Enhancing Visual Geolocalization

Augmentation, Aggregation, and Domain Adaptation Strategies for Improved Accuracy

Enrico Chiavassa
s302626@studenti.polito.it

Valerio Gallo
s302671@studenti.polito.it

Emanuele Pinna
s296455@studenti.polito.it

June 22, 2023

Abstract

Visual geolocalization is the task of estimating the geographical position of an image by comparing it to the ones in a large dataset. The current state-of-the-art approach is a mining-free method named Cosplace, which exploits the features extracted by a Geolocalization classifier. In this paper we conduct extensive testing in order to find improvements in key areas, including data augmentation, aggregation, backbones and domain adaptation. Specifically, we delve into an in-depth analysis of the impact of different lighting augmentations and implement a street-view Random Erasing technique. We then explore the use of enhanced backbones, both convolutional (EfficientNet) and hybrid (MobileViT), and aggregation methods such as Adaptive Pooling and Attention Pooling. Furthermore, we tackle the challenge of domain shift by introducing a secondary self-supervised task that facilitates the learning of more robust features by the network.

1. Introduction

Visual geolocalization is a challenging computer vision task that involves estimating the geographical location of an image by matching its visual content to a large-scale reference database. In this paper, we focus on improving the performance of visual geolocalization systems by investigating various strategies. We begin by adopting the CosPlace method [4] as our baseline. This method's key idea is to train the model with a classification task and then use it at inference to extract discriminative descriptors for the retrieval. The main advantage is that samples mining is not required and therefore scalability is much less of an issue.The reference dataset is SF/XS, a subset of the well known SF/XL [4], and it is used for training and testing. In order to verify the generalization property of our models we also test them on Tokyo/XS, a subset of Tokyo 24/7 [15]. To enhance the performance of the baseline, we systematically analyze the impact of different augmentation techniques. In particular, we investigate the effect of illumination parameters adjustments and implement a street-view specific Random Erasing technique. The former help to tackle the issue of lighting changes while the latter deals with occlusions. Furthermore, we delve into the realm of backbone architectures beyond the traditional ResNet and VGG models. We experiment with EfficientNet models as well as the MobileViT_V2 hybrid models. By exploring these alternative backbones, we investigate the potential of well-established models in enhancing the performances. To improve the aggregation of visual features, we test various pooling techniques beyond the original GeM pooling [13]. We investigate a simple adaptive pooling, an attention-based pooling, and introduce the recent MixVPR algorithm. These techniques vary in complexity and offer different approaches to capture more discriminative and representative information from the visual features.

In order to achieve domain adaptation and robustness against domain shifts, we introduce a self-supervised learning task. We propose a double loss framework that combines the loss of the classification task and an invariant loss on the descriptors. By jointly optimizing these losses, we aim to leverage the benefits of self-supervised learning for domain adaptation. Finally, we study the impact on performances of different optimizers and schedulers. The former deal with weight decay in various ways, while the latter enforce a dynamic reduction of the learning rate. We try to identify the most effective combinations for enhancing our models. Through an extensive experimental evaluation, we provide insights into the effectiveness of each component and highlight the improvements achieved by our proposed modifications. The results of our study contribute to the advancement of visual geolocalization systems, especially Cosplace, paving the way for more accurate and robust localization in real-world scenarios. The code for the experiments can be found at https://github.com/Emanuelepinna00/MLDL_project.

2. Related works

In the realm of visual geolocalization, various methods that have been proposed over the years are based on the concept of contrastive learning. Although their performances often reached the SOTA, these models are usually trained using a triplet loss, which heavily depends on sample mining. Upward scalability is therefore an issue. Notable examples are NETVLAD [2] and VLAD+CRN [9], which are based on the idea of leveraging vector of locally aggregated descriptors (VLAD) encoding to achieve robust global representations. Another approach to visual geolocalization is to consider it as classification task, dividing the geographical area of interest in cells. The underlying assumptions are that images belonging to the same region may share some semantic elements, for example architectural style, and that learning those similarities is the way to achieve good classification results. This models are easily scalable to huge datasets, but their precision is often sub-par compared to contrastive methods. One reason for that is the fact that cells are typically very large and include a high number of different scenes. **Cosplace** is the first classification technique that reaches SOTA performances in the geolocalization task. This is done by assuming availability of dense data and defining the classes in a way that guarantees the presence of one single scene in each of them. Moreover, during training, at each epoch only a group (called *Cosplace group*) of non-adjacent classes are used. This is done because the geographical subdivision in cells may create some ambiguity by collocating extremely similar scenes in different (but adjacent) classes because their position differs by a couple of meters. By taking far away classes we solve this inconvenience. The full explanation for how the groups are formed is in [4]. At inference time, the classifier is used to the extract the descriptors of the queries for the image retrieval phase.

