Image Retrieval for Visual Geolocalization

Project 2 - Group 5

Sepehr Alidousti Shahraki
Department of Control and Computer Engineering
Polytechnic University of Turin
Turin, Italy
S289456@studenti.polito.it

Ali Ghasemi
Department of Control and Computer Engineering
Polytechnic University of Turin
Turin, Italy
S289223@studenti.polito.it

Shadi Nikneshan
Department of Control and Computer Engineering
Polytechnic University of Turin
Turin, Italy
S289716@studenti.polito.it

***Abstract*—In the field of image retrieval for visual geo-localization, the task is to accurately predict the geographical location of an image based on its content. There are many reasons that make image retrieval for visual geo-localization challenging such as image variability, image scale, data sparsity, ambiguity in geo-location information, and computational complexity. In this paper, we experiment with methods such as smart data augmentation, synthetic images as data augmentation, fine-tunning the optimizer parameters, and using ensembles of descriptors obtained from different models.**

Keywords—Image Retrieval, Visual Geo-localization, NetVLAD, GeM pooling

I. INTRODUCTION

Image retrieval for visual geo-localization is a rapidly growing field that has seen significant advances in recent years due to the increasing availability of geo-tagged images and the growing demand for geolocation-based services. The goal of image retrieval for visual geo-localization is to identify the geographic location of an image by matching it with a database of reference images that have known geo-locations. This field has numerous applications in environmental monitoring, urban planning, and tourism, among others.

However, the task of image retrieval for visual geo-localization is challenging due to several factors such as image variability, image noise, image scale, data sparsity, ambiguity in geo-location information, and computational complexity. These challenges require advanced techniques to accurately retrieve relevant images based on their geographic location. To overcome these challenges, researchers have developed various algorithms and methods that utilize image features, geospatial information, and machine learning techniques.

In this paper, we experiment with different approaches to image retrieval for visual geo-localization that leverage the contributions of the GeM pooling [1] and NetVLAD [2] techniques with ensembles, fine-tunning the optimizer parameters, smart data augmentation, and synthetic images as data augmentation. Our approach integrates these techniques to try to enhance the precision and robustness of image retrieval systems.

The GeM pooling technique is used to aggregate feature maps and produce a global representation of an image. This technique helps to overcome the problem of image variability by producing a more robust and discriminative feature representation. The NetVLAD technique is used to learn a compact and discriminative feature representation for images. This technique helps to overcome the problem of data sparsity by learning a more representative feature representation.

In this report, ensembles [3] are used to improve the robustness of image retrieval systems by combining the image descriptors from multiple models. This technique helps to overcome the problem of ambiguity in geo-location information by producing a more reliable prediction. Optimization is a key aspect in deep learning and involves adjusting model parameters to minimize a loss function, measuring the difference between predicted and true outputs, using algorithms such as gradient descent [4] or Adam [5]. The choice of optimization algorithm depends on various factors, including dataset size and model complexity, and has a significant impact on model performance. Smart data augmentation is used to increase the size of the training dataset and to reduce overfitting. This technique helps to overcome the problem of data sparsity by producing a more representative training dataset. Synthetic images are used as data augmentation to increase the diversity of the training dataset. This technique helps to overcome the problem of image variability by producing a more diverse training dataset.

II. RELATED WORKS

A. GeM pooling

GeM (Generalized Mean pooling) is a type of pooling operation used in deep learning architectures, particularly in convolutional neural networks (CNNs). Conventional pooling operations like max pooling and average pooling compute the maximum and average of the values within a pooling window respectively. On the other hand, GeM pooling generalizes this concept and computes a power mean of the values within the pooling window, where the power can be any real number greater than or equal to 0.

The formula for GeM pooling is given by [1]:

$$GeM(X) = \left(\frac{1}{|X_k|} \sum |x|^{p_k}\right)^{\frac{1}{p_k}}$$

where X is the input tensor and p are a learnable hyperparameter. The default value of p is 3, which gives good results in most cases. By allowing the value of p to be learned, GeM pooling provides a more flexible pooling operation compared to the max and average pooling, which can adapt to different image characteristics.

B. NetVlad

NetVLAD is a deep learning model designed for identifying specific locations in computer vision using weak supervision, specifically designed for place recognition. Place recognition refers to the task of identifying the same location in different images or videos. This is an important problem in various applications like autonomous navigation, mapping, and robotics.

