

School of Computing

FACULTY OF ENGINEERING AND
PHYSICAL SCIENCES



UNIVERSITY OF LEEDS

Final Report

A Privacy-Preserving AI Approach to Predicting Geographical Locations from Images

George Harding

**Submitted in accordance with the requirements for the degree of
BSc Computer Science**

2024/25

COMP3931 Individual Project

The candidate confirms that the following have been submitted:

Items	Format	Recipient(s) and Date
<i>Final Report</i>	<i>PDF file</i>	<i>Uploaded to Minerva (30/04/25)</i>
<i>Link to online code repository</i>	<i>URL</i>	<i>Sent to supervisor and assessor (29/04/25)</i>
<i>Demonstration Video</i>	<i>URL</i>	<i>Sent to supervisor and assessor (29/04/25)</i>

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) George Harding

Summary

This report investigates the creation of a privacy-preserving image geolocator capable of predicting the location of an image in the UK. With the rise of highly accurate geolocators built using powerful Large Vision Language Models (LVLMs), there are major privacy concerns due to the model's ability at identifying location-specific details such as landmarks and signs. The project addresses this by building and training an LVLM from scratch, employing privacy-preserving techniques to analyse their effectiveness on mitigating a geolocator's privacy risk. A balance must be found between accuracy and privacy, as geolocators are key tools with many positive uses.

This project successfully produced a functional baseline LVLM geolocator, trained with street-view images in the UK, alongside a privacy-preserving version utilizing two privacy techniques: blurring text in images during training to make the model inherently ignore sensitive text, and secondly increasing the classification size to encourage broader predictions. The results demonstrate that a geolocator can effectively reduce its pinpoint accuracy, while still retaining broad accuracy to maintain its utility as a tool. These findings contribute to ethical AI development, designing a system that protects privacy while still being useful.

Acknowledgements

Firstly, I would like to thank my project supervisor, Professor Kristina Vušković, for her constant support and guidance throughout my project. The regular feedback and discussions were integral to shape my work. Even with the fact that she didn't specialise in the machine learning field, she always put in great effort to support me and the group.

I am also grateful to my assessor, Dr Haiko Müller, for his valuable feedback on my initial project draft and my working implementation. Also, the interesting discussion during our assessor meeting really helped direct and inspire a lot of the ideas in my report.

Special thanks to Dr Samson Fabiyi, specialising in machine learning, who generously offered to help me with my machine learning topic, despite not being assigned to my project. His advice and feedback on my project outline helped me massively at the start to set a strong foundation.

I also extend my appreciation to all the open-source libraries and platforms I used during my implementation. In particular, Mapillary for making their huge image dataset available for research purposes, HuggingFace for their accessible implementations of BLIP-2, BERT and ViT, and finally PyTorch, for providing the tools and support to allow me to build and train my own machine learning models.

Table of Contents

Summary	iii
Acknowledgements	iv
Table of Contents	v
Chapter 1 Introduction and Background Research	1
1.1 Introduction to Image-based Geolocation	1
1.2 Privacy Concerns.....	1
1.3 Motivation	1
1.4 Project Aim, Objectives and Deliverables	2
1.5 Background Research	3
1.5.1 Large Vision Language Model (LVLM)	3
1.5.2 Multilayer Perceptron (MLP)	4
1.5.3 Previous Work on Image-based Geolocation.....	5
1.5.4 Previous Work on Privacy Preservation.....	5
Chapter 2 Planning and Methodology	7
2.1 Dataset	7
2.1.1 Data Collection	7
2.1.2 Data Cleaning.....	8
2.2 Design	8
2.2.1 Geocells	8
2.2.2 Baseline LVLM	8
2.2.3 Neural Network Design (MLP)	9
2.2.4 Privacy-Preserving LVLM	10
2.2.5 Training the MLP	11
2.2.5.1 Cross Entropy Loss	11
2.2.5.2 Haversine Loss.....	11
2.2.5.3 Validation Set	12
2.3 Risk Analysis	12
2.4 Environments.....	13
2.5 Development Approach: Prototyping	13
2.6 Version Control.....	13
Chapter 3 Implementation and Validation	14
3.1 Code Design and Standards	14
3.2 Collecting Data	14

3.3	Geocells	14
3.4	Preprocessing.....	15
3.4.1	Image Captions	15
3.4.1.1	Caption Generation	15
3.4.1.2	Text OCR Reader and Cleaning	16
3.4.2	Text Encoding	16
3.4.3	Vision Encoding.....	17
3.4.4	Fused Encoding	17
3.4.5	Blurring Images	17
3.4.6	Saving Data.....	18
3.5	Neural Network (MLP)	18
3.5.1	Feature Extraction (Hidden Layers)	18
3.5.2	Classifier	18
3.5.3	Regressor.....	19
3.6	Training	19
3.6.1	Loss Functions	19
3.6.1.1	Cross Entropy Loss	20
3.6.1.2	Haversine Loss.....	20
3.6.2	Gradient Calculation	20
3.6.3	Optimizer	20
3.6.4	Scheduler	21
3.6.5	Validation Set	22
3.6.6	Batches	22
3.7	Using the Models.....	22
3.8	Validation.....	23
3.9	Testing.....	23
Chapter 4	Results and Evaluation	24
4.1	Overview	24
4.2	Qualitative Analysis	24
4.2.1	Overall Performance Across Models	24
4.2.2	Case Study: Alpine Guest House	25
4.3	Quantitative Analysis	27
4.3.1	Baseline LVLM	27
4.3.2	Privacy Technique 1: Blurring Images	27
4.3.3	Privacy Technique 2: Larger Geocell Size	28

4.3.4 Combination of Both Techniques: Privacy-Preserving LVLM	28
4.4 Final Discussion: Balancing Privacy and Utility	29
Chapter 5 Conclusions	30
5.1 Conclusions	30
5.2 Future work.....	30
List of References	31
Appendix A Self-appraisal.....	37
A.1 Critical Self-evaluation	37
A.2 Personal Reflection and Lessons Learned.....	37
A.3 Legal, Social, Ethical and Professional Issues	38
A.3.1 Legal Issues	38
A.3.2 Social Issues.....	38
A.3.3 Ethical Issues	39
A.3.4 Professional Issues	39
Appendix B External Materials.....	41
B.1 Software and Python Libraries	41
B.2 Pretrained Models	41
B.3 APIs	41
B.4 Tools.....	41
B.4 Online Resources (guides, discussion forums, blogs)	41
Appendix C Code Attribution	42
C.1 Python Libraries:	42
C.2 Online Resources:	42
Appendix D Image Examples	45
D.1 Geolocated Images:.....	45
D.1 Non-geolocated Images:	46

Chapter 1

Introduction and Background Research

1.1 Introduction to Image-based Geolocation

Geolocation involves determining the location of a person or object in the world. Tools and technology known as geolocators are frequently used to get exact geospatial coordinates or even predict them [24]. This report is going to be looking specifically into image-based geolocation [31], also known as image geo-localization [53], where the goal is to get geospatial coordinates from just an image. For example, finding a way to get the latitude and longitude of a bridge in a picture. Without metadata, there is no way to explicitly fetch this information, so machine learning based algorithms need to be used to predict the location like a human would.

Image-based geolocation has many interesting uses, ranging from identifying the location of old family photos in a generations-old photo album, to locating crime scenes from criminal photographs [34]. The list continues with environmental surveying over time, autonomous driving, navigation, geography education, and investigations in journalism [23].

1.2 Privacy Concerns

However, there are major privacy concerns with such technology. One's location is a sensitive piece of personal information which is often overlooked, especially in today's digital age with people posting so much on social media. A geocator could be used for privacy violating activities such as stalking, theft, abduction, surveillance, military usage, and grey corporate usage [7,23]. Companies could collect user locations without the user's explicit permission, for targeted ads or data collection as examples. Some groups are particularly vulnerable such as children, as their location can be found from any images they post, especially when they have little regard for obscuring location-identifying features in their images [26,46]. We should fear a world where AI can easily find our location with just a photo, yet this is a reality which is rapidly approaching.

1.3 Motivation

In recent years, there has been lots of investment into perfecting a machine learning algorithm for image geolocation, which is concerning. As observed in [23], every state-of-the-art paper

on image geolocation over the past 5 years was funded by military contracts, such as the Department of Defence and the US Army. They note that military usage of this technology should come under particularly scrutiny. Ethics and privacy are not of major concern in these military-funded papers, only improving the technology.

GeoSpy [22] was a publicly available geolocator tool which was marvelled on release, being particularly good at predicting locations from images in the US [43]. However, public access ended up being revoked by the founder due to privacy and security concerns, especially stalking issues [10]. GeoSpy is now marketed as a tool just for Government and Law Enforcement [22], which was its original intention. This was very eye-opening for the negative potential of geolocators, and raises questions of responsible usage - who should be able to access such a privacy-violating tool?

Finally in 2024, a geolocation model named “Ethan” was developed [31], employing a Large Vision Language Model (LVLM) with alarming accuracy. LVLMs are a type of machine learning model that are extremely good at using identifiable features to pinpoint exact locations, combining a textual encoding with a visual encoding from the image. The model Ethan was not released publicly due to privacy concerns, with the authors calling for immediate investment and research into responsible AI development to mitigate the privacy risks of such a geolocator. They briefly explored the theory for a privacy-preserving version of their model, but it has not yet been implemented.

1.4 Project Aim, Objectives and Deliverables

The aim of this project is to develop a machine learning tool to predict the location of an image in the UK, all while retaining privacy, by exploring and utilising a privacy-preserving LVLM. The performance and accuracy of a model designed with privacy at its core will be evaluated and compared to a baseline LVLM model trained normally to prioritise accuracy. A balance must be struck between privacy and accuracy, still allowing the model to be useful for positive geolocation tasks. Is a privacy-preserving geolocator just a *bad* geolocator?

The objectives are as follows:

- Research methods into which a geolocator could preserve privacy.
- Prepare and preprocess a dataset of images in the UK with captions.
- Create a baseline LVLM and a privacy-preserving LVLM in Python.
- Train and fine-tune the LVLMs with a training set and validation set.
- Develop a method to interact with the geolocators to compare.
- Analyse the performance of the models in geolocating unseen images.

- Evaluate the trade-off between accuracy and privacy preservation for the models. Does the privacy-preserving LVLM address the privacy concerns of geolocators?

The deliverables are as follows:

- The GitHub software repository used to create and use the LVLMs.
- Two datasets of the cleaned preprocessed data used for training and validation (.pkl files), one being the blurred version.
- A baseline LVLM geolocator (.pth file).
- A privacy-preserving LVLM geolocator (.pth file).
- Two sanity LVLMs employing only one privacy-preserving technique (.pth files)
- A method to upload an image and predict its location with the models.
- A project report discussing the findings.