3. Datasets

The two datasets used in this paper are San Francisco eXtra Small (SF-XS) and Tokyo eXtra Small (Tokyo-XS).

- San Francisco eXtra Small (SF-XS) Dataset: It is a subset of the SF-XL dataset [4] containing about 100k images and it is used for training, validation, and testing purposes. SF-XS is specifically designed to provide a manageable dataset size for experimenting with the Cosplace method. It contains geotagged images of diverse urban scenes and landmarks from the San Francisco area. The image density plot is in Fig. 1.

- Tokyo eXtra Small (Tokyo-XS) Dataset: It is a subset of the Tokyo 24/7 dataset [15] containing 13k images and it is exclusively used for testing. Tokyo-XS focuses on geolocalized images from Tokyo, Japan. This dataset encompasses various scenes, including iconic landmarks, cityscapes, and cultural sites within the bustling metropolis.

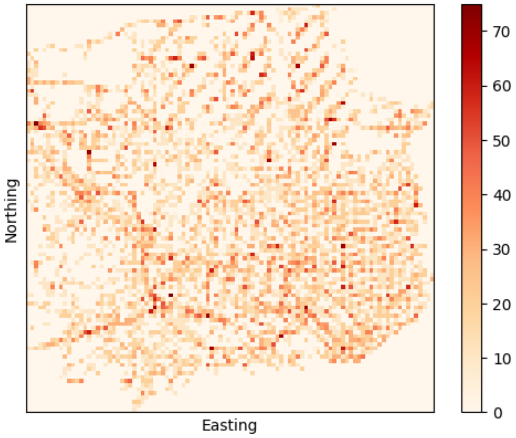

Figure 1. Heatmap of the images in SF/XS. The dataset covers an area of about 140 km^2 almost uniformly. Lower densities areas corrispond mainly to parks.

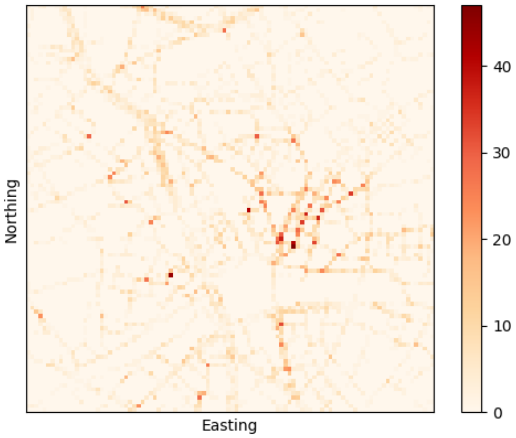

Figure 2. Heatmap of the images in Tokyo/XS. The dataset covers an area of less than 2 km^2 with high sparsity

The evaluation process involves measuring the accuracy of geolocalization by determining if the ground truth location of a query image falls within 25 meters of the top retrieved database images, and emphasizes the importance by reporting recall@N values (specifically recall@1 and recall@5) to quantify the percentage of queries where the top 1 or top 5 predictions are within the 25 meters threshold.

4. Proposed Methods

4.1. Data Augmentation

Data augmentation allows to enlarge the training set by adding some images that are a modified version of the orig-

inal ones. In our work we conducted some tests focusing on the variations one would observe in a different shot of the same scene. In this code a transformation is applied to each image with a fixed probability, thus the batch is composed of some original images and some modified ones, which are substituted to the genuine one and not added. The concept of "augmentation" comes from the fact that each image in the training set is taken more than once, therefore it will be occasionally altered and occasionally not. For example with a setting of 25 epochs, 250 iterations per epoch and a batch size of 64, which is the configuration in which we conducted the majority of our tests, each image is taken 6.7 times on average. It is evident that this type of augmentation would require a substantial number of epochs and iterations to yield significant improvements. With the setups we are limited to there is a risk that the transformations may only result in noise rather than a true addition of information. This explains the fact that we do not observe a significant improvement in performances with only augmentations. We test some transformations and focused on random erasing and variations on the illumination conditions, that are further explored in the following two subsections.

