Improving CosPlace: limitations and results

Gioè Tiziano
Politecnico di Torino
s300886@studenti.polito.it

Lo Truglio Samuele
Politecnico di Torino
s295285@studenti.polito.it

Palmeri Mario Gabriele
Politecnico di Torino
s302740@studenti.polito.it

Abstract

*Visual geo-localization consists in calculating the geographical position of a given image. This is usually done by taking the **query** image, match it against a **database** of images that represent the known world, and create a list of database images sorted by similarity with our input (query) image. This is done through a kNN search. In this paper, we try some extension of an already existing method: **CosPlace**. This method partitions the geographical area of interest in oriented cells, which serve as the classes the network uses to train itself. We dive into methods to better handle perspective changes and domain shifts, while also improving the general performances, finding CosPlace's limitations and advantages.*

1. Introduction

Visual geo-localization is the task recognizing the geographical whereabouts of an image. This is often approached as an **Image Retrieval** problem: that is, given a *query* image we want to geo-localize, we match it against a *database* of labeled image, and determine the location using the images with the most similarity with the *query*. One of the latest approach is **CosPlace [3]**. CosPlace casts the network training as an **Image Classification** problem: it partitions the geographical area of interest in oriented cells and iteratively considers subsets of these cells as classes to train the network. At inference time, the features extracted by CosPlace are used for retrieval as usual. As many other Visual Geolocalization approaches, CosPlace presents some limitations:

Sensitivity to perspective changes. VG neural networks struggle in handling perspective changes in images (e.g. if the query image represents a monument to the right, whereas in the database that same monument is to the left). Considering the "raw" nature of the query images, it is crucial to find a way to deal with these scenarios.

Handling domain shifts. In real world applications, test images could belong to very different domains. For instance, one could consider images taken during the day or night, with or without rain. Therefore it's very important to focus on resilience to domain shifts, which is something many VG approaches struggle in dealing with.

Contributions. In this paper we address these limitations with the following contributions:

- A new dataset, called **Tokyo Night**, where all query images are taken at night and database images are the same of Tokyo XS.
- Use of different **loss functions** to replace the original cosface loss.
- A new **Gradient Reversal Layer** [8] that helps the model to be less domain dependent.
- Various data augmentation techniques like **FDA** [25], **ToDayGAN** [1] and query related processing.
- Application of **GeoWarp** [4], a way to re-rank images found by the network.
- Use of **different networks** as backbone and model ensemble to find out general improvements.

2. Related works

Visual geo-localization. Visual geo-localization is commonly approached as an image retrieval problem, with a retrieved image deemed correct if it is within a predefined range (usually 25 meters) from the query's ground truth position. Typically VG methods perform the retrieval using learned embeddings that are produced by a feature extraction backbone equipped with a head that implements some form of aggregation or pooling. These architectures are trained via contrastive learning, typically using a **triplet loss** [2] and leveraging the geo-tags of the database images as a form of weak supervision to mine negative examples [2]. Despite the variations, all these methods suffer from poor train-time scalability caused by expensive mining techniques. The problem of scalability also arises

at test time, and it relates to the **size of the descriptors**, which impacts the required memory and the retrieval time. Recent works like [5] try to solve the scalability problem with data augmentation techniques at inference time and other strategies, whereas **CosPlace** [3] accomplishes the task with a new way of thinking training part without triplet loss and negative examples.

Domain adaptation. Domain Adaptation is used to reduce the gap between the source and target distribution of our data. In the unsupervised approach, we use labelled source data and unlabeled target data. The first main approach to apply Unsupervised Domain Adaptation relies on style transformation: this way, we can convert images from our domain to another one. The second is based on learning features that are not too domain-specific, so that if the domain changes, the features will still give us some useful insights about the data. These two approaches were discussed in works like AdaGeo [6], where both methods were merged in a single process, by using only a few samples from the target domain.

