IMAGE GEO-LOCATION WITH VISUAL TRANSFORMERS

PROJECT REPORT

Tamás Csizmadia
Department of Computer Science
Aarhus University
au724747@uni.au.dk

Jędrzej Głowaczewski
Department of Computer Science
Aarhus, Denmark
au726831@uni.au.dk

December, 2023

ABSTRACT

This project report delves into the challenging domain of image geolocation, aiming to predict the location of an image solely based on its visual content. Employing Visual Transformers (ViT) instead of traditional Convolutional Neural Networks (CNN), the study explores the limitations of utilizing limited resources for training. The dataset, primarily sourced from Google Street View, focuses on Denmark, leveraging administrative boundaries for geocell classification. The research incorporates insights from related works, including PIGEON and TransLocator, while experimenting with data augmentation, learning rate adjustment, and a custom loss function to enhance model accuracy. Results indicate notable improvements in accuracy, yet the study identifies limitations and proposes avenues for future work, such as refining geocell generation, exploring continuous location prediction, and investigating multi-task network architectures.

1 Introduction

Image Geolocation, predicting the location of an image only from its visual content, is a very challenging task. A photo often includes only a few location cues, and the conditions in which it was captured can be very different. Additionally, location, being a continuous metric, presents a complex scenario for training neural networks and most papers we see out there use classification, where a class represents a geocell, a shape delimited by administrative, geographical or other boundaries. The most popular approach is to use Convolutional Neural Networks (CNN) to solve this task. However, in recent times, there has been a notable popularity increase of Vision Transformers (ViT) in image recognition [4]. Furthermore, the newest research on image geolocation shows that ViT models achieve higher accuracy than previous CNN models.

Since the task is complex, most of these attempts use high-end hardware to train models on millions of images, often for multiple days. Unfortunately, this is unreachable by an average university student. Hence, our goal is to train a network that performs well with the limited resources we have. We hope using a pre-trained ViT model will help us achieve results comparable to existing CNN attempts.

2 Related Work

There are many different publications regarding both planet-scaled and localized geolocation. However, the most innovative and insightful ones research the planet-scale geolocation. Hence, even though it is a slightly different domain, we will focus on them in this section.

The initial attempt at a planet-scale geolocation estimation of an image was undertaken by Hays and Efros (2008) [6], who introduced Im2GPS – algorithm using k-NN on a handcrafted set of features extracted from over six million GPS-tagged images. Weyand et al. (2016) [15] pioneered the application of CNN to geolocation by introducing the PlaNet model. This model predicts from over 26 thousand geographical cells, each constructed to contain a similar number of images out of 126 million in the training dataset. Müller-Budack et al. (2018) [11] achieved the best results among CNN models. They used Individual Scene Networks (ISNs) and exploited hierarchical (geo)information. In their

approach, scene probabilities are extracted for each photo using one network, and the image is passed to corresponding ISNs based on the scene predictions. The classification occurs across three spatial resolutions, and the most detailed subdivision prediction is enhanced using information derived from coarser representations. They trained the model on 4.72 million images.

TransLocator (Pramanick et al. 2022) [12] is the leading model for worldwide geo-localization, utilizing a ViT model, argued to be better suited for the task than CNN. The authors introduced a dual-branch transformer architecture, which in parallel predicts an image's geocell and environmental scene. They used the same train dataset as Müller-Budack [11] and achieved significantly higher accuracy.

Haas et al. (2023) [5] used CLIP [13] and similarly to us fine-tuned their model PIGEON on Google Street View images. They used only 100,000 locations, but unlike us and other mentioned papers, they decided to use four higher resolution images per location, resulting in a whole Street View panorama. Their approach to creating geocells was the inspiration for our algorithm - they used administrative boundaries, further divided into smaller regions with a clustering algorithm. Such a semantic construction of geocells has been proven to improve results (Theiner et al. 2022)[7], however, it was not utilized in other mentioned papers. Furthermore, they introduced a custom loss function based on distance, in contrast to the cross-entropy used in other studies. Izbicki et al. (2019) [10] also emphasized the importance of the distance metric. Hence, we included it in our experiments as well. In a separate publication, Haas et al. introduced StreetClip [8], a model trained on 1.1 million Street View images. It achieves performance comparable to the state-of-the-art on the IM2GPS3k dataset through zero-shot learning, showcasing its robustness to distribution shifts.