4.1.1 Analysis of illumination

The various lighting conditions in which a photo can be captured make this type of data augmentation of primary importance. In the code this type of modifications are made by the class `DeviceAgnosticColorJitter` that allows to set four parameters: *brightness, contrast, hue* and *saturation*. By setting for example a value *x* for the brightness, a random change of the picture brightness will be performed with a factor uniformly chosen in the interval

$$[max(0, 1 - x), 1 + x] \tag{1}$$

We conducted some tests isolating each of the four parameters to see how they affect the learning process. The issue lies in the fact that in some cases, high values of these parameters result in overexposed or underexposed images, that lose their informativeness. As we will show in Sec. 5, this kind of augmetation did not lead to significant improvements in performances, probably owing to the limited number of epochs we are constrained to use by computational limitations. In Fig. 3 we show an example of how these transformations affect a random image.

4.1.2 RandomErasing

One of the many issues a computer vision model has to face is occlusions: if an images is taken in a realistic setting it is likely that not all elements will be clearly visible. Some may overlap or simply be covered up by something else. The Random Erasing method for data augmentation was

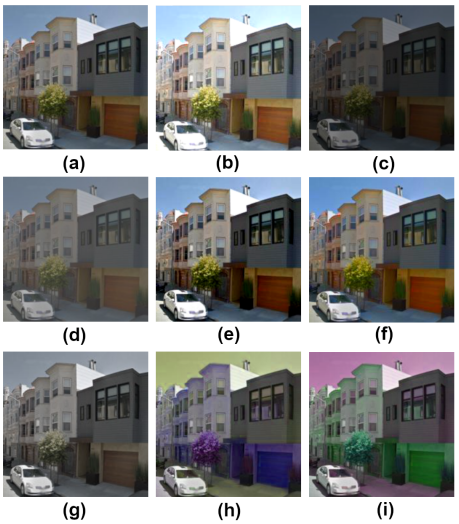

Figure 3. Example of color jitter. Image (a) is the original one, images (b) and (c) are obtained changing the brightness, images (d) and (e) changing the contrast, images (f) and (g) changing the saturation and images (h) and (i) changing the hue.

first introduced in [18] as a lightweight technique to improve the robustness of CNN models to this problem. During the training phase, a random rectangular region of some images is erased by setting its pixel to fixed values. The goal is to make the network effectively handle test samples that lack complete information. Proposed extensions to this method have been presented in [12] and in [17]. In Image GeoLocalization occlusions have a different aspect to them: time. Most datasets are composed of pictures taken over years, if not decades, so if an element is covering up informative features of a place but its position, shape and appearance are constant over time, then it may very well provide informative features of its own. For example a large tree may hide some parts of a building, but its presence is by itself a way to distinguish between that place and other ones. We therefore define as occlusions all objects that appear in an image but are not characteristic of the depicted place over a long period of time, such as cars, buses, bikes and pedestrians. It is important that models learn to ignore such distractions, but in our experiments we found instances of wrong predictions caused by the presence of vehicles. (see Fig. 4)

The classic Random Erasing technique is a fitting solution to this issue, but we believe it can be improved upon. It is important to consider the nature of the images in the dataset: since they are all taken from Google Street View, it is reasonable to suppose that most occlusions are not uniformly distributed on the pictures, but rather are concentrated in the lower half. In order to verify this hypothesis we utilize the pre-trained object detection model $yolov5\_x$ on 3000 images randomly sampled from the sf/xs dataset, restricting

Figure 4. Example of a misclassification. The query image (on the left) contains some vehicles, like the incorrect match (middle), while the correct match (on the right) does not.

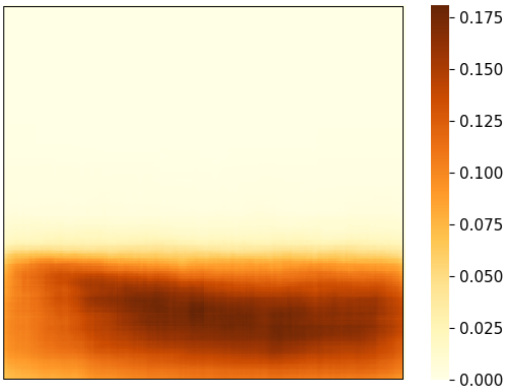

Figure 5. Average occlusion heatmap.

the search to the following classes: *person, bicycle, car, motorcycle, bus, train and truck*. For each picture we generate the occlusions heatmap with the following steps:

1. Define a zero matrix matching the size of the picture.

2. For every object detected, update the matrix entries corresponding to the pixels in the bounding boxes using the following:

$$new\ value = 1 - (1 - x) * (1 - c) \qquad (2)$$

   where x is the current value for a pixel and c is the confidence of the detection.