Re-ranking methods. Re-ranking is a process with which predictions can be re-ordered to improve the recalls, at the cost of more computation during inference time. It is usually done with Spatial Verification methods [15], which are used to find local correspondences in pairs of images but there are also solutions involving an auxiliary warping network like GeoWarp [4], which helps in datasets that contain images with a large perspective difference between them. The re-ranking process is divided in two steps: feature matching and consistency check. Feature matching is used to detect correspondences among a pair of images, using brute force search to find local features that are nearest neighbors. Consistency check is used to analyze the consistency of spatial transformations and verify the reliability of the correspondences.

3. Method

Considering the limitations of CosPlace presented above, we now dive into each of them, presenting the contributions for this project.

3.1. Loss functions

CosPlace uses the Large Cosine similarity loss, also known as **CosFace** [22], which is a cosine based loss that adds a margin to the cosine function to better separate the decision boundaries between classes. We replace the CosFace loss with another implementation of the CosFace loss and two other alternative cosine based losses: **SphereFace** [12] and **ArcFace** [7]. As a matter of fact, as shown in [7], these losses are specific cases of the general angular margin

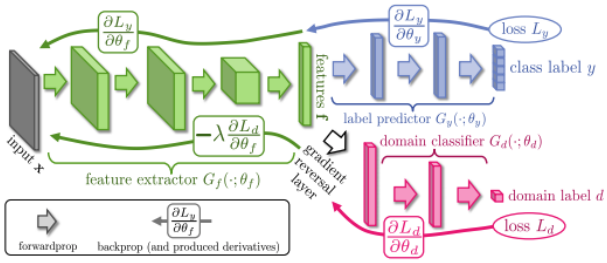

Figure 1. GRL scheme. During the forward pass this layer acts like the identity, whereas during backpropagation we apply the gradient reversal operation.

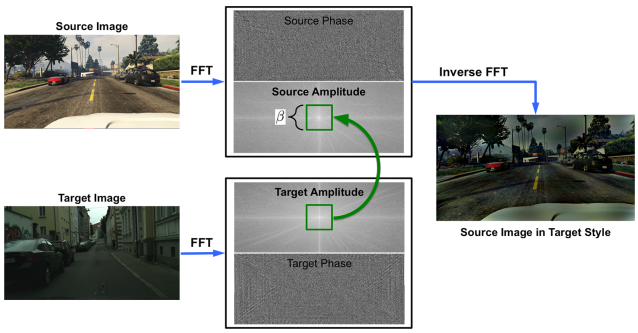

Figure 2. FDA scheme. Note how we apply the target image's fft amplitude to the source one, resulting in an altered version of the source version that recalls the style of the target domain.

penalty-based loss Eq. (1)

$$L = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s(cos(m_1\theta_{y_i}+m_2)-m_3)}}{e^{s(cos(m_1\theta_{y_i}+m_2)-m_3)}+\sum_{j=1,j\neq y_i}^{n} e^{scos\theta_j}} \tag{1}$$

where m_1 is the parameter used by SphereFace, m_2 is the parameter used by ArcFace and m_3 is the parameter used by CosFace.

3.2. Robustness to Domain Shifts

Even though CosPlace generalizes well to unseen domain, there's still room for more improvements.

An initial approach is to add a **Gradient Reversal Layer (GRL) [8].** GRL is a layer used for transfer learning: it helps the model to ignore certain information during training by reversing the sign of the gradient during backpropagation, thus discarding features that are too domain-specific. To apply it, we simply add it as a layer in between the already defined layer of our network of choice. Fig. 1 shows the layers' flow with GRL.

Another method we can look into is **FDA** [25]. It consists in manipulating the fourier transform of the **source** images, to apply the style of the **target** image to them.

Fig. 2 shows a scheme representing how FDA basically works. Due to very extended testing times, we decide to ap-

ply FDA only as a data augmentation technique to Tokyo-Night. Further optimization attempts can be done considering FDA at train time.