3 Methods

To train we used a Macbook Pro with the Apple M2 chipset, and we ran TensorFlow on the GPU with TensorFlow-Metal.

3.1 Dataset

First, we set out to gather the dataset. The papers we discussed above have some premade datasets that are publicly available, but these include indoor images, and are often focused on something irrelevant or small in size. We could not get our hands on the larger Creative Commons dataset that includes all kinds of pictures from Flickr. In the end, we decided that gathering our dataset would be the most useful. Google Street View [2] is a tool that has images of roads in almost all of the world. It is also the source of the PIGEON [5] and StreetCLIP [8] training datasets. Their API is not for free, but with the free credits that every user gets at the start of the month, we could download 40k images.

Regarding the issue of whether to train on the whole world or just Denmark, we decided that with our limited access to images, we will focus on Denmark only.

Since we can download only a small subset of Street View images, we needed to decide which of them we wanted to use. We do not have access to a list of locations with Street View imagery, but many publications mention that image density correlates to population density. Haas et al. [5] used this information when collecting train data for PIGEON and StreetClip, and this is also the approach we agreed to follow. Fortunately, Denmark is included in the OpenAddresses database [1], which we decided to use for selecting the locations of images to download.

The next step in the dataset preparation was to define the classes. We decided to use administrative boundaries since it has been shown to improve results [10, 5] when compared to rectangular cells. Hence, we utilized Municipality borders. We assume these boundaries evolved with time, accounting for geographical differences, but maybe cultural differences too, making the classes more distinct. It also ensures there is a population centre in each class, making the number of images per class fairly even. However, in Figure 1 we can see that Copenhagen and Aarhus have much higher numbers of addresses than the average. To combat this, we could limit the number of images per class to a specific number, but then our mission to generate accurate locations for images would be hindered as most pictures would likely come from these two cities. Our solution, partially inspired by PIGEON, was to do a K-means clustering algorithm on the largest clusters until we reached the desired number of classes. It resulted in 129 classes, we can call them geocells. We decided to use clustering to divide the cells, as it allowed us to retrieve different dense areas, helping to distinguish between parts of municipalities. Figure 2 shows how our geocells are laid out on a map of Denmark with locations as dots.

We chose K-means because it works well with two-dimensional data, and we were not specifically concerned with density. It is an efficient way to cluster data into a user-defined number of groups. In our case, it would have been beneficial to separate with a weight system where if one cluster is too small, the algorithm adjusts the clustering. Sadly, we did not have enough time to research this topic, so we decided on just using K-means.

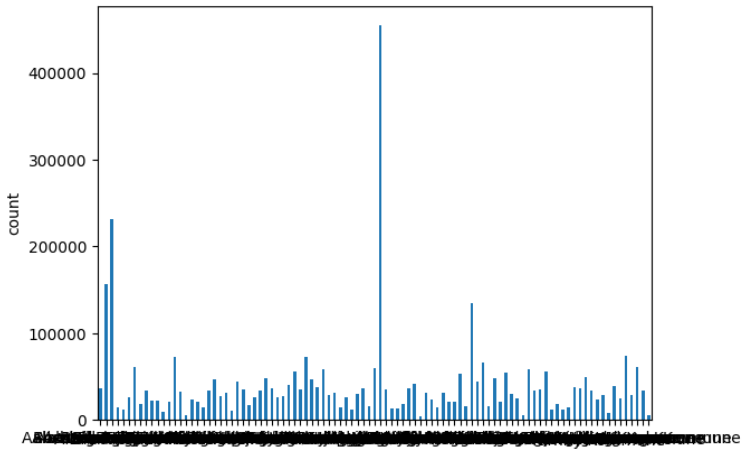

Figure 1: Locations within municipalities

To download the dataset, we sampled 42570 (330 for every geocell) addresses from the OpenAddresses database. To preserve the original distribution of the images within each geocell, we ensured to sample a proportional number of locations from smaller administrative units specified in the addresses. Street View API returns the closest image to a specified location, hence it may happen that the same image will be retrieved multiple times. We discarded such duplicates. The images were downloaded in slightly higher resolution and cropped to 224x224 to remove the logo.