1.5 Background Research

1.5.1 Large Vision Language Model (LVLM)

An LVLM is a large-scale multi-modal machine learning model, which processes textual and visual data together. They stem from the success of Large Language Models (LLMs) in textual tasks, where an additional vision encoder was introduced for visual tasks [50]. LVLMs have seen high success in complex multi-modal tasks such as object recognition [8], scene interpretation [12], and content moderation [29], as reported in [31]. Illustrated in Figure 1, LVLMs use a vision model to extract features from the image, and a language model for the text. These two encodings are then fused together with approaches such as cross-attention, a method which links the visual and textual data from each source to find connections [11]. The model can then feed the fused encoding into a neural network for machine learning heads such as generation, classification, and regression.

A “geolocation-rich” [31] image with identifiable landmarks, geographic features, street signs, and infrastructure are all very vulnerable to be pinpointed by an LVLM geolocator over single-modal (or “unimodal”) alternatives. The LVLM would be provided with an image of a location and a caption describing it, for example, a picture of a red-arched bridge with the caption “a Japanese bridge” [31]. The fused encoding generated from both visual and textual data will contain key connections and associations of text with features, allowing the model to easily learn relationships just from the training data without any extra knowledge or pretraining. In the example above, the LVLM will easily associate the image of a red arched bridge with the word “Japan” – it now knows what a Japanese bridge looks like. This is a powerful advantage

over single-modal machine learning models which can miss context and connections with only one data source [4]. Fused encodings are powerful inputs, allowing LVLMs to build strong understandings of the relationship between visual and textual data very well.

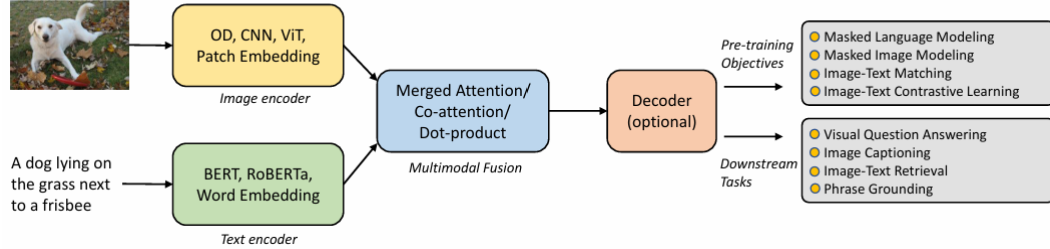


Figure 1: General framework of Vision Language Models from [16]

1.5.2 Multilayer Perceptron (MLP)

An MLP is a type of neural network used in deep learning, characterised by its multiple layers of neurons [1,56]. An MLP has an input layer, followed by hidden layers, and finally its output layer. Figure 2 displays a simple MLP with one hidden layer. Each hidden layer applies a linear transformation on the input using weights and biases, also known as its parameters. These parameters are key for the network to learn and are iteratively adjusted during training with optimisation algorithms such as gradient descent [37]. Each hidden layer in the network has a non-linear activation function applied after its linear transformation. The purpose of an activation function is to introduce non-linearity – without it, the entire network would simply act as a linear regression model applying a singular linear function to the input [3]. Activation functions are key for the network to capture complex, non-linear patterns within the data.

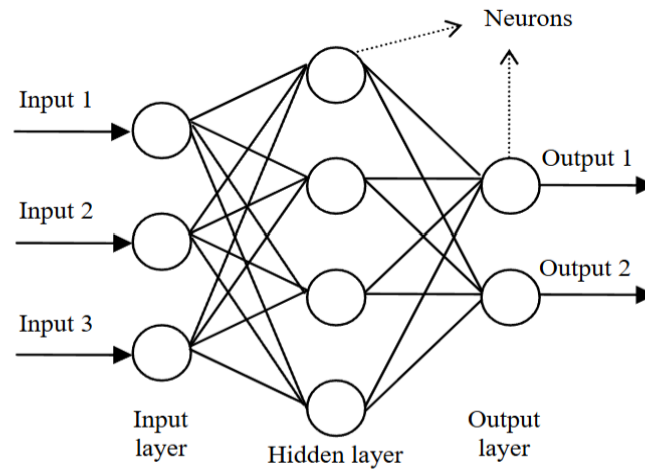


Figure 2: MLP Diagram from [48]

1.5.3 Previous Work on Image-based Geolocation

Two image-based geolocation papers have seen groundbreaking success over the past year, creating “PIGEON” [23] and “Ethan” [31]. These innovative models are some of the first to be able to generalize well on unseen locations anywhere in the world. Firstly, PIGEON, a forerunner to Ethan, managed an average distance of 251.6 km and successfully predicted 5.36% of images within 1 km [23], as shown from Figure 3. The model has a key focus on the classification task of splitting the world semantically into “geocells” with administrative boundaries, clustering, and Voronoi tessellation. Since “it was difficult for deep learning models to directly predict geographic coordinates via regression” [23], this breakthrough allowed CNNs and vision transformers to be revisited for use in geolocators. Synthetic captions were also generated to help aid the visual encoding used in the model, allowing it to capture geospatial cues more effectively.

Ethan is an extremely accurate geolocator, built using an LVLM and a similar geocell method to PIGEON. It has an average distance of 105 km and manages to predict 27% of images within 1 km, a major improvement from the previous paper PIGEON and the commercial tool GeoSpy [22]. Ethan extracts the key visual and textual features into embeddings, but instead of then classifying the fused encodings directly with a neural network, a clever chain-of-thought reasoning is employed to refine predictions. All these factors help the model achieve such accuracy, but arguably the main improvement is due to the usage of an LVLM.

	Average distance (km)	Within 1 km (%)
PIGEON	251.6	5.36
GeoSpy	110.3	25.5
Ethan	105	27

Figure 3: Accuracy of notable image-based geolocators [22,23,31]

1.5.4 Previous Work on Privacy Preservation

Most papers do not address privacy preservation in detail, concerning themselves mainly with the accuracy of the model. Those who do mention ethical and privacy concerns do so briefly, calling for somebody else to look into addressing the privacy concerns.

The paper presenting Ethan responsibly delves into the privacy concerns of their geolocator and suggests how they can be addressed [31]. As LVLMs pick up on location-specific details,

a mitigation would be to detect and obscure this information, preventing a model from predicting a photo's location to protect users. For example, when posting a picture on social media, a bridge in the background is easily detected by the model, and so you should obscure the bridge if you wish to not be located when posting such a picture. This approach is simple but would require such a tool to be employed by all social media platforms or their users individually. Additionally, educating users about cautious sharing and the dangers of posting "geolocation-rich" photos online is important, yet with human variance or negligence this mitigation is not at all ideal.

More interestingly, they briefly explore how the geolocator itself could be made to preserve privacy with two methods, creating a "privacy-preserving LVLM". Firstly, the geolocator could add some random noise to its final prediction to prevent pinpoint estimations. Secondly, altering the actual LVLM to inherently ignore location-specific details. This could be done by blurring all location-specific details in the images used for training, or perhaps the loss function can be altered to punish using them, so it cannot pick up on these details at all. As mentioned before, none of these privacy mitigations have been explored thoroughly or implemented yet. Creating a privacy-preserving LVLM will be a good example of responsible AI development. Although, other people could always make their own privacy-violating LVLM and demean the efforts put into a privacy-preserving version – particularly if the model is not as accurate or useful. Legal measures may be required to regulate and ban geolocators in the future to protect our privacy.

Chapter 2

Planning and Methodology

2.1 Dataset

2.1.1 Data Collection

To effectively train the model, a very large dataset of images in the UK is required, as every part of the country needs to be represented in the dataset. For enough coverage and image diversity, a range from 20,000 to 50,000 random images was deemed sufficient, with more the better. Mapillary [36] was the perfect choice, offering free and open-source access to billions of user-uploaded street-view images in the world, with very high coverage of the UK. The dataset is approved for research-usage and model training, licensed under [CC-BY-SA](#).

To access images from Mapillary, the API [35] is used by sending a bounding box, which returns a random selection of data entries back from within the selected areas. Two bounding boxes were defined, aiming to cover all the UK without clipping any of Europe, as shown in Figure 4. The database contains lots of key information, most importantly an image in multiple resolutions and the geospatial coordinates it was taken at. For this project, the image in 2k resolution alongside its coordinates are retrieved to create the dataset. Using higher resolution images to train will make processing longer but improve the model's quality, especially for text detection.



Figure 4: Bounding boxes for Mapillary query. Adapted from [21].



Figure 5: Example grid of geocells. Each cell is a category. Adapted from [21]

Map data ©2025 GeoBasis-DE/BKG (©2009), Google, Inst. Geogr. Nacional.

2.1.2 Data Cleaning

The Mapillary dataset is crowdsourced, introducing many challenges regarding data integrity. There are many erroneous entries which don't have a 2k image resolution or are missing coordinates. Additionally, some images include metadata watermarked on the bottom of the image, such as the date taken, coordinates of image, and the user who uploaded it. All this data could unintentionally affect the model training and should not be considered. Therefore, these images need to be cleaned, and this text filtered out when processing.

2.2 Design

2.2.1 Geocells

Geolocation should be primarily treated as a classification task. Regressing coordinates directly has been proven ineffective, as outlined in [23] and the findings later in Section 2.5. Instead, the map is split up into cells, geocells, each representing a different category. These geocells can range from a tiny square to an entire county. Using semantic borders such as counties could provide very meaningful divisions and avoids problems such as cutting similar locations in half. However, a simple grid approach is used, involving splitting the map into small squares using a grid pattern. This method simplifies the process and allows for easy manipulation of the category sizes. Each cell in the grid is assigned an ID, which is its category label. Figure 5 displays this simply, with very large grid sizes.

2.2.2 Baseline LVLM

The data flow of the baseline LVLM is depicted in Figure 6. Firstly, an image is inputted into the model and a descriptive caption is generated by a separate third-party AI model. Captions are automatically generated because it would be unfeasible to manually create captions for every image in the dataset. The caption is supplemented with any text detected on the image, such as street names.

The caption and image are used to generate text and visual encodings individually, which are then fused together with cross-attention, and finally fed into a trained neural network. An MLP is used to classify the image into a geocell of size $0.05^\circ \times 0.05^\circ$, as well as predict the latitude and longitude offset within the geocell. The final coordinate prediction is calculated as outlined in Equation (1) using these two outputs.

This baseline model is designed for pinpoint accuracy, ensuring it picks up on all location-identifying features such as street names, and utilises small geocells for precise classification.

$$\text{Coordinate Prediction} = \text{Geocell centre} + \text{offset} \quad (1)$$

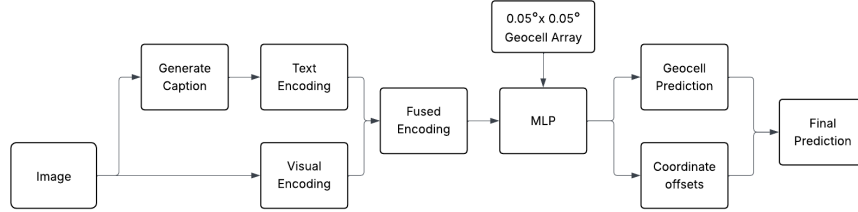


Figure 6: Baseline LVLM. Created with [33].