NetVLAD is based on the concept of VLAD (Vector of Locally Aggregated Descriptors) [2,6], a popular image representation method for place recognition. In VLAD, local descriptors of an image are first extracted and then aggregated into a compact global representation by computing the residuals between the descriptors and a set of cluster centers. NetVLAD extends this idea by using a deep neural network to learn the cluster centers and the residuals, making it possible to handle large variations in appearance and viewpoint.

NetVLAD has been shown to produce state-of-the-art results in various place recognition benchmarks, demonstrating its effectiveness in handling large variations in appearance and viewpoint. The ability to learn both the local descriptors and the global aggregation makes NetVLAD a powerful tool for place recognition in computer vision.

C. GAN

Generative Adversarial Networks (GANs) [7] are a class of deep learning models and are designed to generate new data samples that are similar to a given training set. They consist of two components: a generator network and a discriminator network. The generator network is trained to generate new data samples, while the discriminator network is trained to distinguish between the generated samples and the real data samples from the training set.

GANs have been applied to a wide range of tasks, including image synthesis, style transfer, and data augmentation. They have shown remarkable results in generating high-quality images, such as photographs of faces and animals, and have been used in various fields, including computer graphics, computer vision, and robotics. GANs have also been used to generate new data samples in medical imaging, such as generating 3D brain scans or synthesizing new medical images from limited training data.

D. ResNet

ResNet (Residual Network) [8] is a deep learning architecture introduced to solve the problem of vanishing gradients in very deep neural networks. The ResNet architecture is based on the idea of residual connections, which allow information to bypass multiple layers and be directly passed from earlier layers to later layers. The residual connections help prevent the vanishing gradients problem, allowing for much deeper networks to be trained effectively.

ResNet-18 [8] is a specific instantiation of the ResNet architecture, consisting of 18 layers (including convolutional layers, pooling layers, and fully connected layers). The basic building block of a ResNet is the residual block, which consists of two or more layers followed by a residual connection. The residual connection allows information to bypass one or more layers and be directly passed from earlier layers to later layers. This design helps the network learn the residual mapping, instead of the complete mapping, which makes the optimization problem easier.

E. Adam & AdamW Optimizers

Adam (Adaptive Moment Estimation) [5] and AdamW [9] are optimization algorithms for training deep neural networks. They are extensions of the traditional stochastic gradient descent (SGD) optimization method that combines moving averages of the first and second moments of the gradient updates to dynamically adapt the learning rates for each parameter.

AdamW was introduced as a variant of Adam and is designed to improve the generalization performance of deep neural networks by adding weight decay, a regularization technique that penalizes large weight values. In AdamW, the weight decay term is combined with the gradient updates, leading to more robust and generalizable models.

One advantage of Adam and AdamW over other optimization methods is that they require less fine-tuning of the learning rate, as the adaptive learning rates help to automatically balance the convergence speed and stability of the optimization process. Additionally, both Adam and AdamW have been shown to be highly effective in optimizing deep neural networks for various tasks, including image classification, natural language processing, and reinforcement learning.

The AdamW optimizer is a variant of the Adam optimizer that includes weight decay regularization. The update rule for the AdamW optimizer can be written as [9]:

$$t = t + 1$$

$$m_t = \beta 1 \cdot m_{t-1} + (1 - \beta 1) \cdot \nabla \theta J(\theta)$$

$$v_t = \beta 2 \cdot v_{t-1} + (1 - \beta 2) \cdot \left(\nabla \theta J(\theta)\right)^2$$

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

$$\theta = \theta - \alpha \cdot \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon} - \lambda \cdot \theta$$

where t is the current iteration step, β1 and β2 are hyperparameters that control the exponential decay rates of the running averages of the gradient and squared gradient, m_t and V_t are the running averages of the gradient and squared gradient, $\widehat{m_t}$ and $\widehat{v_t}$ are the bias-corrected versions of m_t and V_t, ε is a small constant used for numerical stability, and λ is the weight decay hyperparameter. Note that in the above formula, the weight decay term $\lambda \cdot \theta$is applied to the parameters directly, rather than to the gradient like in other weight decay regularization methods. This results in a more effective and efficient regularization.

III. PROPOSED METHOD

A. Overview

We trained our models using ResNet18 as the backbone architecture and experimented with Gem pooling and NetVLAD aggregators. Due to the limitations of time and computational power, we avoided testing all proposed methods on GeM pooling and NetVLAD aggregators. For most of the cases, we used NetVLAD instead of GeM pooling since it performed better in our first experiments. The comparison of NetVLAD and GeM pooling for the baseline model is available in Table 6, and the overall comparison of proposed methods is shown in Table 4. In this section of the report, we are going to explain what we have done regarding each extension.