3.2 Model

Our base model consists of the ViT layer, which was pre-trained by Google and uploaded to Hugging Face as "google/vit-base-patch16-224" and a classifier layer. Originally we used the default classifier which was by our understanding using linear activation. The loss function was Sparse Categorical Cross-Entropy. We decided to train our model for 20 epochs maximum since each epoch took around 1 hour. It was not convenient for us to allow it to run for a longer time, and we wanted to keep the number of epochs consistent among different iterations to allow for a fair comparison. Other hyperparameters were: batch size: 32, learning rate: 6e-5, and weight decay rate: 0.1. We used a Hugging Face optimizer that uses Adam with Polynomial decay. By default, we unfreezed all of ViT as this task differs considerably from the task it was trained on.

We can clearly see it is not perfect in Figure 3 and 4, it is overfitting. The loss function is showing signs of a high learning rate. So the first thing we tried was a lower learning rate, but it just resulted in very low accuracy.

The next improvement that we decided to explore was adding data augmentation. We reviewed previously mentioned papers on data augmentation techniques but found they lacked justification for the chosen policies. Consequently, we expanded our search to more general papers focusing on data augmentation for ViT. Drawing from the insights of Steiner et al. (2022) [14], we adopted RandAugment by Cubuk et al. (2019) [3]. This method involves applying a number of augmentation layers (l) with specified magnitude (m). Through experimentation, we settled on l=2 and m=0.15, noting that the implementation in keras_cv offers a different magnitude range from the original paper. We decided not to use image flipping, as it did not make sense in our case since the streetscape is mostly not symmetrical. Data augmentation improved our accuracy from 29% to 35%.

Another way we tried to improve accuracy was to make the learning rate drop faster. We did this by adjusting Adam's beta1 and beta2 parameters. This only helped 2 percentage points.

We also tried training a model that had a frozen ViT, but it achieved 10% accuracy even after tuning hyperparameters.

3.3 Distance

One thing our model did not account for is distance. We hoped that if we incorporated it into the loss function, the model would learn the differences between larger regions than our classes, so even if the predicted geocell is not correct, it is closer than a randomly selected one. We agreed to use Haversine distance since it takes into account the earth's spherical geometry and was also successfully used by Haas et al. [5, 8]. We started with computing the Haversine distance between the true and the predicted geocell centroids. We expected it would teach the model that the closer

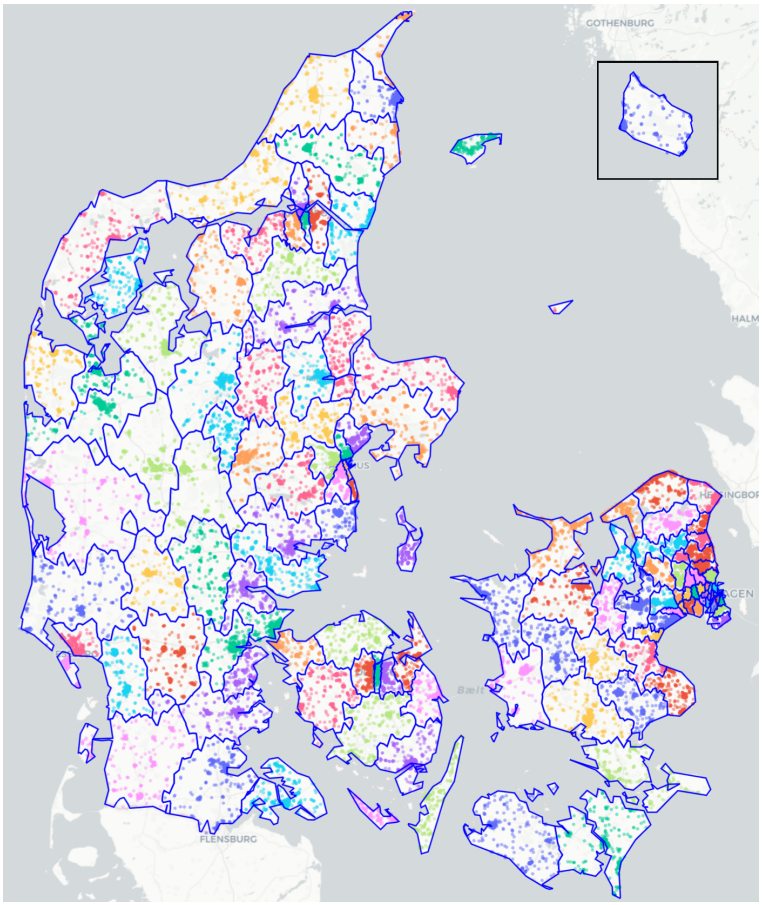

Figure 2: Geocells in Denmark visualized with Voronoi diagram. Voronoi cells of the same cluster are grouped together

