CSCI 1430 Final Project Report: GeoTrainr

Team: Iris Cheng, Ziang Liu, Praccho Muna-McQuay, Manuel Quezada.
TA name: Alexander Choi. Brown University

Abstract

This paper discusses the development of a computer vision system that aims to accurately geolocate images. The approach used in this project is a classification or prediction framework that utilizes convolutional neural networks (CNNs). The system was designed to predict either the precise location (coordinates) or just the country that the image is in. We acquired a diverse set of images from across Europe using the Google Street View API, and subsequently fine-tuned pre-trained models, BiT and ConvNeXt, to accurately classify a country or predict the coordinates of an image.

1. Introduction

Released in 2013 after only two weeks of development, the browser-based game GeoGuessr was an instant hit. The concept was simple – given just the Google Street View at some random location around the world, the player is tasked to pinpoint their exact location on the globe, rewarding higher scores to closer guesses. Now, as its player base has grown to over 40 million globally, the game has been praised for its ability to educate its players on how to study and understand various geographic and cultural terrains. Could a computer vision system do the same?

Two modes offered by GeoGuessr are identifying the precise location, or just the country, from Street View. For the former, the task of the network would then be to predict coordinates (latitude and longitude), which would be continuous values. As for the latter, identifying the country from Street View becomes a classification task, instead. We also consider the task of determining the distance between a given pair of images, which would be a continuous value.

One last consideration in our approach is the input. In GeoGuessr, the user has the ability to move, pan, and zoom around Street View. However, since we are using pre trained CNNs, our input has to be formatted according to that network's architecture, which is most likely just a single image. We thus decided to adjust the task to predicting the precise location and country given just a single frame from Street View. While this deviates from the idea of GeoGuessr, where the user is allowed to move freely, one could theoretically take snapshots around their location and feed these to the model. This also affects how we evaluate the success of a model. Given just one frame from Street View, how accurately would a human be able to identify the country or position.

Previous efforts to build an accurate geolocalization model have yielded mixed results. These attempts often utilized older CNN's or used a query and search framework in conjunction with the neural network. However, our approach utilizes a simple classification or, in the case of coordinates, prediction framework using newer, more advanced CNN's. The ability to construct and train a model to accurately geolocate images would mean much more than allowing one to rise above the ranks in GeoGuessr. In cases where additional metadata is unavailable, having the ability to accurately define the location of an image would be requisite for gathering further information of the captured subject or setting. In an increasingly digitized world, such a model would be an invaluable asset in areas such as photo forensics or the identification of missing persons.

2. Related Work

Building a model that can accurately geolocate an image is far from being an unattempted feat. Previous attempts to do so can be subdivided into two categories – natural and restricted. The former involves geolocation at the global-scale whereas the latter is bound to specific environments or imagery.

Our approach, in which we limit input images to those taken in Europe, would be considered a restricted framework approach. However, while most approaches in this area apply retrieval techniques matching the query image to a reference dataset, ours does not [7]. Instead, we opted for a much simpler framework of direct classification or prediction, but in conjunction with more advanced models.

Although networks available today greatly outperform those employed in previous approaches, we still needed to ensure the selection of models that would best suit our needs. We chose to use BiT and ConvNeXt models due to their abil-

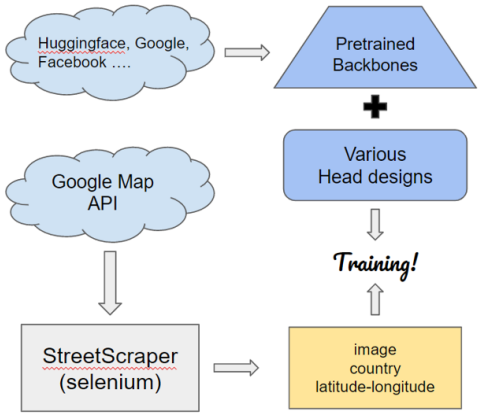

Figure 1: Our proposed workflow.