Assuming that the confidence is a good estimate of the probability of the detection being correct, then the final value for each pixel will be one minus the probability that all detections are wrong.

The average heatmap is shown in Fig. 5. This result suggests that we should not focus on all the images, but only where occlusions are more frequent. We therefore introduce **Hard Random Erasing**, a Random Erasing variation that chooses the region to erase from a uniform distribution over the lower half of images. As a more moderate approach we also introduce **Soft Random Erasing** which follows a 3 to 1 ratio between lower and upper halves.
Numerical experiments proves that these novelties outperform the standard RE.

4.2. Backbones

The correct choice of the backbone of a deep learning algorithm is often hard, expecially when computational resources are limited and sheer size is not a viable option. In the original paper [4] the authors tested a wide array of ResNets of different sizes and VGG-16, which achieved the best performance. We test two other families of networks that have shown great potential across many tasks: EfficientNets and MobileViTs.

EfficientNets The well-known EfficientNets were firstly introduced in [14] with the idea that convolutional neural network dimensions, depth, width and resolution, should not be scaled up independently but rather in a carefully planned manner. This innovation allowed for higher computational efficiency and multiple experiments have proven that these models consistently outperform ResNets, even if the latter use considerably more parameters.

MobileViT Even if they were originally introduced for natural language processing, Transformers have emerged as a powerful approach for computer vision tasks [7]. Their ability to capture long-range dependencies and enable efficient parallel processing lead to state-of-the-art performances in multiple settings. However, most implementations are far from being lightweight. Pytorch's pre-trained vision transformers range from 90 to 600 milions parameters, too many for our limited resources. One of the reasons for this is the lack of local inductive bias [7] [16], one of the founding concepts of CNN. Some exceptions do exist, and one of them, called MobileViT, is proposed in [10], in which the authors define a hybrid model between a transformer and a CNN. This allows the model to retain some inductive bias while also learning global representations. For this reasons the size of MobileViT (and MobileViTV2 [11]) is comparable to lightweight CNNs. Our experiments indicates that, although powerful, this model may suffer from domain changes.

4.3. Aggregation

The aggregation layer is the last part of the model. It receives the output of the convolutional layers and returns the vector of descriptors. In most basic implementations of CNNs, this is composed of a pooling layer, such as average pooling, and a fully connected layer. Generalized Mean (GeM) pooling is one promising novelty introduced in [13] and effectively utilized in [4]. We investigate the performances of various aggregation methods and prove through extensive testing that they outperform GeM in both datasets.

Attention Pooling Attention pooling is an aggregation mechanism used in Deep Learning models to assign weights

to input elements based on their importance. It creates a context vector by dynamically focusing on relevant parts of the input sequence. This is achieved by a simple 1x1 convolution that introduces learnable parameters that can adjust the importance of each channel and pixel in the feature map.

Adaptive Pooling Adaptive pooling is a flexible technique in deep learning that dynamically adjusts the pooling operation based on the input's characteristics. Unlike traditional pooling methods, which needs stride and kernel sizes as arguments, adaptive pooling only requires the desired output size and autonomously computes all required dimensions. For our task we tested adaptive averaging pooling.

MixVPR The MixVPR method, introduced in [1], is built on the idea that aggregation should not focus on local features but rather take advantage of the ability of fully connected layers to automatically aggregate features in a holistic way. The propose architecture truncates the CNN and takes the output of an intermediate layer as the input for a sequence of MLPs, that are responsible for the aggregation. The resulting array is then projected onto lower dimensional spaces in order to obtain descriptors of the desired length. This approach was tested with different numbers of MLPs but did not produce good results. The details are discussed in Sec. 5.

4.4. Optimizer and Scheduler

Cosplace algorithm uses Adam optimizer by default. We combined it with two different schedulers, ReduceLROnPlateau and CosineAnnealingLR, and in addition we tested two different optimizers, AdamW and SGD.

4.4.1 Optimizers

Adam Adam [6] is an optimization algorithm commonly used in deep learning. It consists of a method for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. It adjusts the learning rates based on the magnitude of gradients and incorporates momentum to persistently move in the direction of gradients.

AdamW AdamW [8] is an optimization algorithm that extends the Adam optimizer by incorporating a modified weight decay formulation. Unlike traditional Adam, which applies weight decay as a penalty term, AdamW decouples weight decay from the learning rate updates. This modification helps prevent the learning rates from being affected by weight decay, leading to improved stability, convergence, and generalization performance in training neural networks.