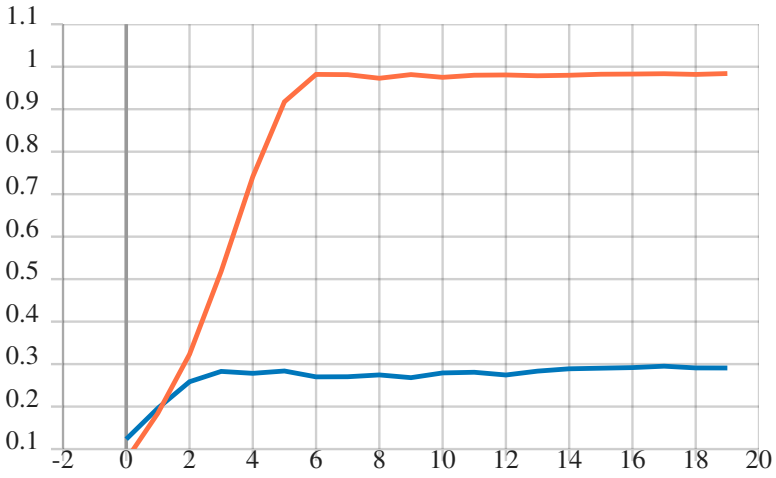

Figure 3: Base Model accuracy (orange - training, blue - validation)

it gets to the true location the better. Unfortunately, this is not a continuous function as it switches between different constant values, making it unfit to use this as the loss. The other way was to multiply the distances of all predicted cells and the true cell with their probabilities.

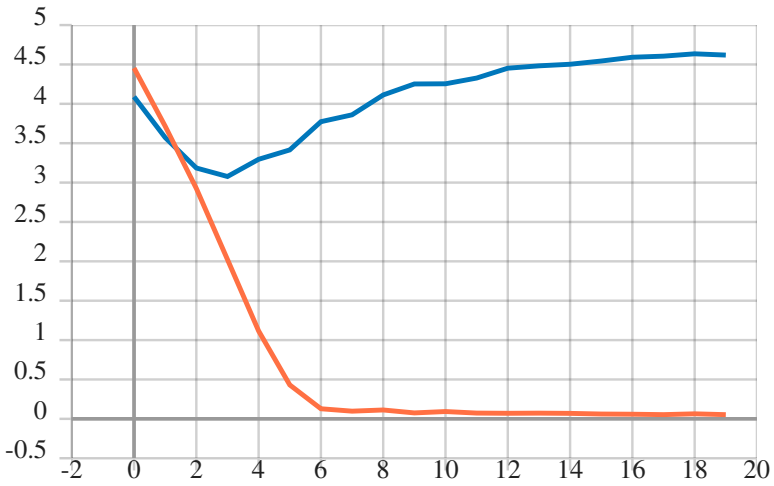

Figure 4: Base Model loss (orange - training, blue - validation)

$$Loss_{dist} = (\sum_{i=1}^{n} p_i * dist(geocell_i, geocell_{true}))^2 \tag{1}$$

Sadly this did not result in a better accuracy, as we hoped, but it did improve the average distance between predicted and true geocells. To keep a good accuracy but improve the distance between cells we decided to sum the cross-entropy loss with this distance loss. We also squared the distance loss so that it learns distance even faster.

$$Loss = Loss_{dist} + Loss_{categorical_cross_entropy} \tag{2}$$

This led to better results, but after consideration, we ended up redefining the distance loss. This tries to help learning by putting the highest weight on the highest probability.

$$Loss_{dist} = \sum_{i=1}^{n} -log(1 - p_i) * dist(geocell_i, geocell_{true}) \tag{3}$$

Truthfully, this did not help much either. To be able to do all of this, we changed the default classifier to have softmax as activation. We also decreased the batch size to 16, as the new custom model used more memory.

4 Results

Our results show how hard it is to geolocate images, even with newer models. Data augmentation proved to be the biggest influence on our results. As our base model was overfitting very heavily, this is not surprising. We improved our accuracy by 8 percentage points compared to the base model. As a result, the final accuracy was 37%, which is not as much as we hoped. Nevertheless, we find it quite impressive, since we (inexperienced with the geolocation of images) could not distinguish between different locations. Moreover, Denmark is a small country with a relatively low architecture and landscape diversity, which makes the task more difficult than planet-scale geolocation.