B. "Smart" Data Augmentation

Data augmentation is a technique in computer vision that involves artificially increasing the size of a training dataset by applying various transformations to the original data samples. The goal of data augmentation is to increase the diversity of the training data, reducing the risk of overfitting and improving the generalization performance of deep learning models. Common data augmentation techniques for images include flipping, rotation, scaling, translation, adding noise, and color adjustments. The idea is to apply these operations to the original images to generate additional training samples, creating a larger and more diverse training set.

"Smart" Data Augmentation is a type of data augmentation technique that aims to improve the performance of deep learning models in computer vision tasks. By using an intelligent approach to determine the best augmentation operations to apply to the training data, "Smart" Data Augmentation can enhance the diversity and quality of the training data, leading to better generalization performance and reduced risk of overfitting.

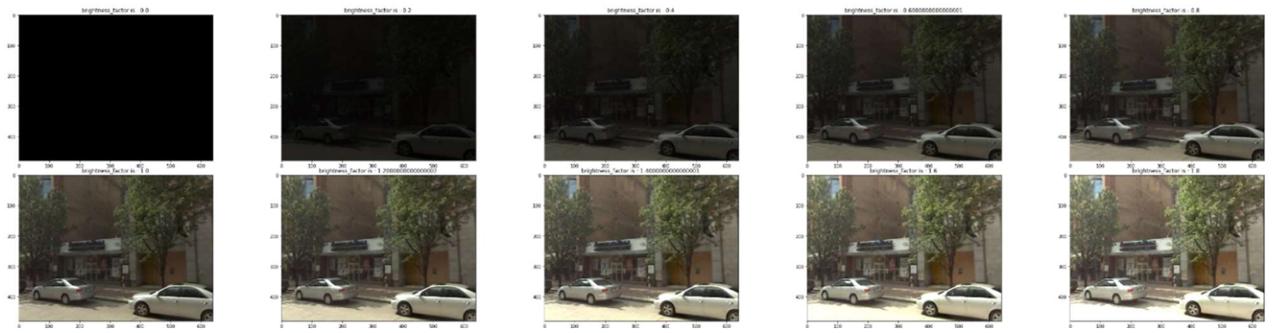
In this study, the brightness of images was adjusted using PyTorch adjust brightness function to improve the performance of the model when analyzing images of cities at night. The need for this adjustment arises from the fact that images taken at night often exhibit low brightness levels, which can negatively impact the accuracy of the model.

We conducted three experiments to determine the optimal range for brightness values in image processing. In the first experiment, we utilized a pre-trained base model (using NetVLAD aggregator) and fine-tuned it with brightness values in the range of [0.1, 0.3]. In the second experiment, we repeated the fine-tuning process with the range of [0.3, 0.9]. In the third experiment, we trained the model from scratch with brightness values in the range of [0.3, 0.9]. It is important to note that training the model from scratch with brightness values in the range of [0.1, 0.3] was not effective as the images were too dark and the model was unable to learn the essential information from the images. After evaluating the models on the Tokyo-night dataset, we found that the best performance was achieved with the pre-trained model fine-tuned with the brightness values in the range of [0.1, 0.3]. We further tested this optimized model on other datasets and the results are presented in Table 4. You can see the results of these three experiments on Tokyo-night dataset in Table 1.

Table 1

Experiment No.	Brightness Range	Result
1	[0.1, 0.3]	R@1: 21.0 R@5: 37.1
2	[0.3, 0.9]	R@1: 20.0 R@5: 36.2
3	[0.3, 0.9]	R@1: 12.4 R@5: 24.8

In our experiments, we determined that using brightness values outside the range of [0.1, 1] would result in images that were either too dark or too bright.


brightness_factor is : 0.0
brightness_factor is : 0.2
brightness_factor is : 0.4
brightness_factor is : 0.6000000000000001
brightness_factor is : 0.8
brightness_factor is : 1.0
brightness_factor is : 1.2000000000000002
brightness_factor is : 1.4000000000000001
brightness_factor is : 1.6
brightness_factor is : 1.8
Figure 1

As shown in Figure 1, this highlights the importance of choosing appropriate brightness values. To prevent bias, we randomly selected brightness values within the specified range during experimentation.