2.2.3 Neural Network Design (MLP)

An MLP was chosen for the neural network implementation, as they are widely established for delivering accurate and efficient performance in classification and regression tasks [37]. Alternative designs such as the single-layer perceptron is outperformed since MLPs have more hidden layers and neurons, allowing for more complex patterns to be learned and increasing the capacity for the number of features it can capture. CNNs and Transformers were also viable options, but MLPs are simpler and more computationally efficient [51].

Figure 7 shows the MLP architecture. The design utilises three hidden layers to allow the model to progressively learn complex features from the fused input. The fused encoding is scaled to 1024 dimensions, 512, and 256 for each layer respectively with decreasing dimensionality, to refine and extract features. Once the patterns have been learned in the final layer, the MLP has two built in heads to classify and regress the learned features into outputs. The classification head predicts the geocell and contains neurons equal to the total number of geocells, representing all possible classes. The regression head predicts the coordinate offset by scaling the learned features to 2 neurons, representing latitude and longitude.

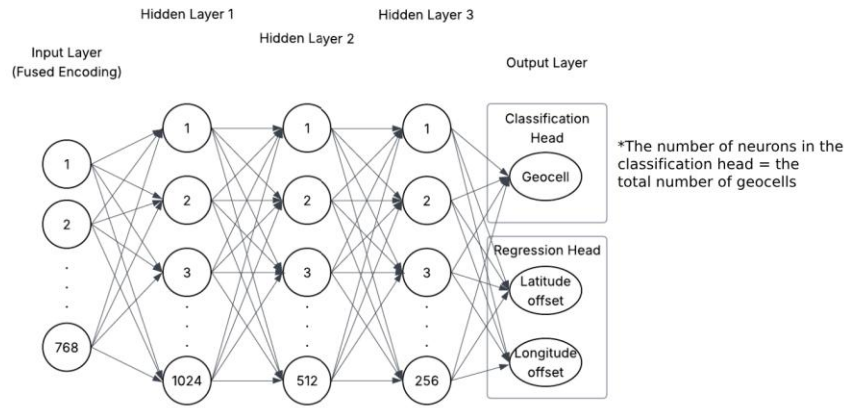


Figure 7: Neural network architecture (MLP). Circles are neurons. Created with [33]

2.2.4 Privacy-Preserving LVLM

The privacy-preserving LVLM is built identically to the baseline LVLM to allow for a fair and meaningful comparison. Two key changes were designed to enhance the privacy of the baseline model:

- **Text Blurring:** For training, all images in the dataset have any textual elements blurred such as street signs, road names, and shop names. This is so the model cannot pick up on any textual information during training, which directly informs the location of the image. When it comes to using the model, any text in uploaded images will be inherently ignored as the model did not learn to use it.
- **Larger Geocell Size:** The geocell size used for classification in the privacy-preserving model is larger than the baseline. The baseline LVLM uses a small geocell size of $0.05^\circ \times 0.05^\circ$ to promote precision and pinpoint accuracy, whereas the privacy-preserving LVLM uses a larger geocell size of $0.1^\circ \times 0.1^\circ$ to encourage more generalised area predictions and reduce pinpoint accuracy.

For sanity, to validate the individual impact of each privacy technique, two supplementary model versions are created which only apply one of the two changes. The four LVLM models are as follows:

- Unblurred images with geocell size $0.05^\circ \times 0.05^\circ$. (Baseline LVLM)
- Blurred images with geocell size $0.05^\circ \times 0.05^\circ$.
- Unblurred images with geocell size $0.1^\circ \times 0.1^\circ$.
- Blurred images with geocell size $0.1^\circ \times 0.1^\circ$. (Privacy-preserving LVLM)

2.2.5 Training the MLP

The baseline and privacy-preserving models' MLP must be trained with the exact same approach to ensure a fair comparison. All hyperparameters, such as learning rates, are the same for all models. The classification head uses cross-entropy to evaluate the geocell predictions, and the regression head uses a Haversine loss function to evaluate the real-world distance from the final prediction and the true coordinates. Because the MLP uses both heads together, the overall loss was decided to be a weighted combination of the regression and the classification losses, outlined in Equation 2. The classification loss should be weighted more heavily, as correct geocell classification is far more integral for a pinpoint prediction.

$$Total\ Loss = Cross\ Entropy\ Loss + Weight(Haversine\ Loss) \quad (2)$$

2.2.5.1 Cross Entropy Loss

Cross entropy is a widely used machine learning loss function for optimizing classifiers, measuring how similar probability distributions are [6]. Classifiers generate a probability distribution, which is then compared to the real distribution. For example, an animal classifier may predict that an image of a dog is 90% dog, 8% cat and 2% fox. This distribution would be compared to the real probability distribution of 100% dog, 0% cat, 0% fox, and generate a score. The mathematical function for cross entropy is intricate and needs prior knowledge of information theory, but conveniently there are machine learning libraries which offer a ready-to-use implementation of cross entropy loss in Python [45].

2.2.5.2 Haversine Loss

Haversine loss measures the real-world distance between two coordinates, accurately accounting for the shape of the globe. This loss metric is an obvious choice to optimise the coordinate offset regression, getting the actual distance from the predicted coordinates to the true coordinates. Other popular regression losses, like mean squared error and Euclidean distance, are less meaningful for geographic coordinates as they assume a flat plane and ignore curvature. The Haversine formula to find the distance between two points on a sphere is as follows, with three equations [32,49]:

$$a = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat_1) \cdot \cos(lat_2) \cdot \sin^2\left(\frac{\Delta long}{2}\right) \quad (3)$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a}) \quad (4)$$

$$d = R \cdot c \quad (5)$$

Where:

- R = Earth's radius (km)
- $\Delta lat = (lat_2 - lat_1)$
- $\Delta long = (long_2 - long_1)$
- c = axis intersection calculation
- d = distance (km)

2.2.5.3 Validation Set

My dataset of 40,000 items is split into a training set and validation set with an 8:2 ratio, 80% of which is used for training and the remaining 20% for validation. This validation set acts as unseen data, which is used to test whether the MLP is overfitting to the training data or generalizing well. Additionally, the validation set is utilised by the learning rate scheduler to help convergence during training and generate the key performance statistics of my models once training is complete.

2.3 Risk Analysis

Before software development began, it was clear that this would be a very large project with many dependencies and potential risk. Following the designed project approach, a few major risks were identified.

- Data Loss: Losing code or datasets would be detrimental, so regular backups and version control are essential.
- Design Flaw: It would waste lots of valuable time if designs were ineffective. Continuous prototyping and testing will ensure the viability of designs.
- Processing Time: Data collection, preprocessing, and training could take a substantial amount of time. The code must be optimised, utilising the GPU where possible and training models with batches to improve performance. As a contingency, the university lab machines could be used to process the code.
- Poor Model Performance: Since a lot of the UK's architecture and landscapes look similar, the model could struggle. If performance is too low, the dataset could be extended to cover Europe too, with its large variation of biomes and languages, but more data would have to be collected and preprocessed. Prototyping will help determine this early on.
- Privacy Concerns: A highly accurate model poses a risk of being misused. If the model performs well, care must be taken not to share the models or source code publicly.

2.4 Environments

For the implementation, all code is written in Python. To manage packages and dependencies effectively, the Anaconda environment [2] is used, pre-installed with a multitude of essential Python packages that simplifies their management.

As identified in the risk analysis, processing may be very computationally intensive and time-consuming. A device with a dedicated NVIDIA GPU was used, which is well-suited for intensive AI processing tasks that benefit from parallel computation. To utilize the GPU, CUDA [40] is installed, NVIDIA's software toolkit to interact with a device's GPU.

The majority of the LVLM implementation is developed with PyTorch [42], a widely used machine learning library designed to help build and train neural networks in Python. PyTorch also integrates with CUDA to allow for easy and efficient GPU utilization during processing and training.

2.5 Development Approach: Prototyping

Based on the risk analysis, employing a solid software development approach was vital. With the time constraints of this huge project, wasting time fully developing any unfunctional designs would cause delays. Therefore, agile development practices were adopted to continuously develop quick iterative prototypes during implementation, producing physical proof of concepts to validate and test the work.

Prototyping proved to be extremely useful. A major flaw was discovered with the initial neural network design, solely relying on regression for predictions. After validating the prototype, it was found that it consistently predicted the centre of the UK, as the mean location provided its lowest average loss. After some thorough research on the matter, the design was revised to treat the geolocation as a classification task as described in [23]. If this issue was discovered at the end of the development, it would have consumed lots of time.

2.6 Version Control

To manage code during development, GitHub [20] was used. Version control is an essential tool in a large and complex project, acting as both back-up and documentation. If code was lost or corrupted, or if previous versions of code needed to be accessed, this was easily accessible with GitHub's clean web interface. Additionally, GitHub allows the secure sharing of project code privately with the supervisor and assessor without any risk of the geolocator code or model files being misused.

Chapter 3

Implementation and Validation

3.1 Code Design and Standards

Given the scale of the project, strict coding practices were followed for maintainability. Clear and detailed code comments were written, explaining the use and logic behind functions and key sections of code. This included referencing any external materials used in-code. These practices ensured the code remains easy to understand over time. The implementation is also fully modular, with each component of the process separated into its own function or class, allowing for easy testing, debugging, and extension. Overall, the final implementation is very organised and professional.

3.2 Collecting Data

To fetch street-view images from the Mapillary API in Python, the Requests [47] library is used. Requests allow HTTP GET requests to be sent to Mapillary to retrieve data for a specified coordinate range. As stated in Section 2.1.1, there are two bounding boxes, each needing an individual GET request. Once both responses are received, the two results are combined proportionally based on their area and shuffled. Because Mapillary has a limit of how many items can be returned per GET request, the code had to be run many times to gather 40,000 items total. This dataset is then deep copied, one to be used for the baseline LVLM and the other to be blurred to train the privacy-preserving LVLM.

3.3 Geocells

Splitting the entire Earth into geocells results in a massive number of categories, far beyond the memory capacity of the GPU to be able to classify. To reduce the number of categories to classify into, the bounding boxes from data collection were used as the area to split into geocells. Two functions were implemented: one to convert a coordinate to a geocell and the second to convert a geocell to its centre coordinate. A coordinate's latitude cell and longitude cell are calculated using integer division, and the resulting 2D pair is flattened to a 1D geocell index in row-major order. This operation is reversed to find a geocell's centre coordinate.

3.4 Preprocessing

3.4.1 Image Captions

For the textual element of the LVLM, image captions are generated and any geographic words detected in the image are appended to the end of the string. Examples can be seen in Figures 8-10.



“a grassy hillside with some rocks and a fence on the side of the road”



“uk motorway Leicester Peterborough 447 Gedney B1167 Hill B1167”



“a small town on the banks of a river eurospar”

Figures 8, 9 and 10: Captions. Images from Mapillary [36], licensed under [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/).

3.4.1.1 Caption Generation

BLIP-2 [30] is a pretrained machine learning model for zero-shot image-to-text generation. The “blip2-opt-2.7b” variant is downloaded and imported from HuggingFace’s Transformers library [54], a very large model containing 2.7 billion parameters, requiring 14 GB of storage. The model offers prompted image captioning, allowing for tailored responses when providing it with a formatted question. The following prompt was chosen to concisely provide meaningful geographic captioning:

“Question: Describe this UK location concisely, including road type, buildings, signs, vegetation, and surroundings.”