Stochastic Gradient Descend SGD (Stochastic Gradient Descent) is a popular optimization algorithm for deep learning. It updates the model's parameters based on mini-batches of training data, iteratively minimizing the loss function. The issue with SGD is that it can oscillate and get stuck in shallow local optima.

4.4.2 Schedulers

ReduceLROnPlateau ReduceLROnPlateau is a scheduler provided by Pytorch which reduces the learning rate multiplying it by a chosen factor if a selected quantity, the loss in our case, does not decrease for a fixed number of epochs. Even a threshold can be chosen arbitrarily.

CosineAnnealingLR CosineAnnealingLR is a scheduler provided by Pytorch as well which reduces the learning rate in accordance with

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{T_{cur}}{T_{max}}\pi\right)\right) \quad (3)$$

where η_{max} is the initial learning rate, T_{max} is the maximum number of iterations to be set and T_{cur} is the number of epochs performed since the last restart.

4.5. Self Supervised Strategies

In this section we delve into the integration of a secondary task within a visual recognition framework to enhance the performance and robustness of the system. Specifically, we take inspiration from the architecture presented in [5], where a single CNN is trained on a primary task and a secondary, self-supervised task, specifically built to encourage the network to focus on domain invariant features, such as shapes and spatial correlation. In our implementation we keep Cosplace's classification as the main objective and we add an auxiliary one. There are a wide range of possible losses that one can use, but we chose VicReg loss [3] for its general lack of constraints. In particular it seems fitting with Cosplace since neither require any sample mining nor any other complex operation. This loss is a combination of an invariant loss, a variance loss and a covariance loss. The first is the simple mean square error between the compared predictors. The second is computed using the variance of every variable of the predictors across the batch and forces them to have high variability. The third is computed using the correlation of each pair of variables of the descriptors and forces them to be uncorrelated. Our architecture follows this steps: for every batch of images another batch is generated by augmenting the original images. For our test we focused on illumination changes and random erasing. Both batches are fed to the network and descriptors are returned. All of them are used to compute the VicRec loss, while the one corresponding to non-augmented

Batch Size	San Francisco XS				Tokyo XS	
	Validation		Test		Test	
32	61.6	75.8	17.5	31.9	27.0	49.2
64	64.9	77.1	21.2	34.6	31.1	**52.7**
128	66.2	78.1	**22.8**	**36.9**	**32.1**	52.1
256	**66.6**	**78.4**	22.4	36.1	**32.1**	**52.7**

Table 1. Results of the batch size tests, all models are trained over 25 epochs of 250 iterations each.

images are regularly fed to the classifiers. The total loss is then computed as a linear combination of primary and secondary loss and all weights are updated. By doing this we aim at fulfilling the requirements for domain adaptation and by bringing the descriptors of original and augmented images closer together, to make the net focus on capturing invariant features, thereby improving domain generalization. This approach shows great performances at test time.

5. Experiments

In this section we present our numerical experiments and the results we obtain. Before diving in the numbers, we believe some remarks are necessary. Firstly, our computational resources are limited. Most models are composed of light-weight elements and are trained over a moderate amount of iterations and epochs, using Google Colab's GPUs. Each experiment takes from 2 to 4 hours. Secondly, Cosplace model works very well with large datasets but is known to have downwards scalability issues. Both SF/XL and Tokyo 24/7 are very small compared to SF/XS and Tokyo 24/7, so we cannot make comparisons between our results and the original ones in [4]. We therefore not interested in the absolute value of the results, but rather the relative improvement with respect to the appropriate baseline that we define each time. The score metrics are R@N for N $\in \{1, 5\}$, the percentage of queries for which one of the first N prediction is correct. In each table we report both scores in this order for the validation, sf/xs and Tokyo/xs testsets.

We first test the effect of different batch size on a regular Cosplace Network with a simple normalization as the only augmentation. We use a ResNet18, GeM pooling and the Adam optimizer with a learning rate of 10^{-5}. Results are shown in Table 1. The best results are obtained with the value 128, however we end up selecting a batch size of 64 for the rest of the experiments as the increase in R@1 is not large enough to compensate for the increase in the duration of the training phase. This specific model we'll be used as the base on which apply variations (e.g. different backbones). When an element of a network is not discussed then it is assumed to be the same as the base.