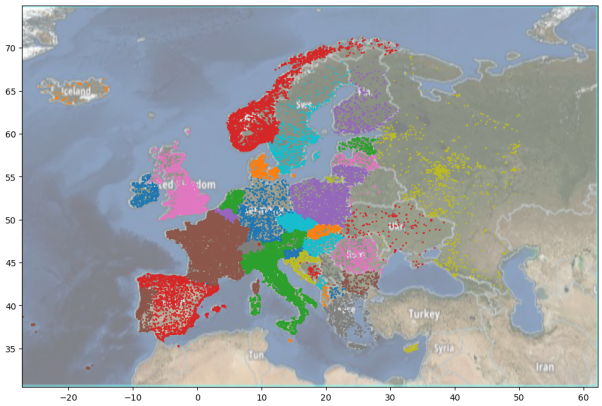

Figure 2: Spatial distribution of all our images.

ity to classify natural images accurately. BiT, which stands for Big Transfer, is a state-of-the-art model that uses large amounts of labeled data from ImageNet and other sources to train a highly accurate image classification model [4]. ConvNeXt, on the other hand, is a CNN model that is designed to perform object detection and recognition tasks on natural images [6].

3. Method

Since this task is one that requires being able to learn and distinguish features from the natural world, and eventually classifying based on these, training a convolutional neural network (CNN) is the ideal approach. Rather than training a CNN from scratch, we decided it was more practical to utilize a pre-trained model's learned features, since ones trained on ImageNet come from the natural world as well. The goal of fine tuning would then be to decide the head of the network, which depends on the particular task.

Another consideration in our approach is the input. In GeoGuessr, the user has the ability to move, pan, and zoom around Street View. However, since we are using pretrained CNNs, our input has to be formatted according to that networks architecture, which is most likely just a single image. We thus decided to adjust the task to predicting the precise location and country given just a single frame from Street View. While this deviates from the idea of GeoGuessr, where the user is allowed to move freely, one could theoretically take snapshots around their location and feed these to the model. This also affects how we evaluate the success of a model. Given just one frame from Street View, how accurately would a human be able to identify the country or position?

3.1. Data

Before considering any models, it was important to ensure that we could acquire the necessary data to train on, which in this case would be a diverse set of images from across Europe. The actual spatial distribution of our data is shown in Fig 2. In order to acquire images, we had to leverage the Google Street View API. For each request, a client gives coordinates and optional camera data (field of view, orientation, etc.) and receives an image at that location. However, if Street View is not available at the passed coordinates, no image is returned. Thus, when generating random, viable coordinates to make requests, we found in the vast majority of cases there were no available photos, since roads do not cover most of the Earth. While this is not necessarily an issue if run long enough to get sufficient images, making unnecessary API requests would be economically inefficient given the limited free credit available. To minimize the number of misses on API requests, we turned to the GeoNames database, which offers free access to the coordinates and associated information of 11 million global locations. Though not perfect, we found that the coordinates of these locations would be near enough to streets to lead to API hits at a significantly higher rate than random coordinate generation. While this does have the potential of bias as the distribution of locations among the countries in Europe is not even, for the sake of finances it made the most sense.

In the final web scraping script, available in webscraper.py, we load and filter out non-European countries from the GeoNames database. We then used Selenium to handle the API requests and save images for a randomly sampled set of locations. This yielded a total of 47,737 images, with resolution 256×256. Examples of such are shown in Fig 3.

One issue encountered in the data collection process was that for some coordinates, the image available was not actually a Google van-captured Street View image, but a shot

Figure 3: Left-to-right: images obtained by webscraping script from France, the UK, and Norway

Figure 4: Left-to-right: unwanted indoor images obtained by webscraping script from Albania, Italy, and Portugal

from inside a building. Some examples obtained from the script are shown in Fig 4. Though a rare occurrence, this does create some bias in the training and testing data as GeoGuessr is strictly outside of any buildings, and could potentially confuse the CNN.