Although it provides very accurate and meaningful captions, it is slow and memory intensive to generate text. To accelerate this process, the GPU was used, which drastically increased processing speed. Despite this, a single image caption would still take an average of 15 seconds to generate. Since every image in the dataset requires captioning twice for their blurred counterpart, 80,000 image captions are needed totally. At 15 seconds each, captioning alone required 13.9 days of continuous processing, not accounting for the rest of the model processing.

3.4.1.2 Text OCR Reader and Cleaning

To extract any textual elements in images, an Optical Character Recognition (OCR) tool was needed. EasyOCR [27] was selected, offering a simple and efficient OCR implementation in Python, to quickly scan the images for any text and return it as a list. However, since the dataset is user-uploaded, images can contain noise or even embedded metadata and need to be cleaned. As shown in Figure 11, even GPS coordinates can be embedded on the image. This data needs to be ignored by the OCR reader to retain fairness and privacy.

To address this, a complex cleaning function was created to filter out certain text using a combination of word lookups, regex patterns, and other rule-based filtering. An integral part of the processes utilises GeoPy [19], a geocoding library, to verify whether a word refers to a geographic term or location such as “bridge” or “London”. For example, Figure 11 has a noisy OCR output which is cleaned to simply “St Ninian High School”, which is the only text added to the end of the generated caption.


Image	OCR output	Cleaned text	Caption
	['St Ninian \$', 'High School', 'GARMIN', '20/12/2021 09:08:49 AM', '55.79857 - 4.30273 13 MPH', '"cotncaouy", '7']	“St Ninian High School”	“a car park in the middle of a residential area St Ninian High School”

Figure 11: OCR text cleaning. Image from Mapillary [36], licensed under [CC-BY-SA](#).

3.4.2 Text Encoding

To generate the text encoding of each caption, the machine learning model BERT [14] is used. BERT was chosen for its strong performance, outperforming other text embedders for natural language processing such as Word2vec [17]. BERT is pretrained on a massive amount of text data and has a deep contextual understanding of words in a sentence, whereas Word2vec is unable to capture the context around words. Similarly to BLIP-2, HuggingFace’s Transformer library is used to download and implement the “bert-base-uncased” model in Python. Once an image caption is passed through the BERT model, the CLS token is extracted from the first position of the output. CLS tokens are a singular vector which represent the entire input sequence [17], exactly what is needed for a textual encoding. This textual encoding represents the context and meaning of the entire sequence.

3.4.3 Vision Encoding

To generate meaningful visual encodings, the Vision Transformer (ViT) model was chosen, using HuggingFace's "vit-base-patch16-224-in21k" variant [13,15]. The ViT model is highly effective at capturing context, and was mainly selected for its powerful pretraining database, ImageNet-21k. Consisting of 14 million images with 22,000 classes, it excels at object recognition, perfect for diverse and complex street-view images. Identically to the textual encoding, the CLS token is captured from the ViT output. This represents the image as a vector sequence, containing the objects, semantics, and overall context of the image.

3.4.4 Fused Encoding

The visual and textual encodings are fused with cross-attention, a state-of-the-art machine learning mechanism that links relationships between two different data encodings. This method was used as the fusion technique due to its proven success at combining a multitude of diverse data sources, outperforming alternatives such as self-attention [11]. A PyTorch implementation by [55] was chosen, which calculates the similarity between each text-image pair, then produces a weighted sum of image features for each piece of text. The fused output is a vector of length 768 and will be passed into the neural network.

3.4.5 Blurring Images

For the privacy-preserving LVLM, all training images must be blurred to prevent the model from extracting text. This approach uses EasyOCR [27] and OpenCV [41]. EasyOCR is used to detect any text in the image, returning the bounding box coordinates. For each detected word, the bounding box is used to calculate a Region of Interest (ROI) in the image array. Then, OpenCV's GaussianBlur function is applied within every ROI to blur the text regions. Figures 12 and 13 demonstrate the blurring result, successfully obscuring text on the sign.



Figures 12 and 13: Image blurring. Mapillary [36], licensed under [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/), modified.

3.4.6 Saving Data

Preprocessing is extremely time-consuming, especially the caption generation. To avoid repeating this process, once the fused encodings are generated for a batch of images, they are saved as a `.pk1` file. This allows easy loading, appending, and resaving of data incrementally, whilst gathering and processing the full dataset. Once all preprocessing is complete, the saved fused encodings are conveniently loaded and used to train the models multiple times.

Later in the process, after the training is completed, the final models are saved as `.pth` files. These file types save the current model parameters, meaning the trained models can be loaded and used for predictions without needing to retrain the model each time.

3.5 Neural Network (MLP)

The neural network consists of three main components, as outlined in Section 2.2.3: the feature extraction, the classification head, and the regression head. The implementation was achieved using libraries and functions from PyTorch [42].

3.5.1 Feature Extraction (Hidden Layers)

PyTorch's `torch.nn` library offers well-made tools and implementations for building and training neural networks in Python. Using the fused encoding as input, the network transforms the input through 3 hidden layers to generate a vector of learned features with a length of 256. The layers are built using `nn.Linear` to perform the linear transformations, alongside the activation function `nn.GELU` to add non-linearity. The most popular activation function `nn.ReLU` was not used, as GELU produces smoother outputs and has been found to consistently exceed the accuracy of ReLU [25]. The resulting learned features can then be passed into the classification and regression heads.

3.5.2 Classifier

The classification head in the MLP is very simple to implement using PyTorch. A single `nn.Linear` function performs a linear transformation on the learned features to match the desired number of classes. For example, using a geocell size of $0.1^\circ \times 0.1^\circ$, there are 14,190 possible geocells, and so the learned features of length 256 are transformed to a length of 14,190. This output represents the geocell logits, the score for each geocell, where the highest

logit is the final classification. However, the final classification is not returned directly, as the logits are required by the classification loss function during the training loop. Additionally, the logits do not need to be converted into a probability distribution as this is handled automatically in the cross-entropy loss function.

3.5.3 Regressor

The regression head uses a single `nn.Linear` to project the length-256 learned features vector into 2 dimensions, latitude and longitude. These values represent the offset from the centre of the predicted geocell. To bound these coordinates within the geocell, they are passed into a sigmoid function, using `torch.sigmoid`, to clamp the coordinates to a range between 0 and 1. A sigmoid function was chosen instead of normalization to ensure every input is always between 0 and 1, especially since the minimum and maximum values aren't known in advance. Finally, 0.5 is subtracted from the result and then scaled to the geocell size, so that the offset represents a coordinate within the geocell, relative to its centre.

3.6 Training

Initially, the neural network performs poorly at making predictions with its default parameters. The MLP must be trained through gradient descent to minimise the loss. In the training loop, the optimizer tweaks the neural network parameters to continuously improve its predictions on the training set. Every complete loop through the dataset is called an epoch, and training continues until the results stop improving over multiple epochs. To improve efficiency, the training and validation sets are split into batches, allowing multiple items to be processed in parallel. All of the following training processes are implemented with PyTorch to ensure compatibility with the MLP.

3.6.1 Loss Functions

The training loop calculates the mean classification loss and mean regression loss for the current batch. These losses are combined into a total loss using the weighted loss equation shown in Equation (2). After trialling many weights, weights of 0.6 for the regression and 1 for the classification were selected, as it resulted in the best predictions. This weighting gives greater importance to the classification task while still accounting for the regression, creating a balance between classification accuracy and precision.

3.6.1.1 Cross Entropy Loss

For the classification loss function, PyTorch's `nn.CrossEntropyLoss` class is used. It takes the MLP's classification logits as input and calculates the mean loss for every item in a batch at once. This loss function has a built-in softmax function, which normalises the input logits to sum to 1, therefore becoming a probability distribution over all the categories.

3.6.1.2 Haversine Loss

The Haversine algorithm for regression, described in Equations (3-5), is achieved using PyTorch's mathematical functions such as `torch.sin`, `torch.cos`, `torch.sqrt`, and `torch.atan2`. These functions are designed to work with batches, allowing the mean loss to be computed for an entire batch at once rather than slowly processing them individually. In the training loop, the coordinate prediction is calculated using the geocell centre and the coordinate offset. This coordinate is compared with the true coordinate using the Haversine loss measuring how far the final prediction is from the goal.

3.6.2 Gradient Calculation

Once the total loss for the current batch is computed, the PyTorch function `tensor.backward` is called on the loss. This function performs backpropagation - calculating the gradients of the loss with respect to each of the current neural network weights [18]. This function neatly encompasses the complex gradient computation from the user, differentiating using the chain rule. The gradients calculated from the loss are later used by the optimizer to perform gradient descent and update the model parameters.

3.6.3 Optimizer

The optimizer is responsible for updating the neural network parameters to improve the predictions and minimise loss. One of the most popular optimizers `torch.optim.Adam` is used, built by PyTorch to implement the Adam algorithm [28]. Adam is an adaptive learning rate algorithm, capable of fine-tuning the learning rate for specific parameters rather than relying on a global learning rate [52]. This optimizer enables fast training times and minimal hyperparameter tuning. The only hyperparameter which had to be set was the learning rate, using the default at 0.001, as this performed very well. The learning rate determines by how much the neural network parameters are updated each epoch.

During the training loop, the optimizer has two important tasks for each epoch [18]: firstly, clearing the old gradients using `Optimizer.zero_grad`. Then, after `tensor.backward` is called to calculate the new gradients from the loss, the optimizer uses `Optimizer.step`. This function passes the computed gradients to the Adam algorithm and updates all parameters in the neural network. This process is performing the gradient descent, updating the parameters of the neural network towards the direction of minimising loss.

Before calling `Optimizer.step`, a key step is performed called gradient clipping. The gradients computed from the loss can potentially become too large, or “explode”, causing the optimizer to make unstable updates [5]. To prevent this, gradients are clipped to a total maximum value of 1 using Pytorch’s `nn.utils.clip_grad_norm_` function. If they exceed the limit, they are all scaled down proportionally.

3.6.4 Scheduler

A key component of the training loop is the learning rate scheduler. The learning rate represents the step size of gradient descent. If the step size is too small, the model will take too long to converge to the minimum [9]. Yet if it’s too large, the model will bounce around the minimum and never converge. The ideal case is for the learning rate to begin large, quickly approaching the minimum, only then reducing to reach the precise minimum.

Multiple PyTorch schedulers were thoroughly tested from the `lr_scheduler` library, all reducing the learning rate with different criteria, namely `StepLR`, `CosineAnnealingLR` and `ReduceLROnPlateau`. For each scheduler, hyperparameters were adjusted and results were recorded each time. `ReduceLROnPlateau` was eventually selected after it showed the best performance, with the hyperparameters `factor=0.5` and `patience=3`.