C. Ensembles

Ensemble learning is a machine learning technique that combines multiple models to produce a more robust and accurate prediction. In computer vision and image retrieval, ensemble learning can be used to concatenate image descriptors, which are numerical representations of images, to produce a more descriptive and informative feature vector for the image. The idea behind this is to leverage the strengths of multiple image descriptors and combine them into a single representation that if it's more powerful than any single descriptor or not.

Out of the models trained, we selected two models (one using GeM pooling and the other using NetVLAD) and used them to generate image descriptors. To further test the performance, we combined the descriptors from these two models by concatenating them. The resulting ensemble was then tested on different datasets to evaluate its overall performance. The goal of this experiment was to investigate the potential benefits of combining multiple models for improved accuracy and robustness in computer vision tasks.

D. Synthetic Images as Data Augmentation

Synthetic images as data augmentation using GANs refers to the process of using Generative Adversarial Networks (GANs) to generate new, artificial images to augment an existing dataset. This can be useful in cases where the original dataset is limited in size or diversity, and additional data is needed to train machine learning models effectively.

To generate a synthetic dataset, we used the pre-trained model of the GAN network available in this repository to create night images from the Pitts30k dataset. We first computed the average brightness of each image (value of brightness divided by the number of pixels) and applied an experimental brightness threshold to classify the images as either day or night. Then, night images were generated from the day-classified images. After the end of this process, we had 1345 new images in the dataset which were generated by the GAN model. The reason why the number of generated images is way less than the images in the training dataset is because, the model can learn more features from day images since the generated night images are darker and there are less features to learn. You can see some samples of generated images using GAN in figure 2.

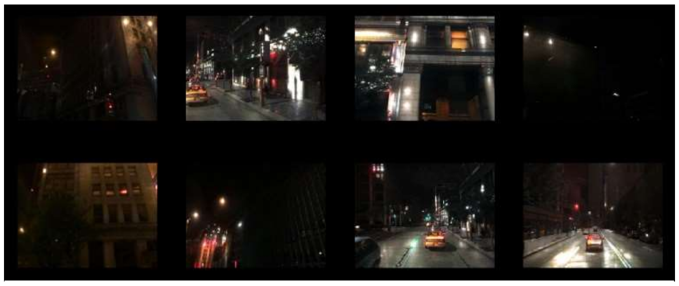


Figure 2

For evaluating the effectiveness of the generated synthetic dataset, we conducted three experiments. In the first experiment, we used a pre-trained model with the NetVLAD Aggregator and continued training on the synthetic dataset. The second experiment involved only training on the synthetic dataset without any prior training. Lastly, in the third experiment, we trained the model from scratch using a combination of both synthetic and real datasets. All models were tested on the Tokyo night dataset. The best result belongs to the last experiment which produced better results for some datasets than the normal model. You can see the results of these three experiments on Tokyo-night dataset in Table 2.

Table 2

Experiment No.	Explanation	Result
1	Baseline model+ resume with synthetic images	R@1: 21.9 R@5: 37.1
2	Train just with synthetic dataset	R@1: 9.5 R@5: 20.0
3	Combination of real and synthetic dataset	R@1: 47.6 R@5: 69.5

E. Optimizer

Optimizers are essential algorithms in training deep learning models that are used to adjust the parameters of the model to minimize the loss function. The loss function measures the difference between the model's predictions and the ground truth, and the goal of the optimizer is to adjust the model parameters such that the loss is minimized. Some popular optimizers used in deep learning include Stochastic Gradient Descent (SGD), Adaptive Moment Estimation (Adam), and Root Mean Squared Propagation (RMSprop).

When training machine learning models, it's important to carefully tune the hyperparameters of the learning rate and weight decay to achieve efficient optimization and good generalization. The learning rate determines the step size of updates made to the model parameters during training, and weight decay adds a penalty term to the loss function to encourage smaller weights and a simpler model. However, setting the learning rate too high can lead to aggressive updates and suboptimal results, while a low learning rate can slow down convergence. Similarly, a high value for weight decay can lead to over-regularization and under-fitting, while a low value can result in under-regularization and over-fitting. To avoid these issues and find a good balance, it's crucial to carefully adjust the values of the learning rate and weight decay.

In Figure 3 you can see how the triplet loss changes over each epoch, some experiments were halted hence there was not much improvement in triplet loss. As you can see in Figure 3, we decided to choose case 6 parameters as the best of our experiment and test other models with the same parameters. In Table 3 you can check what parameters were in each experiment case.