With the images obtained, we split our dataset into two parts – 80% of all images are sampled as a training set and the remaining 20% is the testing set. We set a fixed random seed for reproducibility. The validation set is acquired after all training parts are finished. We use the scraper again to download images not presented in the previous data for validation.

3.2. Models

After acquiring our data, we move on to architecture, which as mentioned in our Related Works section would involve fine-tuning the ConvNeXt and BiT models. With these backbones chosen, we then modify the heads of the models according to our specific tasks – country classification, coordinate prediction and distance prediction. The total pipeline of our experimentation is shown in Fig 5.

Country Classification. This part follows the traditional image classification pipeline and is the initial step of our experimentation. A simple one-layer classification head is added at the end of the pretrained backbone model. Our data are gathered from 46 countries in Europe so 46 is the size of the classification layer. We use a small learning rate and train both the backbone and classification head. Given that it is a categorical classification problem, the most appropriate loss function is categorical cross-entropy. However, we also used label smoothing to account for any noise in our data.

Coordinate Prediction. The architecture part of the coordinate prediction is also quite simple. We still use a simple

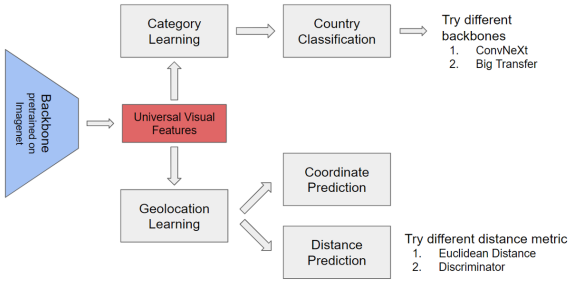

Figure 5: Experimentation pipeline. We try to utilize the visual features to solve three GeoGuessr-related tasks.

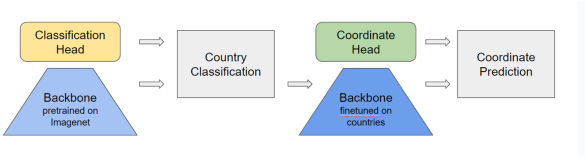

Figure 6: Extra study of the effect of task-aware pretraining.

one-layer head with output size 2. The only difference is that we freeze the backbone in this task. The tricky part is deciding the range of coordinates. Using improper scales of supervision values could easily cause the output values to be out of the sensitive range of activations. To this end, we use a single transform to make sure the coordinates are within the sensitive range of the model activations. All geolocations are transformed into coordinates in a reference frame whose origin is latitude 50N and longitude 10E. The network part of latitude-longitude prediction is just one linear layer with an additional scaling coefficient of 10.

One further attempt we made in coordinate prediction is task-aware pretraining. The backbones we use are trained on ImageNet so they are focused on extracting visual features related to common objects. Training it again on geography-related tasks before applying it to our task would be beneficial [2]. The most convenient task is country classification which we did initially. We use the checkpoint with the highest test accuracy from the first task to initialize our model for coordinate prediction. The training pipeline of this is shown in Fig 6.

Distance Prediction. While mapping visual features to a continuous field of coordinates is quite unusual, learning how different features are is quite common [5, 3]. We tried many techniques to relate geographic distances to visual features. The first attempt is to use Euclidean distance on high-dimensional features. Seeing the features as points on the feature-space manifold, it is reasonable to learn a manifold on which feature-space Euclidean distances can represent geographic distances.

Another way to do this is to use a discriminator that takes

two features as input and predicts the distance between them. Using a discriminator requires more care in the training process. As the experimenting goes, we incrementally added multiple techniques, including distance clipping, sigmoid output and scaling, counter sample balance, fixed anchors, and identity loss.