The `ReduceLROnPlateau` scheduler reduces the learning rate when no improvement has been seen across multiple epochs, when it “plateaus”. The validation loss, calculated on the validation set, was chosen as the measurement the scheduler should track. Validation loss is a key indicator of how well the model is generalising to unseen data and using it avoids overfitting to the training data. This secondary loss is passed to the scheduler through the `ReduceLROnPlateau.step` function, which adjusts the learning rate if the validation set performance hasn’t improved over multiple epochs. With the chosen hyperparameters, this specifically means that the learning rate is halved if no improvement has been seen to the validation set over 3 epochs. This successfully allows the model to converge both quickly and precisely to the minimum.

3.6.5 Validation Set

As mentioned in Section 2.2.5.3, the dataset is split into a training set and a validation set, using the validation set to measure model performance on unseen data. This is achieved using PyTorch's `random_split` with a manually set random seed, dividing the dataset and ensuring reproducibility across runs. After every epoch in the training process, the current model performance is evaluated on the validation set. This process is very similar to the training in terms of code, except for the loss function used. The weighted loss is not required for validation, as the Haversine loss accurately reflects the average distance between predictions and true coordinates. This validation loss is returned to the training loop for the learning rate scheduler to track, as it offers key insights into the model's generalisation capability. The validation set is also used to generate the final model statistics, such as the classification accuracy and median distance, which are recorded from the model performance on the final epoch of training. These validation statistics are constantly used for analysing and comparing models throughout the project.

3.6.6 Batches

Batches involve dividing the dataset into smaller segments and processing each segment in one go. This approach offers many benefits such as computational efficiency and stability [38,44]. Firstly, using batches allows for faster computation as multiple items within the batch can be processed in parallel, and loss is calculated for the whole batch rather than individually. Secondly, by taking an average loss across a batch, the optimizer's updates are smoothed out to prevent erratic parameter updates. Any large individual gradients could cause large unstable updates, slowing convergence. However, if the batch size is too big, each epoch would only backpropagate once for the entire dataset and wouldn't learn well. Ideally, we would want the neural network to be updated by every individual training item, but this is computationally inefficient and causes large jumpy updates. To balance the trade-offs, a batch size of 32 is used, which is a common default. This is implemented with PyTorch's `DataLoader`, allowing simple batch creation and offers useful features such as parallel loading and reshuffling of the batches every epoch to avoid overfitting.

3.7 Using the Models

With the LVLM models saved as `.pth` files after training, they can easily be used for evaluation. The code has two modes: predicting multiple images and predicting singular

images. In the singular image mode, the user uploads an image file to predict. This is the normal function that a real-world user would use to predict a custom image location. For the multiple image mode, a batch of random Mapillary images are fetched to predict. The primary use of this mode is to test and evaluate model performance on many images at once, allowing efficient testing of which type of images predict correctly or struggle.

As the images are processed, they follow the same process to generate fused encodings, which are then passed into the trained neural network to predict. The final coordinate is calculated by adding the outputted geocell coordinate and the geocell offset, as stated in Equation 1. This is performed for all LVLM models, with the results printed for each one, allowing easy comparison of model accuracies on the same image.

3.8 Validation

Throughout development and training, models were continuously validated using the validation set statistics, testing different learning rate schedulers, hyperparameters, and prototype designs. In addition to quantitative validation, predictions were generated for 100 new and unseen Mapillary images using the multiple images mode for every prototype created. Their performance was manually analysed to observe common trends and evaluate individual image performance alongside their statistics. This validation proved itself very important, as mentioned in Section 2.5, where an early model was discovered to simply guess the centre of the UK for a low average loss without properly learning. In the next chapter, I thoroughly validate the performance and completeness of my final models as part of the evaluation process, using statistics and examples from the validation set.

3.9 Testing

Most of the testing was performed manually for simplicity and time efficiency. Each new feature was thoroughly tested before development continued. For example, to test the image blurring function, it was run with 100 Mapillary images to observe the results. Specific images were also uploaded to test a variety of edge cases, such as a book page with lots of text, an image with no text, and an image with very small text. While more formal documentation and testing could have been performed, such as creating test plans or implementing automatic unit testing, this rapid development process prioritised quick iterations with manual testing, prototyping, and direct inspection to continuously validate and improve the models.

Chapter 4

Results and Evaluation

4.1 Overview

Four different models were trained, listed in Figure 14. The model “baseline005LVLM” is the baseline LVLM with no privacy-preserving techniques, and “blurred01LVLM” is the privacy-preserving LVLM which employs both techniques: blurring images and larger geocell size. The intermediate models each employ a single privacy technique, so the effectiveness of individual techniques can be validated.

File Name	Blurred/Unblurred	Geocell Size (°)
baseline005LVLM.pth (Baseline model)	Unblurred	0.05
blurred005LVLM.pth	Blurred	0.05
baseline01LVLM.pth	Unblurred	0.1
blurred01LVLM.pth (Privacy model)	Blurred	0.1

Figure 14: Trained LVLMs

4.2 Qualitative Analysis

4.2.1 Overall Performance Across Models

Before analysing each model’s individual statistics, some example images have been compiled from the validation set in Appendix D, to demonstrate patterns across all models. This analysis confirms that this project has successfully created a machine learning model capable of geolocating images, setting the foundation to explore the impact of each privacy technique. The following observations are universal to all the geolocators.

Appendix D.1 showcases example images that the models predicted with high accuracy. Images containing distinctive landscapes and features are always predicted well, demonstrating that the models are effective at identifying geographic landmarks like mountains and rivers. From testing predictions on hundreds of images, it was observed that the models consistently correctly predicted areas such as Cornwall, Scotland, and Ireland. These regions have unique landscapes and architectural styles, which the models can easily identify, compared to mainland England which often looks similar. Coastal towns are also

geolocated well, with their distinctive colourful architecture and ocean views. In many cases, an image only showing houses is geolocated to the exact coastal town they are located in.

Although, it was recognised that sometimes the models could “cheat” by relying on unintended features to geolocate. Some images in the user-uploaded dataset have obstructions on the screen, such as a car bonnet or motorcycle helmet. Because these obstructions were consistent across all images uploaded from the same user, who tend to upload images from the region they live in, the models are able to exploit these patterns and predict images with no clear location-specific features.

Appendix D.2 showcases examples that all models struggled to geolocate. Many of the user-uploaded images are low-quality, badly composed, and out-of-focus, such as a blurred bush, which is simply not locatable. These images bring the model statistics down significantly. Additionally, images inside buildings perform badly, which is expected as the models are trained primarily on outdoor street-view data. Interestingly, the models struggle to geolocate images from mainland England, unlike coastal towns and mountainous regions as mentioned earlier. This is likely due to the generic nature of English housing and motorways throughout the country, which lack distinctive features and are very homogeneous. During validation, it was often seen that models predict a visually similar but incorrect road or town. In many cases, these predictions likely fail due to the limited size of the training dataset. The models are only trained on 32,000 images total, which is not fully representative of every location in the UK. As a result, many towns and roads are completely unseen to the model, and so it instead predicts a similar looking location that it has encountered.

4.2.2 Case Study: Alpine Guest House

This case study provides a clear demonstration of the impact of the privacy-preserving techniques. Figure 15 displays an image from Mapillary, and Figure 16 displays the text in the image. As you can see, the image details a rainy road with a sign reading “Alpine Guest House”. Figure 17 presents the predictions of the four geolocators on this image.



Figures 15 and 16: Alpine Guest House. Mapillary [36], licensed under [CC-BY-SA](#).

Model	Distance (2.d.p)
Unblurred 0.05 (Baseline model)	0.03 km (30 metres)
Blurred 0.05	271.67 km
Unblurred 0.1	2.08 km
Blurred 0.1 (Privacy model)	350.06 km

Figure 17: Model performance on image shown in Figure 15.

The unblurred models are able to predict this image's location precisely. They successfully read the text on the sign and use it to pinpoint the location to 0.03 km and 2.08 km. Based on the text "Alpine Guest House" and the surrounding environment, having encountered this location before in the training dataset, they can accurately identify its location. The baseline model using a smaller geocell size manages to predict 30 metres away, demonstrating extreme accuracy. Small geocell sizes allow for finer classification, which when correct, enable precise predictions and easy regression for the offset. However, the larger geocell model predicts 2 km away. Even though it still recognises the text in the image, classification is less precise, making the regression task much more challenging. The coordinate offset is now being predicted over a quadratically larger area by doubling the geocell size, which we can see here it struggles to do. While this model still finds the general area, the increased geocell size mainly just adds noise to the prediction, which incidentally does help privacy by weakening pinpoint accuracy, albeit in an indirect way.

Alternatively, the blurred models can not predict the location. As text was blurred during training, the only information the models can rely on is the surrounding environment instead of signage. Figure 15 doesn't include many identifiable features, and the trained model has no information about "Alpine Guest House", so it ignores this text. Consequently, blurred models are unable to determine the location of this image precisely, predicting 271.67 km and 350.06 km. This shows that training the model with blurred images has successfully prevented the location from being revealed, proving the technique's effectiveness. However, the predictions are very inaccurate and not useful for a geolocator, highlighting the trade-off for privacy. Increasing the geocell size in this case has made the model even worse, which seems to be due to incorrect classification rather than struggling to regress, as the models cannot predict the image without using its text.

Overall, the privacy-preserving techniques are effectively improving image privacy. Blurring prevents the model from learning from sensitive text, and increasing geocell size decreases pinpoint accuracy by encouraging broader classifications and adding noise. But, at the same

time, the accuracy for the privacy-preserving models is significantly reduced, questioning the usefulness of the geolocator.

4.3 Quantitative Analysis

Figure 18 displays the final statistics of the four models, calculated using all 8,000 unseen images from the validation set.

Model	Median Distance (km)	Geocell Accuracy (%)	Within 100km (%)	Within 10km (%)	Within 1km (%)	Within 100m (%)
Unblurred 0.05 (Baseline model)	39.241	36.7	57.7	41.9	31.3	23.0
Blurred 0.05	41.896	35.6	58.3	41.6	30.7	22.1
Unblurred 0.1	35.582	38.9	58.8	42.1	29.6	17.1
Blurred 0.1 (Privacy model)	37.770	38.1	58.9	41.3	28.9	13.1

Figure 18: Model Statistics. The best performance for each statistic is highlighted in bold.

4.3.1 Baseline LVLM

As shown in Figure 18, the baseline model delivered great results. With a median distance of 39 km, on average it demonstrates high accuracy in its predictions within the UK. Where the model shines is its alarming pinpoint accuracy – 31% of images are predicted within 1 km, and a surprising 23% of images are located perfectly within 100 m. That is 1,840 out of 8,000 images perfectly predicted, showcasing the LVLM’s power. With a functional, accurate geolocator, the effectiveness of the privacy-preserving approaches can be validated.

4.3.2 Privacy Technique 1: Blurring Images

As shown by the “Blurred 0.05” results in Figure 18, blurring text has a significant impact on the baseline model’s performance. The median distance increases, and the geocell classification accuracy reduces, both indicating a reduction in accuracy. Observing the predictions within 10 km, 1 km, and 100 m, we see similar trends, where blurring the model reduces the overall accuracy slightly. Predictions within 100 m reduce by approximately 1%, implying that 1% of the validation set images relied on text for their pinpoint accuracy in the baseline model, and could no longer be geolocated in the blurred model. There is one

exception, with the blurred model performing slightly better on predictions within 100 km. This suggests that the blurred model generalises better by not relying on text, resulting in slightly reduced pinpoint accuracy but broader accuracy overall. Blurring the images during training successfully prevents sensitive text being used to localise the image.