Jittering	San Francisco XS				Tokyo XS	
	Validation		Test		Test	
No jitter	64.9	77.1	21.2	34.6	31.1	52.7
Brightness	**66.2**	**78.5**	**22.7**	35.7	**31.4**	**56.2**
Contrast	65.8	78.2	21.9	**36.6**	29.8	51.4
Hue	65.3	77.7	20.3	34.3	29.8	48.6
Saturation	65.2	77.3	20.7	34.6	28.6	47.3
Combined	65.0	77.0	19.6	32.6	29.5	47.3

Table 2. Recalls obtained jittering different illumination parameters. The values used are 0.7 for brightness contrast and saturation, 0.5 for hue. Combined is an augmentation jittering all the four parameters, with the same values as above.

Random Erasing	San Francisco XS				Tokyo XS	
	Validation		Test		Test	
Std. Erasing	**69.2**	**80.2**	19.2	33.0	**30.5**	49.8
Hard Erasing	68.9	79.9	22.0	35.2	29.5	50.2
Soft Erasing	69.1	80.0	**22.9**	**36.3**	**30.5**	**52.7**

Table 3. Results of the Random Erasing tests, all models are trained over 25 epochs of 250 iterations each. Learning rate is set to 10^{-4}

5.1. Data Augmentation

For the illumination jittering we isolated each of the four parameters and compared their results with the baseline without any augmentations of this type. What came out is that only the brightness brought a significant improvement, as showed in Table 2. Probably this is due to the limited number of iterations, as already stated. In particular hue and saturation variations are the worst ones, negatively impacting the baseline performance. It is worth noting that this worsening is slighter on the validation set. The last row of the table is about the combination of the four parameters jittering. The results are the worst ones, probably because the net sees each image too few times to achieve an improvement from this type of augmentation.

Street-view specific Random Erasings show good performances on sf/xs, outperforming both the base (second row in Table 1) and the standard random erasing on sf/xs. However it seems that RE is not the correct augmentation technique for the Tokyo/xs, since all 3 models are beaten by the base without any RE. This may be caused by either a general absence of relevant occlusions or the different nature of such obstructions. From Fig. 6 we get the impression that vehicles are simply less relevant that in sf/xs.

5.2. Backbones and aggregations

We experiment with the three families of networks already introduced in Sec. 4.2, limiting ourself to light-weight implementations. In particular we select 4 models smaller

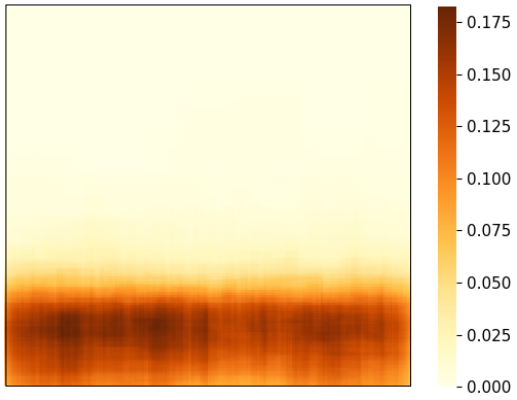

Figure 6. Average occlusion heatmap for Tokyo/xs

Backbones	San Francisco XS				Tokyo XS	
	Validation		Test		Test	
ResNet18	**66.3**	78.3	21.7	35.2	31.1	52.7
EffNetb0	65.8	78.5	21.3	35.2	30.5	47.0
EffNetb1	64.6	77.7	20.1	34.1	30.8	51.1
EffNetb2	66.2	78.7	24.6	39.6	**34.3**	**54.6**
MobVit050	37.9	54.4	8.6	18.4	14.9	31.1
MobVit100	55.9	71.3	15.8	27.8	18.7	36.5
MobVit150	66.2	**79.0**	**26.1**	**42.1**	31.1	47.6

Table 4. Results of the backbone tests, all models are trained over 35 epochs of 250 iterations each.

than ResNet18 (EfficientNets b0 and b1 and MobileViTs[1] 050 and 100) and 2 of comparable size (EfficientNet_B2 and MobileViTV2_150). For CNNs we kept the original idea to train only the last few layers (>90% of all parameters), full results in Table 4. As expected, the first group performance is sub-par. However we can see that small efficientnets are far better than small mobilevits. We believe that the transformer nature of the latter is unfit for such a small number of parameters [10] [11]. Both EfficientNet_b2 and MobileViT_150 significantly outperforms ResNet18 on sf/xs, with the latter reaching the higher scores, but the hybrid model struggles more on Tokyo/xs. We conclude that overall EfficientNet_b2 is the best backbone and that MobileViT_150 should be paired with some more aggressive data augmentation in order to improve his robustness.