The purpose of distance clipping is to simplify training by ignoring large distances. By clipping, the network only need to learn the distances between samples that are close to it. In practice, we use the clipping threshold of 10 degrees (1110 km). Sigmoid output is added as an auxiliary part of clipping. It uses the sigmoid activation function to clip the network output with the sample purpose of ignoring large distances. Since sigmoid output is from 0 to 1, we scale it up by 10 to match the distance clipping threshold. Sample balance is the common trick in GANs [1, 8]. If the model is constantly seeing negative samples, which is the case for our data, it will easily be trapped in local minima. To solve this, we modify the sampling procedure so the network sees equal amount of distance samples and close samples. Fixed anchors is to not model distances between arbitrary samples but any sample to some fixed reference points. The reference points, namely anchors, are chosen to be the center of each country. Identity loss has the same effect as sample balance. It asks the model to predict the distance between a feature and itself (which should be 0).

3.3. Training

We run our experiments on the Brown CS-grid and Google Colab. We use the environment setting of ConvNeXt as it is the basis of our code. Some key libraries in the environment are torch==1.12.1, torchvision==0.13.1, and timm==0.6.13. The devices used include Tesla T4, GTX Titan, and GTX 1080ti.

4. Results

Here are the results from different tasks of our experiments. The country classification results of different backbones are shown in Tab **??**. Out of all backbones chose, ConvNeXt-B performs best in training accuracy and testing accuracy. Bit-R50x1 has the highest score in top-5 accuracy. Notice that the numbers for training are not final since we stop training at the point where testing accuracy stops growing to avoid overfitting. The training curves are shown in Fig 7.

The latitude-longitude prediction results along with ablation of task-aware pretaining (retrain the backbone on country classification) is given in Tab **??**. The results are taken from the same epoch. Task-aware pretaining is quite useful in increasing converging speed and final performance. The training curves are shown in Fig 8. The graphic visualization for this is shown in Fig 9. Though the average error is quite high, the model can produce quite good predictions for some

Stage	Training		Testing	
Method	Acc@1	Acc@5	Acc@1	Acc@5
ConvNeXt-B	85.1%	97.9%	61.2%	86.6%
Bit-R50x1	63.4%	90.2%	59.0%	87.7%
BiT-R101x1	63.6%	91.0%	57.4%	86.3%

Method	error (degree)	loss	$\leq$5 on Eval
Original from Imagenet	8.8	63.6	24%
Pretrained on Countries	6.9	41.8	44%
best model(Original)	6.35	3.55	54%
best model(Pretrained)	4.89	19.72	60%

Method	error (degree)	Acc@1	Acc@5
Euclidean	4.56	5%	20%
Discriminator	2.93	11%	36%

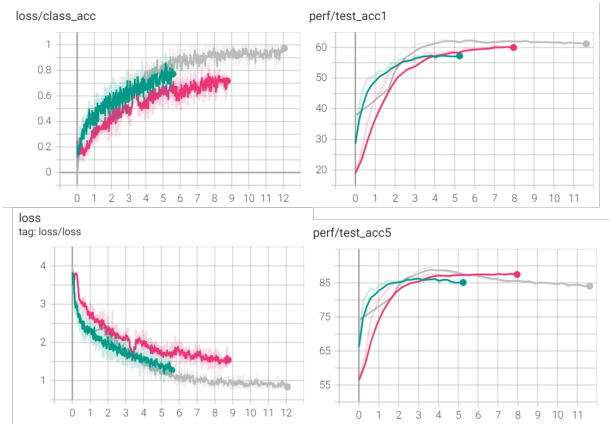

Figure 7: Tensorboard log for classification experiments.

samples. The positive samples (with a latitude-longitude error smaller than 5) occupy a little over 50% in the eval set. This is consistent with the country classification results.

The distance prediction results are shown in Tab **??**. A lot of different training settings were used for this task. Here only the best discriminator-based distance prediction is present. For computing the error values, 46 (number of countries in our data) anchors are used as reference points. The accuracy is measured by if the geographically closest anchor is the feature-space closest anchor. The discriminator performs better than Euclidean distance.