Our results are not comparable with the related papers as we only focused on Denmark. Still, if we want to compare, PIGEON[5] had an accuracy of 40% on city-level estimation (25km) which is comparable to our municipality-level estimation. It had a mean km error of 251, which is much higher than our model (best: 62km). These comparisons are unfair though, as our starting mean km error was 150 km when randomly choosing predicted cells. Therefore we do not argue our model is better, but we can also note that the world has much higher diversity than Denmark, which might have "helped" PIGEON.

In Figure 8 we can see the class confusion matrix. Our takeaways are, that the smaller the class, the better the accuracy. Small islands that constitute as separate municipalities performed better, which was expected. Cities divided into sections are smaller in area than the average, but their accuracy is also better. It is intriguing because different regions of a city would seem to have less diversity between themselves than separate cities. Hence, one would expect a class to be fuzzy, but we think the small size of geocells accelerated the learning process. Classes that were neighbouring

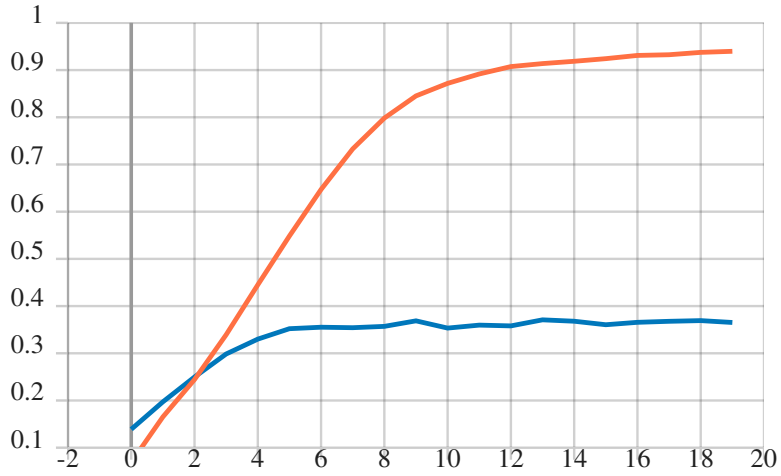

Figure 5: Accuracy of the last model (orange - training, blue - validation)

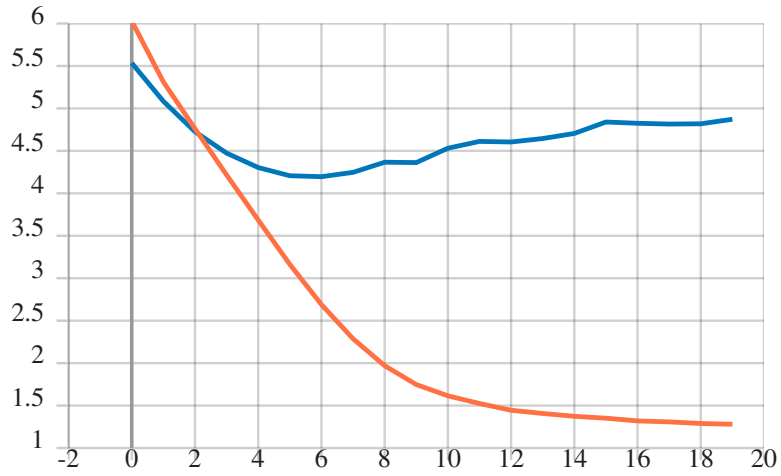

Figure 6: Loss of the last model (orange - training, blue - validation)

each other often had higher confusion - this is good as this shows that it is learning areas of the map, not just randomly choosing.

The model is learning something about these classes, and it is better, the smaller a class is. The question remains, what does it actually learn? From figure 9 we can see that it specifically focuses on the sky, the shape of the clouds, etc. This is unexpected, as clouds move fast, the sky changes a lot even during one day. One thing to note though, is that the StreetView cars probably traverse small regions in short amounts of time. This means that the clouds might be similar in all of the pictures within a class. This could also mean that it will not be able to generalize to new StreetView imagery if Google decides to update its photos.