For the aggregation layer we conducted tests to evaluate the performances of three different approaches: Adaptive Pooling, Attention Pooling, and the MixVPR Model (with depth fixed to 8). The baseline is the base model introduced earlier and at first we only modify the aggregation. The results in the Table 5 show that adaptive pooling achieved the highest score across the ResNet models. We then test it on other backbones and we find that performances are compa-

[1]We don't mention the version for legibility purposes, but we always mean MobileViTV2

Aggregation	San Francisco XS				Tokyo XS	
	Validation		Test		Test	
GeM	64.9	77.1	21.2	34.6	31.1	52.7
Attention	64.7	78.1	21.6	33.3	34.6	49.5
Adaptive	65.2	77.6	23.7	36.5	32.1	47.9
MixVPR	58.2	68.6	14.4	22.1	25.1	45.1
Effb2 Adp.	64.4	77.7	23.5	37.8	33.0	**53.0**
MViT150 Adp.	**68.1**	**80.5**	**26.5**	**39.9**	**34.9**	50.2

Table 5. Results of the aggregation techniques tests, all models are trained over 25 epochs of 250 iterations each. The last 2 rows are additional test made to confirm the good performances of Adaptive Pooling. Where not specified the backbone is ResNet.

rable or even better than those in Table 4 even though they were trained over 10 more epochs. In addition to the aforementioned results, we observed that while attention pooling performed worse in the sf/xs dataset, it exhibited a good generalization for the Tokyo/xs testset. We also argue that, since MixVPR truncates the CNN before its final layer, it may benefit from a larger backbone.

5.3. Optimizers and schedulers

We did not observe significant differences between the original optimizer Adam and AdamW. As shown in Table 6, SGD instead achieved very poor results. Adam and AdamW consider by default a momentum term, unlike SGD for which we experimented the versions with and without momentum. Concerning the scheduler we tested both ReduceLROnPlateau and CosineAnnealingLR in combination with the Adam optimizer and tried some different configurations of the decay parameters. In some cases, expecially with high learning rates or with an high number of epochs, we observed that the recall on the validation set were plateauing. However we used the first scheduler with respect to the loss, which continued slightly decreasing. Therefore the scheduler was not often triggered and we needed to set a very high threshold to be able to see some decaying. Anyway this did not lead to a significant improvement. The CosineAnnealing method was tested with different values of T_{max}. A faster decay leads to a worsening on the performance, while a slower one does not deviate much from the baseline. The majority of our tests have been conducted with an initial learning rate of 10^{-5}, but we also experimented the values 10^{-4} and 10^{-3}. This last one obtained very poor results. With 10^{-4} instead we observed a modest worsening with ResNet18, maybe due to overfitting issues since the recall on the validation has risen, but a significant enhancement with MobileViT.

5.4. Self-supervised task and Domain Adaptation

When training with a double loss it is important to find the right weight for the secondary loss. After some prelim-

Optimizers, schedulers , lr	San Francisco XS				Tokyo XS	
	Validation		Test		Test	
Adam 10^{-5}	64.9	77.1	**21.2**	**34.6**	31.1	**52.7**
AdamW 10^{-5}	64.5	76.9	21.0	34.4	28.9	50.2
SGD 10^{-5}	44.4	61.2	10.3	21.0	17.5	35.2
R.OnPla. 10^{-5}	61.3	75.1	18.2	31.5	27.9	47.0
CosAnn. 10^{-5}	60.4	74.4	17.4	30.5	26.7	47.9
R.OnPla. 10^{-4}	**67.2**	**79.0**	18.4	32.8	30.5	51.1
CosAnn. 10^{-4}	66.6	78.7	18.3	32.0	**31.7**	49.8

Table 6. Results obtained with different optimizers, with and without the use of a scheduler and with different initial learning rates. For the cosine annealing the table refers to $T_{max} = 100$. SGD is with 0.9 of momentum parameter. All the schedulers are tested with Adam.

	San Francisco XS				Tokyo XS	
	Validation		Test		Test	
Model 1	66.4	78.4	19.8	35.0	30.2	52.4
Model 2	74.4	84.6	35.3	50.3	41.9	58.4
Model 3	**77.8**	**87.7**	**39.7**	**53.5**	**45.7**	**66.3**

Table 7. Results of the tests on the self-supervised models. All are trained models are trained over 25 epochs of 500 iterations each.

inary tests we decide to set it to 0.3, as higher values affect performances and lower ones are not significant enough. As for data augmentation, we focus on illumination and Random Erasing. Learning rate is fixed to $5 * 10^{-5}$. We are not able to test this model with every combination of promising elements, but following our previous findings we choose 3 models to actively test:

Model 1) The first is a simple model similar to the baseline, to which we add heavy illumination augmentation and hard random erasing;

Model 2) The second one utilizes the EfficientNet_b2 and attention pooling, both of which seem to perform very well on Tokyo/xs. Moreover we use the AdamW optimizer, reduce the illumination changes for brightness and hue and enhance the random erasing.