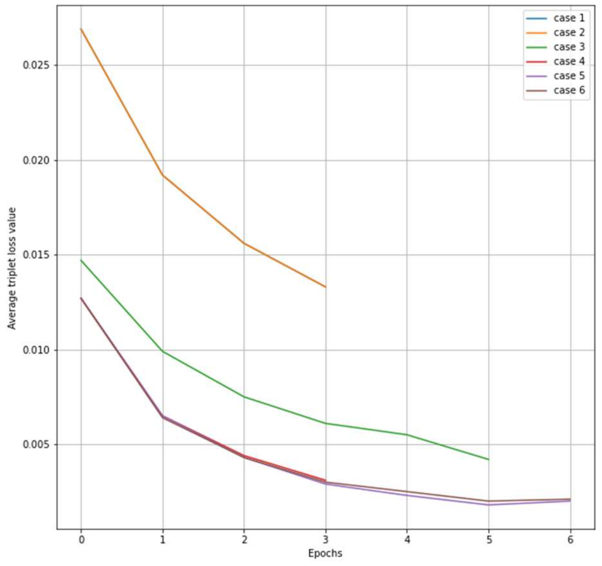


Figure 3

Table 3

Case No.	Parameters
1	Learning rate= 0.00001 Weight decay = 0.01
2	Learning rate= 0.00001 Weight decay = 0.05
3	Learning rate= 0.001 Weight decay = 0.01
4	Learning rate= 0.0001 Weight decay = 0.05
5	Learning rate= 0.0001 Weight decay = 0.1
6	Learning rate= 0.0001 Weight decay = 0.3

It is worth mentioning the values for the first and the second case of the experiment overlap each other, and you can only see one of them in Figure 3.

IV. EXPERIMENTAL RESULTS

In this part of the report, the outputs of the models for different extensions that we implemented are available.

A. Datasets and Experimental settings

We used three datasets for our experiments: pitts30k [10], a visual place recognition dataset of images from Pittsburgh, USA, and two other datasets - San Francisco small (sf-xs) [11] and Tokyo-small (Tokyo-xs), which are subsets of San Francisco eXtra Large [12] and Tokyo 24/7 [13] respectively. pitts30k is also part of a larger dataset called pitts250k [10]. All these datasets contain Google Street View images, labeled with GPS coordinates and headings. We also created a dataset called Tokyo-night by selecting only the night images from the Tokyo-small dataset. Our models were trained on pitts30k and tested on all four. In Table 5 you can see a summary of the used datasets.

In order to understand better how images of pitts30k dataset sere taken we used the geo-tags of all images and marked the map of Pittsburgh city, the results were fascinating, and it can give a better understanding of the distribution of images on train, val, test directories. You can see the result in Figure 4.

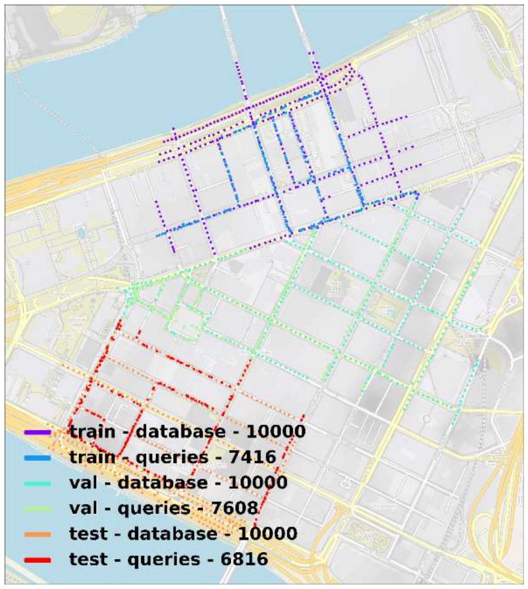


Figure 4

NetVLAD aggregator has been used in all of the models since it performed better in our early experiment with the baseline code without adding any changes and since we were limited in time and computational power, using GeM pooling for the other scenarios has been avoided. To avoid time consumption, we started by evaluating smart data augmentation and synthetic image experimental models on the Tokyo-night dataset. This was a legitimate choice as the dataset is smaller in size compared to others and due to the nature of the images of the dataset and the fact that the goal was to make the model perform better on the night images. And then afterward we proceeded to test the other datasets on the better-performing model.