In the end, we tried our model on GeoGuessr to see the scores it can get in the game. Both country classification and latitude-longitude prediction models are used. The results are shown in Fig 10

4.1. Technical Discussion

The first step of using deep learning on GeoGuessr is country classification, based on the assumption that each country has unique visual features, but the noise within our

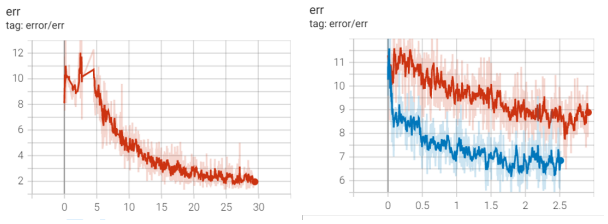

Figure 8: Tensorboard log for latitude-longitude prediction experiments.

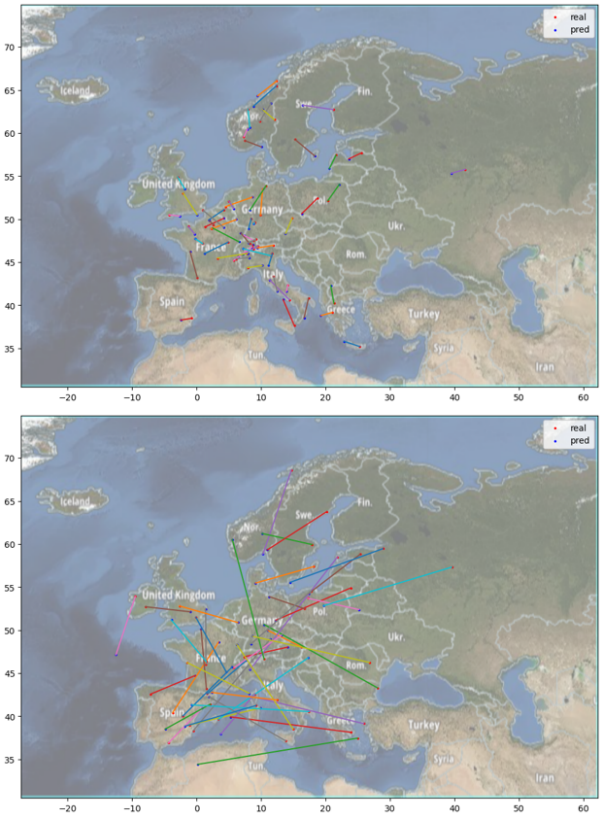

Figure 9: Latitude-longitude prediction results. Top:54 positive samples from eval set. Bottom: 46 negative samples from eval set.

data proves it more complicated. Even by the human eye, most street views, vegetation, and wild fields are indistinguishable. This thought is verified by the classification result of less than 20% accuracy.

Latitude-longitude prediction, on the other hand, is based on different hypotheses. We hope to build a continuous mapping from visual features to geolocations. The most direct way to do this is to create a network that remembers all the images covering every meter of the ground. But it is impossible due to high data collection costs and limited computing resources. As an alternative, we set our hope on creating