Model 3) The last one is the same as the second but it uses a MobileViT_150. Since we see evidence of this backbone struggling with generalization, we want to see if matching it with this framework can alleviate this problem.

We find that the best method is clearly number 3 (Table 7). We conclude that this framework, combined with attention pooling, can help MobileViTs increase their generalization performances. It is also worth noting that also EfficientNet_2 tested great, confirming what we found in the backbones experiments

Ensambles	San Francisco XS				Tokyo XS	
	Validation		Test		Test	
Baseline	66.3	78.3	21.7	35.2	31.1	52.7
Supervised	**77.9**	**87.8**	36.1	50.5	39.4	60.3
Self Supervised	77.8	87.7	**39.7**	**53.5**	**45.7**	**66.3**

Table 8. Results of the best models found compared to the baseline

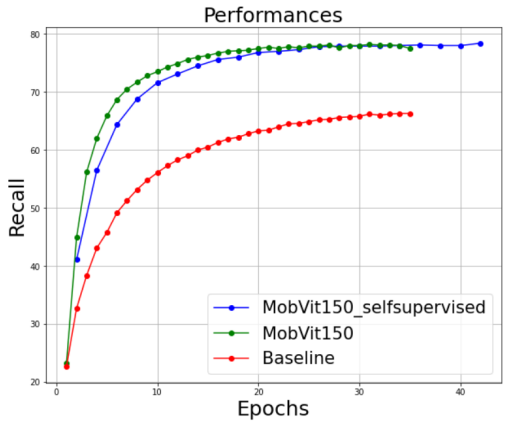

Figure 7. Trend of the recall on the validation set over ≈ 40 epochs for the baseline, the best model without self-supervised loss and the best model with self-supervised loss.

5.5. Best Models

Finally we summarize our results by comparing the two best models we can find with the baseline. The first one is the third one from the previous subsection. The second one is a combination of the most performing elements we found: MobileViT_150, adaptive pooling, lr of 10^{-4}, AdamW and CosAnnealing. Results are compared in Table 8. Both advanced models clearly outperforms the baseline in both testsets. In Fig. 7 we can see that the self supervised model converges more slowly than the other, however it does not plateau so early and has better generalization power.

6. Conclusions

In our study we gained valuable insights into different families of backbones and aggregation methods, into the significance of illumination and occlusion perturbations and into the selection of both optimizers and schedulers. This new knowledge helped us to refine the model architecture and incorporate self-supervised learning. we achieved significant advancements in visual geolocalization techniques, particularly demonstrated by the substantial performance boost on the Tokyo dataset. These findings contribute to the field of visual geolocalization and provide practical insights for future research in this domain.

References

[1] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. Mixvpr: Feature mixing for visual place recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2998–3007, 2023. 5

[2] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016. 2

[3] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2022. 5

[4] Gabriele Berton, Carlo Masone, and Barbara Caputo. Rethinking visual geo-localization for large-scale applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4878–4888, June 2022. 1, 2, 4, 6

[5] Silvia Bucci, Antonio D'Innocente, Yujun Liao, Fabio M. Carlucci, Barbara Caputo, and Tatiana Tommasi. Self-supervised learning across domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5516–5528, 2022. 5

[6] Jimmy Lei Ba Diederik P. Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2015. 5

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 4

[8] Frank Hutter Ilya Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2019. 5

[9] Hyo Jin Kim, Enrique Dunn, and Jan-Michael Frahm. Learned contextual feature reweighting for image geo-localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[10] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021. 4, 7

[11] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680*, 2022. 4, 7

[12] Murat, Fatih Saran, Nurdan Nar, and Saran. Perlin random erasing for data augmentation. *29th IEEE Signal Processing and Communications Applications*, 2021. 3

[13] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018. 1, 4

[14] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 4

[15] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. In *CVPR*, 2015. 1, 2

[16] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. *Advances in Neural Information Processing Systems*, 34:30392–30400, 2021. 4

[17] Zhen, Zhipeng Yang, Wenshan Wang, Xiuying Xu, Zhichao He, Zhijian Wang, and Yin. Region-aware random erasing. *2019 IEEE 19th International Conference on Communication Technology*, 2019. 3

[18] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):13001–13008, Apr. 2017. 3