In our experiments, we used the "partial" mining method, as it was found to be more effective than the "random" method according to [11]. Although the "full" dataset mining method has slightly better performance [11], its high space and time complexity meant that we could only use the "partial" method to overcome limitations in time and computational resources.

Table 4

		Test dataset				
Method	Aggregator	Pitts30k	sf-xs	Tokyo-xs	Tokyo-night	Training dataset
Smart data Aug	NetVLAD	R@1: 86.1 R@5: 92.7	R@1: 35.1 R@5: 50.0	R@1: 54.9 R@5: 70.2	R@1: 21.0 R@5: 37.1	Pitts30k
Synthetic images	NetVLAD	R@1: 84.9 R@5: 92.5	R@1: 34.7 R@5: 48.4	R@1: 67.3 R@5: 82.5	R@1: 47.6 R@5: 69.5	Pitts30k + GANs generated
Ensembles	NetVLAD +GeM	R@1: 85.6 R@5: 92.9	R@1: 35.0 R@5: 50.6	R@1: 54.0 R@5: 70.5	R@1: 22.9 R@5: 41.9	Pitts30k
Optimizer	NetVLAD	R@1: 84.8 R@5: 92.3	R@1: 30.8 R@5: 45.3	R@1: 48.9 R@5: 67.0	R@1: 17.1 R@5: 38.1	Pitts30k

Table 5

Dataset	# train/val	# test	size (GB)
Pitts30k	17416/17608	16816	1.84
Sf-xs	-	28191	1.24
Tokyo-xs	-	13086	0.70
Tokyo night	-	12876	0.69

B. Results

In this section of the report, the overall experimental outputs which were achieved by performing different experiments using different models for various extensions will be shown in Table 4. As shown in the table, the best outputs for each extension have been mentioned.

Aggregator	Pitts30k	Sf-xs	Tokyo-xs	Tokyo night
GeM	R@1: 76.3 R@5: 88.8	R@1: 12.0 R@5: 26.8	R@1: 34.6 R@5: 50.8	R@1: 8.6 R@5: 26.7
NetVLAD	R@1: 85.9, R@5: 92.7	R@1: 36.6 R@5: 51.0	R@1: 59.0 R@5: 73.0	R@1: 27.6, R@5: 44.8

Table 6

All these experiments have been explained in detail and the results were provided. The models have improved in some areas and haven't in some other areas but due to the limitation of time, computation power any further enhancements were not possible. Also using only one specific dataset for training or not using different backbone architectures can help with the lack of improvements in some cases.

V. CONCLUSION

In this paper, we examine different approaches to enhance the performance of a renet18-based model for image retrieval for visual geo-localization tasks. We have used pitts30k to train and used different datasets to test our models and mostly used the NetVLAD aggregator in our models to perform better. We have experimented with different parameters and techniques to improve the outputs of the models. Techniques such as smart data augmentations, use of syntenic images, ensembles, and fine-tuning the optimizations parameters of AdamW optimizer.

REFERENCES

[1] Radenović, F., Tolias, G., & Chum, O. (2018, July 10). *Fine-tuning CNN image retrieval with no human annotation*.

[2] Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. (2016, May 2). *NetVLAD: CNN architecture for weakly supervised place recognition*.

[3] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In Proc. CVPR, 2010.

[4] Ruder, S. (2017, June 15). *An overview of gradient descent optimization algorithms*.

[5] Kingma, D. P., & Ba, J. (2017, January 30). *Adam: A method for stochastic optimization*.

[6] R. Arandjelović and A. Zisserman. All about VLAD. In Proc. CVPR, 2013

[7] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). Generative Adversarial Nets. Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 10). *Deep residual learning for image recognition*.

[9] Loshchilov, I., & Hutter, F. (2019, January 4). *Decoupled weight decay regularization*.

[10] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla and J. Sivic, NetVLAD: CNN architecture for weakly supervised place recognition, TPAMI 2018

[11] Berton, G. and Mereu, R. and Trivigno, G. and Masone, C. and Csurka, G. and Sattler, T. and Caputo, B. “Deep Visual Geo-localization Benchmark.” CVPR (2022).

[12] Berton, G. and Masone, C. and Caputo, B. “Rethinking Visual Geo-localization for Large-Scale Applications.” CVPR (2022).

[13] A. Torii, R. Arandjelovi´c, J. Sivic, M. Okutomi, and T.Pajdla. 24/7 place recognition by view synthesis. IEEETransactions on Pattern Analysis and Machine Intelligence,40(2):257–271, 2018.