

Team Member	Problem Def.	Lit. Review	Data Collection	Code Implem.	Experimentation	Analysis
Luis Pérez	✓	✓		✓	✓	✓
Jorge Garcelán	✓		✓	✓	✓	✓
Carlos Pérez	✓	✓	✓		✓	✓

- Problem Definition: Detailed description of the problem, inspiration and objectives.

- Literature Review: Related work research and necessary tools to perform our project

- Data Collection: Detailed process for generating the images that our deep neural networks will train.

- Code Implementation: Implementation of the functions needed for generating images, transforming them to tensors, loading tensors, and training and testing of different Deep Neural Networks models.

- Experimentation: Images preprocessing, hyperparameter tuning, testing models

- Analysis: Result analysis, comparisons evaluation and graphics creation.