5 Discussion

We set out to achieve similar results as PIGEON and similar papers but on less data and cheaper hardware. We have achieved results that in some metrics are comparable, but our expectations for accuracy were much higher. Data augmentation worked like we expected it to, learning rate scheduling helped a little, but the custom loss function turned out to be a letdown. As we expected freezing the pre-trained model did not help learning as the task was different enough.

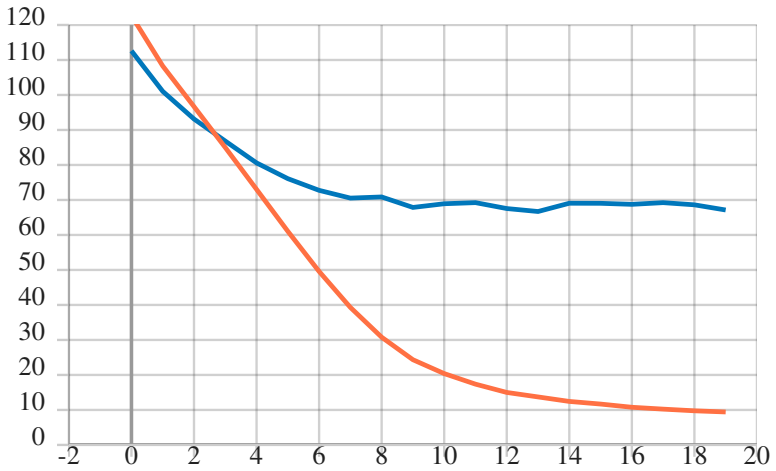

Figure 7: Distance of the last model (orange - training, blue - validation)

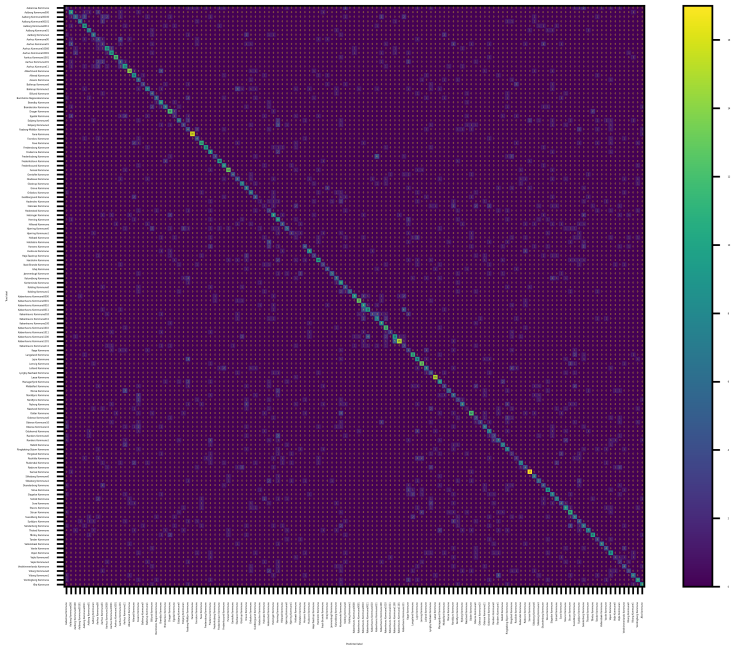

Figure 8: Confusion matrix

5.1 Limitations

The primary limitation of our work is the reliance on Street View images, constraining the analyzed area to more populated regions, and hence, neglecting rural areas. Furthermore, our dataset, derived from a highly specific distribution, may not generalize well in many other scenarios (e.g. hand-taken photos, photos of specific objects, images taken inside buildings, or even Street View photos taken on a different day). Secondly, our method for generating geocells, though quite effective, is not ideal. At the time of writing this report, we visualized the Municipality Boundaries used for class assignments and noted their significant imprecision, often resulting in overlapping. Figure 2 presents real borders between classes computed using the Voronoi diagram, revealing that fortunately, the borders between prepared classes are relatively clean. Nevertheless, this imprecision may have impacted our goal of employing semantically meaningful borders. Additionally, exploring alternative clustering techniques could prove beneficial, allowing for more complex decision boundaries than the linear ones provided by K-means, and possibly resulting in a more even distribution of images in geocells.