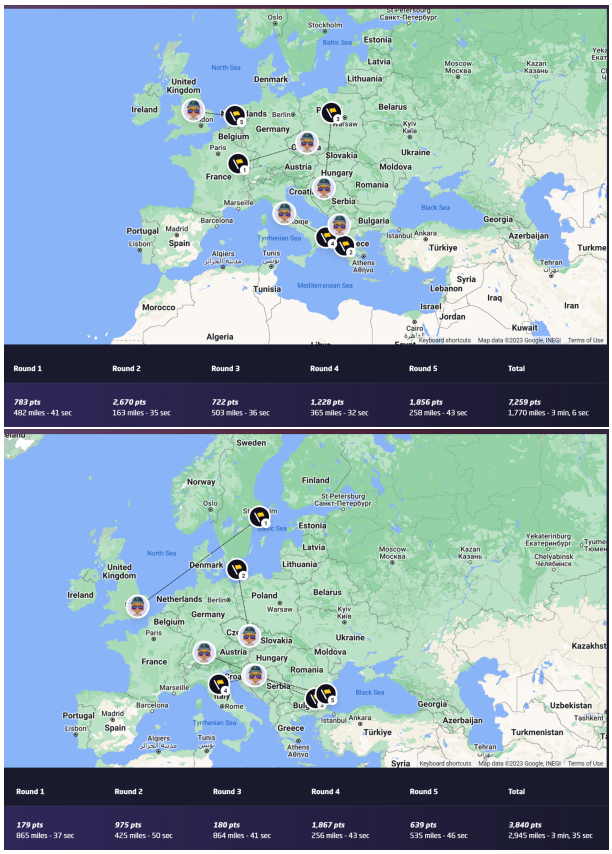

Figure 10: Running country classification model on GeoGuessr. Top: using classification model. Bottom: using latitude-longitude model.

a manifold in high-dimensional space to model the visual features we acquired and maintain the spatial relationship. That's to say, visual features that are close spatially should also be close in feature space. Even though we only have a small amount of data compared to the street views in the whole of Europe, the manifold could have good generalization ability in learned areas.

The latitude-longitude prediction has some interesting results, but unexpected. Based on our assumption, the model should only produce good results when the input is very close spatially to some trained samples. "Close" here refers to the situations where two images are taken in the same block or even just two sides of a building. To our surprise, no such cases were observed. It may be due to the sparsity of our data collection. The model should be able to handle very close samples, but there simply are none. On the contrary, a large portion of samples were predicted to be a few hundred kilometers away. This is still quite a success because of the difficult nature of the problem and the lack of data.

A deeper view of this phenomenon is the visual similarity at a large geological scale. A reasonable argument for this

is that similar building styles, ecological features, or geological patterns may reappear in an area with a scale of a few hundred kilometers. They could be in neighboring cities with similar cultures, one large ecoregion, or one large area sustained from consistent climatic and geological effects.

Based on this new insight into the problem, we explore another path of trying to capture the similarities between images. Obviously, there are no available labels on this from our data. The solution is to use geological distance as a guidance metric. CNN backbones used in previous experiments are still applied here to extract visual features. We tried two ways to model the similarity between features. One method to calculate the Euclidean distance between features. The model doesn't converge well this way since it is hard to tell if Euclidean distance is still appropriate in feature space. The other method is to construct a discriminator that takes in two image features as input and outputs their similarity. This second method brings up many training problems. It is hard to manage the supervision of distance during training. The model usually predicts all distances to be either zero or maximum values. In the end, both methods of distance learning don't work out.

The failure in distance learning gives us a better understanding of the problem. There is no consistent relationship between feature similarity and geographic distance. So for one image or a collection of visual features, it is easier to predict the most probable location of it than its distance from some reference points. In the end, the most effective way to solve this problem is still constructing a giant model to memorize visual features everywhere.

4.2. Socially-responsible Computing Discussion via Proposal Swap

The first concern of privacy and doxxing with a geolocating machine learning system is very valid, considering just how easy it would be to pass any image available on social media through the network. However, it is important to realize that our project is not designed to identify specific locations of individuals. We use computer vision to analyze and classify images based on different features, but our project does not store or share any personal information about any individual. That being said, we recognize the potential consequences and are committed to ensuring that our project does not contribute to the violation of privacy of any users.

We recognize that the second concern regarding unfairness in the game of GeoGuessr is a valid one as well. In games such as GeoGuessr, as popularity grows, the desire to cheat and gain advantage can also potentially grow. We definitely agree that this technology could undermine the integrity of the game, and thus we believe that the GeoGuessr team should implement measures to avoid external resources. It is worth noting that cheating in games is not a new phenomenon, and many successful games have implemented measures to combat this issue. Therefore, the GeoGuessr team can also draw on best practices from other popular games to help design measures that work effectively in the context of GeoGuessr.