4.3.3 Privacy Technique 2: Larger Geocell Size

“Unblurred 0.1” in Figure 18 shows the result of doubling the size of the geocell used in the baseline model. While this change increases overall accuracy, it reduces pinpoint accuracy. The median distance decreases to 35.5 km, approximately 4 km better than the baseline model. Additionally, the geocell classification accuracy, as well as predictions within 100 km and 10 km, all improve, indicating that the model has better accuracy on a broader scale. However, predictions within 1 km and 100 m drop, with perfect predictions within 100 m being 6% lower than the baseline model. Classification becomes broader and more successful, while regression struggles to predict exact coordinates. Generally, this model is more consistently correct but sacrifices the high pinpoint accuracy of the baseline model. It does address the privacy problem, providing noisier predictions that still locate the correct area.

4.3.4 Combination of Both Techniques: Privacy-Preserving LVLM

The final privacy-preserving model combines both techniques, which have proven effective individually. Of course, the statistics for this model reflect the combined trends previously observed in each privacy technique. Compared to the baseline model, the median distance is 1.5 km closer, and the geocell accuracy is almost 2% higher. We also see the highest predictions within 100 km of all models at 58.9%, 2.2% higher than the baseline model. This model generalises better due to the broader classification and less reliance on text in images. However, precise accuracy begins to fall. Predictions within 1 km are 2.4% lower, and within 100 m almost 10% lower. This seems to be very successful, as eliminating the pinpoint precision of the baseline model is key to addressing the privacy concerns. Interestingly, comparing the within 100 m statistics, the models with a smaller geocell size saw a 1% drop by blurring images. Yet for this larger geocell size, the drop is 4%, a much more significant change, suggesting that the model relied more heavily on text with broader geocells. This further highlights the success of ignoring text, as the model now provides more general accuracy with less pinpoint precision – exactly what is needed for a privacy-preserving geolocator.

4.4 Final Discussion: Balancing Privacy and Utility

After analysing the results, we clearly see that the privacy-preserving model provides a major reduction in predictions within 1 km and 100 m. The model manages to reduce predictions within 100 m from 23% to 13%, while largely retaining, or even improving, the other model performances. The privacy-preserving geolocator offers a successful approach to begin addressing the privacy issue, aiming to increase generalised accuracy whilst reducing pinpoint accuracy. The model inherently ignores text in images when geolocating, avoiding one of the most direct privacy violations by protecting a user's exact location from being identified through signs or words. This success means additional privacy mitigations can be implemented, such as obscuring houses, which won't severely damage the model's utility.

To remain a useful tool, the privacy-preserving geolocator must be still accurate and able to make reliable predictions. While the geolocator's reduced accuracy undoubtedly hinders its utility, it is an essential sacrifice to preserve privacy. The tool is nonetheless still able to accurately predict many natural landscapes and features with pinpoint accuracy that pose lower privacy risks.

However, with a 13% accuracy within 100 m, it can be argued that the privacy risk remains. Every feature of an image is potentially privacy-violating, such as mountains, a river, or house architecture, all of which we've seen the models can recognise and predict well. A model trained on billions of images could potentially predict locations precisely without ever needing text. It raises the question of whether every location-identifying feature is a privacy risk. If every detail was obscured, geolocation would be impossible, meaning a truly privacy-preserving geolocator is simply an ineffective one. This highlights the unavoidable trade-off between preserving privacy and retaining the practical utility of geolocator tools.

It should be noted that if the goal of privacy preservation was to achieve 0% pinpoint accuracy, you could just add random noise to the output of the baseline geolocator. However, this approach would undermine the tool's utility for real-world usages like environmental monitoring and disaster response if it was never able to predict precisely. The model seeks a balance: sacrifice precision when it comes to sensitive human features, such as text, while retaining the accuracy for non-sensitive environmental features, preserving its practical value.

Ultimately, the model demonstrates a thoughtful balance between privacy and usability, making reasonable sacrifices from both metrics to satisfy both ends. The results are a promising step towards developing responsible geolocation tools.

Chapter 5

Conclusions

5.1 Conclusions

This project aimed to develop a machine learning tool capable of predicting the location of an image in the UK, while also preserving privacy. Both a functional baseline and privacy-preserving geolocator were successfully created and trained, built with a large vision language model and multilayer perceptron as the neural network. With thoroughly researched privacy techniques, it was demonstrated that the pinpoint precision of a geolocator can be effectively reduced without compromising the tool's overall utility. The privacy-preserving geolocator achieved a meaningful drop in pinpoint accuracy, while slightly increasing the general accuracy. Specifically, it effectively ignores sensitive text in images, while retaining its precision for geolocating natural features. Overall, all objectives were met, and the project positively contributes to ethical geolocation development.

5.2 Future work

The privacy-preserving geolocator offers many opportunities for future improvement. Additional sensitive human features, for example house architecture, could also be obscured to further prevent pinpoint predictions. This would ultimately create a model capable of pinpointing natural landscapes, while preserving the privacy of sensitive human features.

Major enhancements can be made to the underlying machine learning techniques. Expanding the dataset by gathering a larger range of street-view images and internet-sourced images would facilitate a much more accurate geolocator, enabling a stronger evaluation of the effectiveness of privacy techniques. During model training, images could also be distorted, cropped, and transformed to allow for better generalisation. This would also prevent the models from identifying "cheat" features, such as the edge of a motorcycle helmet. Future research could also explore applying the privacy-techniques to pretrained LVLMs, which can offer staggeringly higher performances.

Finally, conducting comprehensive adversarial testing, trying to attack the privacy models, would provide meaningful insights into the resilience of the privacy techniques. This analysis would help identify the nuanced strengths and weaknesses of the privacy features, guiding future implementations to strengthen against reverse engineering or potential vulnerabilities.

List of References

- [1] Almeida, L.B. 2020. C1.2 Multilayer Perceptrons. In: Fiesler, E. and Beale, R. eds. *Handbook of Neural Computation*. [Online]. Boca Raton, Florida: CRC Press. [no pagination]. [Accessed 22 January 2025]. Available from <https://www.perlego.com/book/1485544>
- [2] Anaconda Inc. 2023. *Anaconda* (version 24.9.1). [Software]. [Accessed 19 January 2025]. Available from: <https://www.anaconda.com/>
- [3] Baheti, P. 2021. Activation Functions in Neural Networks [12 Types & Use Cases]. 27 May. V7. [Online]. [Accessed 17 April 2025]. Available from <https://www.v7labs.com/blog/neural-networks-activation-functions>
- [4] Benani, A., Ohayon, S., Laleye, F., Bauvin, P., Messas, E., Bodard, S. and Tannier, X. 2025. Is Multimodal Better? A Systematic Review of Multimodal versus Unimodal Machine Learning in Clinical Decision-Making. *medRxiv preprint*. [Online]. [Accessed 27 March 2025]. Available from: <https://doi.org/10.1101/2025.03.12.25322656>
- [5] Brownlee, J. 2019. A Gentle Introduction to Exploding Gradients in Neural Networks. 14 August. *Machine Learning Mastery*. [Online]. [Accessed 10 April 2025]. Available from: <https://machinelearningmastery.com/exploding-gradients-in-neural-networks/>
- [6] Brownlee, J. 2020. A Gentle Introduction to Cross-Entropy for Machine Learning. 22 December. *Machine Learning Mastery*. [Online]. [Accessed 2 April 2025]. Available from: <https://machinelearningmastery.com/cross-entropy-for-machine-learning/>.
- [7] Brumfiel, G. 2023. Artificial intelligence can find your location in photos, worrying privacy experts. *NPR*. [Online]. 19 December. [Accessed 7 January 2025]. Available from: <https://www.npr.org/2023/12/19/1219984002/artificial-intelligence-can-find-your-location-in-photos-worrying-privacy-expert>
- [8] Chen, H., Huang, W., Ni, Y., Yun, S., Liu, Y., Wen, F., Velasquez, A., Latapie, H. and Imani, H. 2024. TaskCLIP: Extend Large Vision-Language Model for Task Oriented Object Detection. *arXiv preprint*. [Online]. [Accessed 11 January 2025]. Available from: <https://doi.org/10.48550/arXiv.2403.08108>
- [9] Continuum Labs. c2023. *Learning Rate Scheduler*. [Online]. [Accessed 10 April 2025]. Available from: <https://training.continuumlabs.ai/training/the-fine-tuning-process/hyperparameters/learning-rate-scheduler>
- [10] Cox, J. 2025. The Powerful AI Tool That Cops (or Stalkers) Can Use to Geolocate Photos in Seconds. *404 Media*. [Online]. 20 January. [Accessed 11 February 2025].

Available from <https://www.404media.co/the-powerful-ai-tool-that-cops-or-stalkers-can-use-to-geolocate-photos-in-seconds/>

- [11] Daou, S., Ben-Hamadou, A., Rekik, A. and Kallel, A. 2024. Cross-Attention Fusion of Visual and Geometric Features for Large Vocabulary Arabic Lipreading. *arXiv preprint*. [Online]. [Accessed 27 March 2025]. Available from: <https://doi.org/10.48550/arXiv.2402.11520>
- [12] De Curtò, J., De Zarzà, I. and Calafate, C.T. 2023. Semantic Scene Understanding with Large Language Models on Unmanned Aerial Vehicles. *Drones*. [Online]. **7**(2), article no:114 [no pagination]. [Accessed 11 January 2025]. Available from: <https://doi.org/10.3390/drones7020114>
- [13] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*. [Online]. pp.248-255. [Accessed 12 February 2025]. Available from: <https://doi.org/10.1109/CVPR.2009.5206848>
- [14] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint*. [Online]. [Accessed 8 February 2025]. Available from: <https://doi.org/10.48550/arXiv.1810.04805>
- [15] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR*. [Online]. [Accessed 12 February 2025]. Available from: <https://doi.org/10.48550/arXiv.2010.11929>
- [16] Gan, Z., Li, L., Li, C., Wang, L., Liu, Z. and Gao, J. 2022. Vision-Language Pre-training: Basics, Recent Advances, and Future Trends. *arXiv preprint*. [Online]. [Accessed 11 January 2025]. Available from: <https://doi.org/10.48550/arXiv.2210.09263>
- [17] GeeksforGeeks. 2024. *How to Generate Word Embedding using BERT?* [Online]. [Accessed 8 February 2025]. Available from: <https://www.geeksforgeeks.org/how-to-generate-word-embedding-using-bert/>
- [18] GeeksforGeeks. 2024. *PyTorch: Connection Between loss.backward() and optimizer.step()*. [Online]. [Accessed 6 April 2025]. Available from: <https://www.geeksforgeeks.org/pytorch-connection-between-lossbackward-and-optimizerstep/>