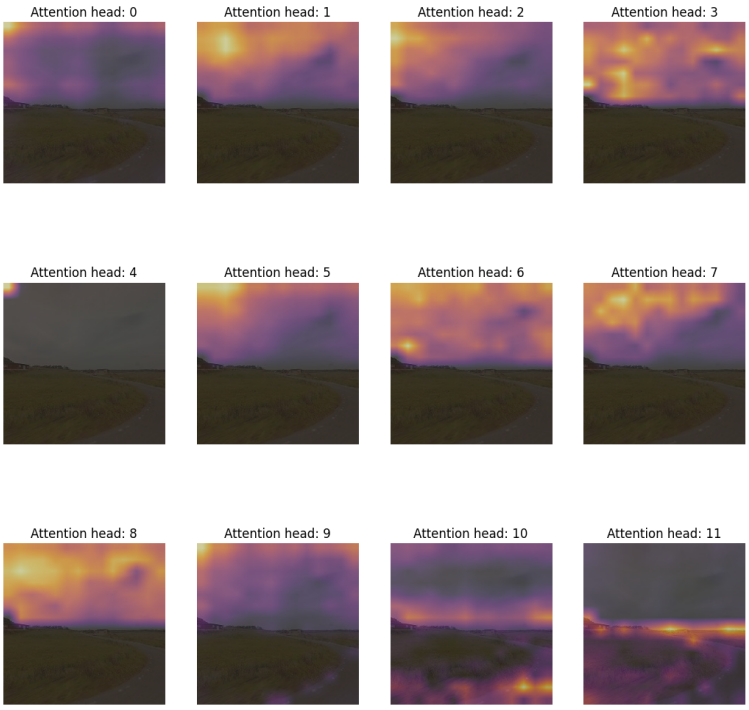

Figure 9: Attention heatmap

5.2 Future Work

Our results are generally satisfactory. However, several options for further investigation were left unexplored due to time constraints.

Incorporating a distance metric into the loss function requires further analysis, as we believe that cross-entropy could be removed from the loss function to prioritize distance over accuracy in predictions. Additionally, image location should be used instead of the geocell centre when computing the distance metric [5]. Another possibility is updating the decoder to return the image location continuously, rather than a single geocell. Our idea is that a few most probable predictions can be combined into one pair of coordinates by computing the weighted average of predicted locations. This way, the final prediction would be placed inside a polygon determined by the centres of the most probable geocells. This approach is especially interesting, as we have not found any papers exploring it.

Further architecture improvements could include using scene-specific classifiers (ISNs [11]) or adopting a more generic approach - predicting environmental scenes in a multi-task network [12]. Another inspiring concept from Müller-Budack et al. involves exploiting hierarchical (geo)information [11], which significantly improved the accuracy of their model.

Another possible direction of research is to further explore the findings of Haas et al. [5, 8] and Luo et al. [9] regarding multi-modal vision transformers. They used CLIP and showed accuracy improvement when enriching the training data with image descriptions [5, 8] or a geolocation guidebook [9] containing instructions on how to recognize a country from an image.

Moreover, while our focus has been on ViT, a comparative study involving CNN architectures like ResNet could offer valuable insights.

References

[1] Openaddresses website.

[2] Streetview api website.

[3] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

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

[5] Lukas Haas, Michal Skreta, and Silas Alberti. Pigeon: Predicting image geolocations, 2023.

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

[7] Ralph Ewerth Jonas Theiner, Eric Müller-Budack. Interpretable semantic photo geolocation. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2018.

[8] Michal Skreta Lukas Haas, Silas Alberti. Learning generalized zero-shot learners for open-domain image geolocalization, 2023.

[9] Grace Luo, Giscard Biamby, Trevor Darrell† Daniel Fried, and Anna Rohrbach. G3: Geolocation via guidebook grounding, 2022.

[10] Evangelos E. Papalexakis Mike Izbicki and Vassilis J. Tsotras. Exploiting the earth's spherical geometry to geolocate images. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019.

[11] Eric Muller-Budack, Kader Pustu-Iren, and Ralph Ewerth. Geolocation estimation of photos using a hierarchical model and scene classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[12] Shraman Pramanick, Ewa M. Nowara, Joshua Gleason, Carlos D. Castillo, and Rama Chellappa. Where in the world is this image? transformer-based geo-localization in the wild, 2022.

[13] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[14] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers, 2022.

[15] Tobias Weyand, Ilya Kostrikov, and James Philbin. *PlaNet - Photo Geolocation with Convolutional Neural Networks*, page 37–55. Springer International Publishing, 2016.