Finally, the last approach we can analyze is **ToDayGAN** [1], a generative adversarial network that allows to transpose day images to night ones, and viceversa. This can also be useful since, in tokyo-night, we only have night queries and day database images. This can allow us to find similarities in images in a better way. We use this model only on the tokyo-night dataset

3.3. Robustness to Perspective Changes

We apply a horizontal flip factor to the train dataset that randomly horizontal flip database's images in order to help the model to work properly in cases where the queries have a strong change in perspective w.r.t. the database.
Also, as a first approach to mitigate this problem, we use **GeoWarp** [4], a method that brings invariance in viewport shifts when extracting dense features.
Re-ranking. Given that GeoWarp can be used for re-ranking in an existing VG pipeline, we use it to re-order the Top K predictions computed with CosPlace. San-Francisco xs images have a substantial perspective change between queries and database images, so we reduce the gap recomputing the ordering of the K predictions.
Fig. 3 shows how warping the image can help in creating a bigger visual **overlap**, allowing to better find the most similar and nearest database images.

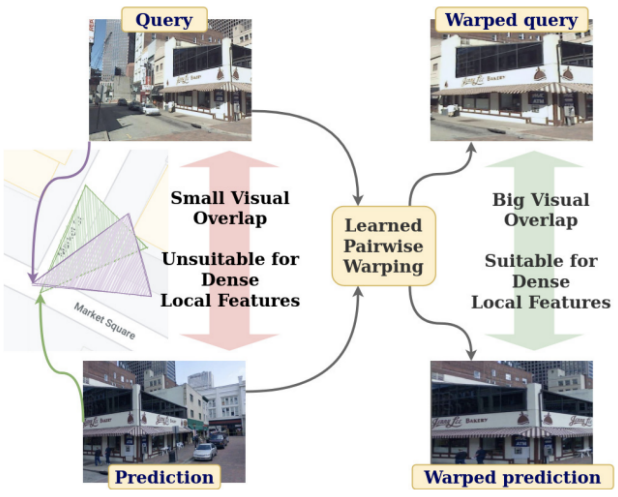

Figure 3. Main functionality of GeoWarp. Warping the images helps in finding similarity and so on re-raking the predictions.

3.4. General Improvements

We investigate in other techniques in order to improve time speed, resources occupation and recalls.

Different backbones. CosPlace is backbone-agnostic, which means that it can works with different networks as backbone. By default the backbone is ResNet-18 [9] preatrined on ImageNet [17]. We change backbones in two ways:

- **ResNet-18 pretrained on different datasets w.r.t. ImageNet.** The same backbone can perform differently if the starting weights are different
- **Different backbones.** We think that newer networks can perform better and in a more efficient way.

Query images manipulation. We notice that currently the only way to process query images is made by the single query extraction, that is a good way to manage the images from the queries at inference time, but it's not scalable, as explained by [5], so we will apply different data augmentation techniques in order to manage more than one image at inference time.
Model ensemble. We try to ensemble different models with two different techniques: the classic model ensemble in which we do the mean of the descriptors obtained by the ensembled networks with an expensive time increase and the ModelSoup [24] greedy soup approach that add weights of models to the soup only if they improve recalls.

4. Experiments

We now apply the techniques mentioned above, and analyze the results. We start from the default CosPlace parameters and apply our chosen approaches. We use Google Colab Free, with its limitations: https://research.google.com/colaboratory/faq.html#resource-limits, so the setup for our tests is composed by 2-core Xeon @2.2GHz, 13GB RAM and NVIDIA M4000. All our training processes consists in three epoches with 10'000 iterations. For our tests we use the provided datasets: SF-XS (training, validation, test), Tokyo XS (test) and Tokyo Night (test). SF-XS and Tokyo-XS are subversions of the respective SF-XL and Tokyo 24/7. At first we try to train with all default settings, as shown in first row of Tab. 1.