- [19] GeoPy. 2025. *GeoPy* (version 2.4.1). [Software]. [Accessed 8 February 2025]. Available from: <https://github.com/geopy/geopy>
- [20] GitHub. 2025. *GitHub*. [Online]. [Accessed 19 April 2025]. Available from: <https://github.com/>
- [21] Google. c2025. *Google Maps*. [Online]. [Accessed 1 February 2025]. Available from: <https://www.google.com/maps>
- [22] Graylark Technologies Inc. c2025. *GeoSpy*. [Online]. [Accessed 11 February 2025]. Available from: <https://geospy.ai/>
- [23] Haas, L., Skreta, M., Alberti, S. and Finn, C. 2024. PIGEON: Predicting Image Geolocations. *arXiv preprint*. [Online]. [Accessed 6 January 2025]. Available from: <https://doi.org/10.48550/arXiv.2307.05845>
- [24] Hashemi-Pour, C. 2024. What is geolocation? Explaining how geolocation data works. August. *Mobile Computing*. [Online]. [Accessed 6 January 2025]. Available from: <https://www.techtarget.com/searchmobilecomputing/definition/What-is-geolocation>
- [25] Hendrycks, D. and Gimpel, K. 2023. Gaussian Error Linear Units (GELUs). *arXiv preprint*. [Online]. [Accessed 19 February 2025]. Available from: <https://doi.org/10.48550/arXiv.1606.08415>
- [26] Information Commissioner's Office (ICO). n.d. *10. Geolocation - Age appropriate design: a code of practice for online services*. [Online]. [Accessed 7 January 2025]. Available from: <https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/childrens-information/childrens-code-guidance-and-resources/age-appropriate-design-a-code-of-practice-for-online-services/10-geolocation/>
- [27] Jaied AI. 2024. *EasyOCR* (version 1.7.2). [Software]. [Accessed 8 February 2025]. Available from: <https://github.com/JaiedAI/EasyOCR>
- [28] Kingma, D.P. and Ba, J. 2017. Adam: A Method for Stochastic Optimization. *arXiv preprint*. [Online]. [Accessed 10 April 2025]. Available from: <https://doi.org/10.48550/arXiv.1412.6980>
- [29] Kumar, D., AbuHashem, Y. and Durumeric, Z. 2024. Watch Your Language: Investigating Content Moderation with Large Language Models. *arXiv preprint*. [Online]. [Accessed 11 January 2025]. Available from: <https://doi.org/10.48550/arXiv.2309.14517>
- [30] Li, J., Li, D., Savarese, S. and Hoi, S. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. *ICML*.

- [Online]. [Accessed 6 February 2025]. Available from:
<https://doi.org/10.48550/arXiv.2301.12597>
- [31] Liu, Y., Ding, J., Deng, G., Li, Y., Zhang, T., Sun, W., Zheng, Y., Ge, J. and Liu, Y. 2024. Image-Based Geolocation Using Large Vision-Language Models. *arXiv preprint*. [Online]. [Accessed 6 January 2025]. Available from:
<https://doi.org/10.48550/arXiv.2408.09474>
- [32] Louwers, J. 2023. Calculate Geographic distances in Python with the Haversine method. 18 September. *Medium*. [Online]. [Accessed 20 February 2025]. Available from: <https://louwersj.medium.com/calculate-geographic-distances-in-python-with-the-haversine-method-ed99b41ff04b>
- [33] Lucid Software Inc. c2025. *Lucidchart*. [Software]. [Accessed 27 March 2025]. Available from: <https://www.lucidchart.com/pages>
- [34] Ma, K., Stewart, P. and Tjangnaka, W. 2024. Using Geoembeddings to Predict Image Geolocations. *Online paper from Stanford University CS231n: Deep Learning for Computer Vision*. [Online]. [Accessed 6 January 2025]. Available from <https://cs231n.stanford.edu/2024/papers/using-geoembeddings-to-predict-image-geolocations.pdf>
- [35] Mapillary. c2025. *Mapillary API Documentation - Image*. [Online]. [Accessed 19 January 2025]. Available from: <https://www.mapillary.com/developer/api-documentation#Image>
- [36] Mapillary. c2025. *Mapillary*. [Online]. [Accessed 19 January 2025]. Available from: <https://www.mapillary.com/>
- [37] MarketMuse. c2025. Multilayer Perceptron (MLP). [no date]. *MarketMuse Blog*. [Online]. [Accessed 17 April 2025]. Available from: <https://blog.marketmuse.com/glossary/multilayer-percpetron-definition/>
- [38] MLangner. 2024. Can someone explain the benefits of Batches? November. PyTorch Forums. [Online]. [Accessed 8 April 2025]. Available from: <https://discuss.pytorch.org/t/can-someone-explain-the-benefits-of-batches/212699>
- [39] Neuhold, G. 2018. Accurate Privacy Blurring at Scale. 19 April. *Mapillary*. [Online]. [Accessed 23 April 2025]. Available from: <https://blog.mapillary.com/update/2018/04/19/accurate-privacy-blurring-at-scale.html>

- [40] NVIDIA Corporation. 2022. *NVIDIA CUDA Toolkit* (version 11.8.90). [Software].
[Accessed 14 February 2025]. Available from: <https://developer.nvidia.com/cuda-toolkit>
- [41] OpenCV. 2025. *OpenCV* (version 4.11.0). [Software]. [Accessed 1 March 2025].
Available from: <https://opencv.org/>
- [42] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems* 32. [Online]. pp8024-8035. [Accessed 6 February 2025]. Available from: <https://doi.org/10.48550/arXiv.1912.01703>
- [43] Pluta, W. 2025. Graylark closes public access to AI tool for geolocation. *Heise Online*. [Online]. 22 January. [Accessed 11 February 2025]. Available from: <https://www.heise.de/en/news/Graylark-closes-public-access-to-AI-tool-for-geolocation-10252293.html>
- [44] pranshupant. 2021. Why use batches? March. *PyTorch Forums*. [Online]. [Accessed 8 April 2025]. Available from: <https://discuss.pytorch.org/t/why-use-batches/113370/2>
- [45] PyTorch. c2024. *Torch.nn.CrossEntropyLoss Documentation*. [Online]. [Accessed 2 April 2025]. Available from: <https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>
- [46] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) – Recital 38*. [Online]. [Accessed 7 January 2025]. Available from: <https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng>
- [47] Requests. 2024. *Requests* (version 2.32.2). [Software]. [Accessed 8 February 2025].
Available from: <https://requests.readthedocs.io/en/latest/>
- [48] Rezvani, R., Katiraei, M., Jamalian, A.H., Mehrabi, S. and Vezvaei, A. 2012. A New Method for Hardware Design of Multi-Layer Perceptron Neural Networks with Online Training. *2012 IEEE 11th International Conference on Cognitive Informatics and Cognitive Computing*. [Online]. pp.527-534. [Accessed 17 January 2025]. Available from <https://doi.org/10.1109/ICCI-CC.2012.6311205>
- [49] Saputra, K., Nazaruddin, N., Yunardi, D.H. and Andriyani, R. 2019. Implementation of Haversine Formula on Location Based Mobile Application in Syiah Kuala University.

- 2019 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom). [Online]. pp.40-45. [Accessed 20 February 2025]. Available from: <https://doi.org/10.1109/CYBERNETICSCOM.2019.8875686>
- [50] Stan, G.B.M., Aflalo, E., Rohekar, R.Y., Bhiwandiwalla, A., Tseng, S.-Y., Olson, M.L., Gurwicz, Y., Wu, C., Duan, N. and Lal, V. 2024. LVLM-Interpret: An Interpretability Tool for Large Vision-Language Models. *arXiv preprint*. [Online]. [Accessed 10 January 2025]. Available from: <https://doi.org/10.48550/arXiv.2404.03118>
- [51] Sun, J., Pi, P., Tang, C., Wang, S.-H. and Zhang, Y.-D. 2023. CTMLP: Can MLPs replace CNNs or transformers for COVID-19 diagnosis? *Computers in Biology and Medicine*. [Online]. **159**, p.106847. [Accessed 18 January 2025]. Available from <https://doi.org/10.1016/j.compbimed.2023.106847>
- [52] Wei, D. 2024. Demystifying the Adam Optimizer in Machine Learning. 30 January. *Demystifying Machine Learning*. [Online]. [Accessed 10 April 2025]. Available from: <https://medium.com/@weidagang/demystifying-the-adam-optimizer-in-machine-learning-4401d162cb9e>
- [53] Wilson, D., Zhang, X., Sultani, W. and Wshah, S. 2024. Image and Object Geo-Localization. *International Journal of Computer Vision*. [Online] **132**, pp.1350–1392. [Accessed 6 January 2025]. Available from: <https://doi.org/10.1007/s11263-023-01942-3>
- [54] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q. and Rush, A. 2020. Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. [Online]. pp.38-45. [Accessed 6 February 2025]. Available from: <https://doi.org/10.18653/v1/2020.emnlp-demos.6>
- [55] xbeat. 2024. *Self-Attention and Cross-Attention in Transformers with Python*. [Online]. [Accessed 13 February 2025]. Available from: <https://github.com/xbeat/Machine-Learning/blob/main/Self-Attention%20and%20Cross-Attention%20in%20Transformers%20with%20Python.md>
- [56] Yu, Y. and Zhang, Y. 2023. Multi-layer Perceptron Trainability Explained via Variability. *arXiv preprint*. [Online]. [Accessed 21 January 2025]. Available from <https://doi.org/10.48550/arXiv.2105.08911>

Appendix A

Self-appraisal

A.1 Critical Self-evaluation

I successfully managed to achieve all the aims set out for this project. The goals were appropriately challenging, not being impossible for the timeframe nor too simple, offering a valuable high-impact opportunity to apply my programming skills. I showed strong initiative to implement a complex LVLM geolocator, utilising a combination of new and established machine learning techniques to deliver the best results. Effective privacy techniques were applied to the model, thanks to my quality research, and proved that the dangers of geolocators can be successfully mitigated.

However, I struggled significantly with time management during implementation, due to underestimating how computationally intensive machine learning models can be. Almost 30 days were spent total leaving my laptop to run code all day and night, during which I couldn't use my laptop for anything else. From preprocessing the total 80,000 data items, to re-running the training many times with different hyperparameters. It was a very slow and energy-intensive process, which I tried optimising to the best I could with multi-threading and GPU usage. If I were to do it again, perhaps utilising the university lab machines would accelerate this process, as it took up a significant amount of time and delayed the planned timeline.

One major mistake was not researching thoroughly enough before beginning the implementation. As mentioned in the report, I initially didn't know that classification was necessary for geolocators, with regression simply not working on this scale, building a pure regression geolocator. Thankfully, I was able to catch onto my mistake quickly with the agile prototyping approach, involving constant testing, allowing me to identify and rectify the issue cleanly without wasting a significant amount of time.

A.2 Personal Reflection and Lessons Learned

I am extremely proud of this project and its achievements. At the start, I questioned whether the geolocator would be effective, yet through hard work these doubts were proven wrong. This was my first time conducting proper academic research, which I was initially terrible at. However, by the end, I can confidently say that my research skills have dramatically improved.