Lastly, we agree with the third concern that the technology could decrease the educational aspect of GeoGuessr; however, it is important to note that the proposed technology is not intended to replace the educational aspect of the game but rather to enhance it. By employing interpretability methods that can explain the decisions of these kinds of models, they will be able to learn more about the culture, geography, and history of the places they visit, which will in turn help them to become more culturally literate. We believe that our project can be used as a learning tool for players to deepen their educational understanding.

5. Conclusion

Our project focused on training two machine learning models, ConvNext and BiT, to classify the location of an image. Specifically, we aimed to classify images by either country or coordinates– a challenging task as there exist a myriad of factors that could possibly influence the appearance of an image.

The successful development of image location classification technology has significant implications for a variety of fields ranging from geography, climate science, and photo forensics. Here, image location classification could help map out and better understand the world around us and aid in disaster response efforts, where it can be crucial to quickly identify areas that are affected by natural disasters. Within climate science, image location classification can help researchers track the spread of diseases or monitor changes in vegetation patterns over time. In photo forensics, the identification of the image's location could make a world's difference in cases involving missing persons.

Although they yielded promising results, our models would need to be tested against an evaluation set with more variety and to be examined at a more granular level to ensure that they are generalizable and meaningful features are being extracted from the images. Moving forward, possible ways to verify the two could be to utilize an evaluation set with more variety in image presentations (quality, lighting, etc. . .) and employ Grad-CAM class activation visualization, respectively. Nevertheless, our results reflect the advancement in CNN performance and with it, our ability to geotag images without any other information but the picture itself.

References

[1] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017. 4

[2] Josh Beal, Hao-Yu Wu, Dong Huk Park, Andrew Zhai, and Dmitry Kislyuk. Billion-scale pretraining with vision transformers for multi-task visual representations. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2022, Waikoloa, HI, USA, January 3-8, 2022*, pages 1431–1440. IEEE, 2022. 3

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2414–2423. IEEE Computer Society, 2016. 3

[4] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning, 2020. 2

[5] Nicholas I. Kolkin, Jason Salavon, and Gregory Shakhnarovich. Style transfer by relaxed optimal transport and self-similarity. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 10051–10060. Computer Vision Foundation / IEEE, 2019. 3

[6] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022. 2

[7] Eric Müller-Budack, Kader Pustu-Iren, and Ralph Ewerth. Geolocation estimation of photos using a hierarchical model and scene classification. In *Computer Vision – ECCV 2018*, pages 575–592. Springer International Publishing, 2018. 1

[8] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251. IEEE Computer Society, 2017. 4

Appendix

Team contributions

Manuel I helped with running the web scraper to scrape images from the Google Street View API. I also aided in the development of the notebook together with Praccho to train the BiT model on the classification task. This involved extensive testing and debugging, as well as careful analysis of the results to ensure that the model was performing. Furthermore, I helped in the design and production of the poster, helped in project milestones and writing reports throughout this past month.

Praccho I implemented the webscraping script, using Selenium and the Google Street View API, that gathered our data of images across Europe. I also altered the training code template to be compatible with BiT, as well as writing and running a notebook to train this model on the classification task. In addition to this, I'm also responsible for the introduction and methods sections in the final report.

Ziang I helped with running the scraper to download images. I designed the experiment pipeline and wrote the basic training code template, which was built on the foundation of ConvNeXt official repo. I ran the experiments on classification with ConvNeXt and made many attempts in distance and coordinate prediction. I am also responsible for the results and technical discussion in the final report.

Iris Throughout this project, I helped to oversee group meetings, outlining action items and upcoming deadlines. I helped to explore avenues for model interfacing such as the Huggingface open source library. In line with these efforts, I developed a script that uploaded our scraped data to Huggingface in the appropriate dataset form. I also conducted research on the significance of our project, related works, and future directions and aided in the design and production of the poster.