4.1. Tokyo Night Dataset

We create the Tokyo Night dataset starting from Tokyo XS; although the database is exactly the same, the queries are formed by the Tokyo XS queries taken at night. The result is a dataset with just 105 query images and default model performance are poor as we can see in Tab. 1. We justify these results considering that database images are always taken at day both in training and testing phase.

4.2. Loss functions

We train the model with ResNet-18 as backbone and the previously cited loss functions adopting the hyperparameter

Loss Function	s	m	SF-XS		Tokyo XS		Tokyo Night	
			R@1	R@5	R@1	R@5	R@1	R@5
CosPlace CosFace	30.0	0.40	52.2	66.3	69.5	84.8	50.5	72.4
CosFace	64.0	0.35	48.7	60.6	71.1	81.9	60.0	67.6
CosFace	30.0	0.40	47.8	63.9	68.9	83.5	50.5	69.5
ArcFace	64.0	0.50	47.6	61.6	69.2	82.9	52.4	66.7
SphereFace	30.0	1.50	50.4	64.4	70.8	85.7	54.3	72.4

Table 1. Test results on different loss functions. First row is the default trained model.

Backbone	GRL	SF-XS				Tokyo XS				Tokyo Night			
		R@1	R@5	R@10	R@20	R@1	R@5	R@10	R@20	R@1	R@5	R@10	R@20
Mobilenet V3 Large	/	50.4	64.8	69.1	74.6	68.6	85.4	89.5	93.0	55.2	75.2	82.9	88.6
Mobilenet V3 Large	0.3	49.8	63.4	68.6	74.0	72.1	85.4	88.9	93.0	64.8	78.1	82.9	88.6
Mobilenet V3 Large	0.5	48.6	62.6	69.1	74	69.5	83.5	87.9	92.4	57.1	73.3	81.9	88.6
Mobilenet V3 Large	0.7	49.5	64.3	70.1	74.8	70.5	82.2	88.6	91.4	54.3	72.4	81.9	86.7
Efficientnet V2 Small	/	64.7	75.1	79.6	82.6	83.5	92.1	94.3	96.8	74.3	85.7	90.5	94.3
Efficientnet V2 Small	0.3	64.6	75.4	80	83	81.3	92.1	93.7	96.8	70.5	84.8	87.6	92.4
Resnet 18	/	52.2	66.3	71.8	76.3	69.5	84.8	89.2	93	50.5	72.4	79.0	85.7
Resnet 18	0.3	44.5	58.5	63.2	68.4	57.8	72.7	81	84.4	33.3	52.4	64.8	71.4

Table 2. Test results on different backbones, with and without GRL, on the SF-XS Dataset. The GRL column values, if specified, indicate the value of the hyperparameter α

suggested by the authors' implementations. We try another implementation of the CosFace with two different couple of hyperparameters, through this process we find that provided hyperparameters can be not the optimal ones for this specific task, because performances are typically the same for all losses, as shown in Tab. 1.

4.3. Gradient Reversal Layer

In GRL, the parameter alpha describes the impact of the gradient reversal operation on the neural network. We evaluate different values for this hyperparameter using a **Random Search** approach. To evaluate the potential improvements without too long training times, we use **Mobilenet V3 Large [10]** as our backbone, and compare the results with those of a model built on the same backbone. We then use the best hyperparameter for other backbones, and compare all the found values. We notice a general boost of 1.7% in Mobilenet V3 Large, applying GRL with $\alpha = 0.3$. Considering only the R@1 and R@5, this boost increases to 3.3%. Since the other two α values produced negative boosts, we only consider 0.3 as a viable value for GRL. Applying GRL with $\alpha = 0.3$ to **Efficientnet V2 Small [20]** and **Resnet18 [9]**, we notice negative boosts. Tab. 2 contains all these results.