Implementing an LVLM in Python from scratch was very frustrating, as it is not well documented online at all. It was a long process, requiring multiple iterations and endless

testing of different solutions. Implementing it myself, learning through scientific papers and small-scale guides online, allowed me to develop a deep understanding of how the model and neural network function. My understandings in machine learning have also been fleshed out after creating my own implementation, compared to just studying theory, which will massively aid me in my future career as a computer scientist.

As mentioned in the critical evaluation, I have also witnessed first-hand how computationally intensive and bad for the environment machine learning models can be. This was eye-opening into how destructive large AI models like chatbots really are, considering that my very small model required my laptop to be left running 24/7 over 30 days.

All in all, I am glad that the project was not to simply create a geolocator but instead addressed the important issue of privacy preservation. This allowed me to contribute positively to the world and responsible AI development, while still being able to learn about and develop a powerful LVLM. Partaking in this project has taught me so much and I will look back very fondly on the achievements I have made.

A.3 Legal, Social, Ethical and Professional Issues

A.3.1 Legal Issues

I ensured that copyright rules were adhered to for all platforms used. For example, Mapillary grants permission to use and adapt their images for research purposes in their terms of use, licensed under CC-BY-SA. This license requires correct attribution, a link to the license, and clear indication if images are modified. Additionally, all subsequent work using the images should come under the same license if distributed. This does not apply to this project, as the work is not being distributed. However, if the privacy-preserving geolocator were to be distributed publicly, it would have to also be licensed under CC-BY-SA.

All images used for training and validation comply with privacy laws. While Mapillary's images may be crowd-sourced, they have all been meticulously filtered to blur personal information such as faces and car number plates, as detailed in [39].

A.3.2 Social Issues

While I have already explored privacy concerns heavily in the main report, they are further discussed here and broken into social and ethical issues.

Geolocators are useful tools, but they can be misused in many ways such as stalking or surveillance. This project addresses the negative social aspects of a geolocator by reducing pinpoint accuracy where exact location is revealed by text. Location is an important personal information that most people want to protect, but at the current rate, geolocators may become normalised and widely used. Geolocators should not be normalised, and the report intends to highlight the dangers and propose potential solutions. To enforce usage of privacy-preserving geolocators, legal intervention may be needed to ban geolocators that threaten our privacy.

A.3.3 Ethical Issues

Ethical AI development is the main goal of the project, aiming to develop a geolocator which cannot be abused and respects the right for location privacy. While this project does develop a model which is privacy-preserving, I have nonetheless created a working baseline geolocator and documented how it was implemented. This poses an ethical concern: I am potentially teaching other people how they can create a geolocator like mine. Also, the deliverables could potentially be misused in the wrong hands, particularly the baseline model, but also the privacy model with 13% pinpoint accuracy. Therefore, the project directory will be kept private and only shared to my supervisor and assessor.

Image geolocators in the past year have seen alarmingly high success. With powerful models and lots of investment from military organisations, image geolocators are very close to, if not already, being used by big organisations such as governments and military. Despite their potential for misuse, they are still legally permitted, which attacks our ethical rights to not be tracked or located without permission. GeoSpy, for example, is currently being advertised to governments and law enforcement. This poses serious ethical concerns regarding surveillance and military usage. While the research in this project will not directly stop these already existing tools, it highlights the need for stricter ethical and legal standards. I hope that it will contribute to raising awareness and encouraging the responsible development of geolocators.

A.3.4 Professional Issues

To maintain professional standards, I ensured to always adhere to copyright and referencing rules. I thoroughly documented every source used for the report and coding, ensuring all credit is properly given for every aspect of the project. Many implementations, such as the BLIP-2, BERT and ViT models, requested specific papers to be cited when the model is used, and I always followed such guidelines if present in their documentation. Additionally, the code

comments and project report heavily document the model design choices and open-source software used to build the privacy-preserving model. This transparency allows for other people to reproduce and collaborate with the work, which is an important professional principle.

The fundamental goal of the project is to follow a professional code of conduct: to develop AI responsibly with privacy as a priority. I successfully created a privacy-preserving geolocator which not only offers an accurate and useful tool but also respects privacy.

Finally, there is always a potential for bias in the dataset. With 40,000 randomly selected Mapillary images, the geographic distribution could be uneven. Urban areas might be overrepresented while rural areas less so, impacting the model performances in certain environments. However, the primary focus of this project was the implementation and evaluation of privacy-preserving techniques.

Appendix B

External Materials

B.1 Software and Python Libraries

- PyTorch [42]
- Transformers [54]
- EasyOCR [27]
- OpenCV [41]
- NumPy (See Appendix C.1)
- Pillow (See Appendix C.1)
- Requests [47]
- Geopy [19]
- CUDA [40]
- Anaconda [2]

B.2 Pretrained Models

- BERT [14]
- BLIP-2 [30]
- ViT with ImageNet [13,15]

B.3 APIs

- Mapillary [36]

B.4 Tools

- LucidChart [33]

B.4 Online Resources (guides, discussion forums, blogs)

- Cross-attention implementation [55] (used directly)
- See Appendix C for all coding references used to support and guide the implementation.

Appendix C

Code Attribution

This appendix lists all references used for the code implementation of this project, not already cited in the main report. All references are also commented within the code where they were utilised.

C.1 Python Libraries:

Pillow. 2024. *Pillow* (version 10.3.0). [Software]. [Accessed 8 February 2025]. Available from: <https://python-pillow.github.io/>

Numpy. 2024. *Numpy* (version 1.26.4). [Software]. [Accessed 19 January 2025]. Available from: <https://numpy.org/>

C.2 Online Resources:

Alammar, J. 2019. A Visual Guide to Using BERT for the First Time. 26 November. *Visualizing machine learning one concept at a time*. [Online]. [Accessed 12 February 2025]. Available from: <https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/>

avitex and Dunn, M. 2011. Haversine formula in Python (bearing and distance between two GPS points). 06 February. *Stack Overflow*. [Online]. [Accessed 20 February 2025]. Available from: <https://stackoverflow.com/questions/4913349/haversine-formula-in-python-bearing-and-distance-between-two-gps-points>

deptra1 and nathancy. 2019. Blur content from a rectangle with Opencv. 30 September. *Stack Overflow*. [Online]. [Accessed 1 March 2025]. Available from: <https://stackoverflow.com/questions/58163739/blur-content-from-a-rectangle-with-opencv>

GeeksforGeeks. 2024. *How to Generate Word Embedding using BERT?* [Online]. [Accessed 8 February 2025]. Available from: <https://www.geeksforgeeks.org/how-to-generate-word-embedding-using-bert/>

Huang, Y.-J. and user13877663. 2020. Pytorch schedule learning rate. 27 July. *Stack Overflow*. [Online]. [Accessed 26 February 2025]. Available from: <https://stackoverflow.com/questions/63108131/pytorch-schedule-learning-rate>

- KFrank. 2022. Is there a way to combine classification and regression in single model? November. *PyTorch Forums*. [Online]. [Accessed 19 March 2025]. Available from: <https://discuss.pytorch.org/t/is-there-a-way-to-combine-classification-and-regression-in-single-model/165549>
- KFrank. 2023. Custom loss function failure: distance between two points on globe (Haversine Loss). October. *PyTorch Forums*. [Online]. [Accessed 20 February 2025]. Available from: <https://discuss.pytorch.org/t/custom-loss-function-failure-distance-between-two-points-on-globe-haversine-loss/190610>
- Khalusova, M. and Li, J. 2023. Zero-shot image-to-text generation with BLIP-2. 15 February. *Hugging Face*. [Online]. [Accessed 6 February 2025]. Available from: <https://huggingface.co/blog/blip-2>
- LearnPyTorch.io. [no date]. 02. *PyTorch Neural Network Classification*. [Online]. [Accessed 19 March 2025]. Available from: https://www.learnpytorch.io/02_pytorch_classification/
- Louwers, J. 2023. Calculate Geographic distances in Python with the Haversine method. 18 September. *Medium*. [Online]. [Accessed 20 February 2025]. Available from: <https://louwersj.medium.com/calculate-geographic-distances-in-python-with-the-haversine-method-ed99b41ff04b>
- Oppidi, A. and Jha, A. 2025. PyTorch Loss Functions: The Ultimate Guide. 27 January. *Neptune Blog*. [Online]. [Accessed 20 February 2025]. Available from: <https://neptune.ai/blog/pytorch-loss-functions>
- Petroules, J., Fraser, I., Ferrari, M., C, E., Sampada, Nasif, Z. and user1439929. 2010. Regular expression for matching latitude/longitude coordinates? 19 August. *Stack Overflow*. [Online]. [Accessed 8 February 2025]. Available from: <https://stackoverflow.com/questions/3518504/regular-expression-for-matching-latitude-longitude-coordinates>
- ptrblck. 2020. Split dataset into two new datasets (NOT subset). February. *PyTorch Forums*. [Online]. [Accessed 23 February 2025]. Available from: <https://discuss.pytorch.org/t/split-dataset-into-two-new-datasets-not-subset/71328>
- PyTorch. 2017. *Training a Classifier*. [Online]. [Accessed 19 March 2025]. Available from: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- RegExLib. c2001-2025. *Regex search for 'latitude'*. [Online]. [Accessed 8 February 2025]. Available from: <https://regexlib.com/search.aspx?k=latitude+&c=-1&m=-1&ps=20>

- Tam, A. 2023. Building Multilayer Perceptron Models in PyTorch. 08 April. *Deep Learning with PyTorch*. [Online]. [Accessed 18 February 2025]. Available from: <https://machinelearningmastery.com/building-multilayer-perceptron-models-in-pytorch/>
- Tammana, S. 2023. Text Detection Using EasyOCR Python. 28 February. *Sreekar Tammana's Blog*. [Online]. [Accessed 1 March 2025]. Available from: <https://sreekartammana.hashnode.dev/text-detection-using-easyocr-python>
- Thunder and Ivan. 2022. Random split a PyTorch dataset of type TensorDataset. 25 August. *Stack Overflow*. [Online]. [Accessed 23 February 2025]. Available from: <https://stackoverflow.com/questions/73486641/random-split-a-pytorch-dataset-of-type-tensordataset>
- Yadav, A. 2024. Guide to Pytorch Learning Rate Scheduling. 28 October. *Data Scientist's Diary*. [Online]. [Accessed 26 February 2025]. Available from: <https://medium.com/data-scientists-diary/guide-to-pytorch-learning-rate-scheduling-b5d2a42f56d4>
- Zhang, X. 2019. Multi-Layer Perceptron (MLP) in PyTorch. 26 December. *Deep Learning Study Notes*. [Online]. [Accessed 18 February 2025]. Available from: <https://medium.com/deep-learning-study-notes/multi-layer-perceptron-mlp-in-pytorch-21ea46d50e62>

Appendix D

Image Examples

This appendix lists image examples from the validation set used for analysing the geolocators. All Images are from Mapillary [36], licensed under [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/).

D.1 Geolocated Images:

These are examples of images that were precisely geolocated by all models.





D.1 Non-geolocated Images:

These are examples of images that could not be geolocated by all models, returning very bad predictions.