4.4. TodayGAN

We apply TodayGAN to tokyo-night, to soften the domain shift problem. Our experiments are conducted on this dataset because it is the most sensible to domain shifts (we have night queries and day database images). The following operations are performed on this dataset:

1. Convert the night query images to day ones
2. Convert the day query images to night ones

We use ToDayGAN self-trained on the Oxford Robotcars dataset and get the results available in Tab. 3. The results were not as good as expected: in fact, they are very low.

Further optimization attempts can be done considering a self-trained version of ToDayGAN.

4.5. FDA

We consider the Fourier Data Augmentation at test time, to transpose images' qualities and informations from one domain to another. In particular, we perform the following operations:

1. Apply the style of sf-xs to the entirety of tokyo-night
2. Apply the style of the tokyo-night database to the tokyo-night queries
3. Apply the style of the tokyo-night queries to the tokyo-night database
4. Apply the style of the tokyo-night queries to the tokyo-night database, adding the new images to the previous one, instead of overwriting them.

We consider Efficientnet V2 Small with GRL and $\alpha = 0.3$ and we apply FDA on the *tokyo-night* dataset. Tab. 4 shows our result: we can see that this method does not handle domain shift well in our case.

Backbone	GRL	Augmentation	Tokyo Night			
			R@1	R@5	R@10	R@20
Efficientnet V2 Small	0.3	None	70.5	84.8	87.6	92.4
Efficientnet V2 Small	0.3	Night query images to day	14.3	30.5	46.7	60.0
Efficientnet V2 Small	0.3	Day database images to night	27.6	50.5	56.2	64.8

Table 3. Test results for the Efficientnet V2 Small + GRL(0.3) + pretrained ToDayGAN

Backbone	GRL	Augmentation	Tokyo Night			
			R@1	R@5	R@10	R@20
Efficientnet V2 Small	0.3	None	70.5	84.8	87.6	92.4
Efficientnet V2 Small	0.3	FDA from sf-xs to tokyo-night	65.7	80	81	88.6
Efficientnet V2 Small	0.3	FDA from the tokyo-night database to tokyo-night queries	68.6	81	83.8	88.6
Efficientnet V2 Small	0.3	FDA from tokyo-night queries to the tokyo-night database	70.5	83.8	88.6	90.5
Efficientnet V2 Small	0.3	FDA from tokyo-night queries to the tokyo-night database (without overwrite)	70.5	83.8	88.6	90.5

Table 4. Test results for the Efficientnet V2 Small + GRL(0.3) + FDA

4.6. GeoWarp

GeoWarp's trained homography regression models. Firstly, we use the default GeoWarp's models. After computing the predictions for the Top 20 nearest images with CosPlace, we re-rank them using GeoWarp. In this process we use different backbones and different pooling methods; we also use a parameter *n* which describes how many predictions are going to be re-ranked.
We use Alexnet [11], Resnet50 [9] and Vgg16 [18] with gem [16] and netvlad [2] pooling. As for n we use values 5, 10 and 15. However, due to computational limit, for Vgg16 we use only n = 5. Even if the inference time is increased by some minutes, the results are very interesting, especially for Resnet50. Tab. 14 contains them.
Division between database and gallery images. As others VG methods, GeoWarp's training is based on a dataset of geotagged images divided in gallery and queries. Since CosPlace does not need this division and given that sf-xs is not divided using this approach, we split it when creating the dataset for GeoWarp. In particular we use:

- the 80% of the images in every class for the **gallery**
- the 20% of the images in every class for the **queries**

Self-trained homography regression models. We now train with GeoWarp using the pre-trained baselines with sf-xs to obtain our homography regression model. Due to computational limit, we train only with Alexnet (both with gem and netvlad) and Resnet50 (only with gem); we use also a limited number of epochs. After obtaining the models, we then proceed to repeat the previous testing process. This leads to overall worst results than the previous try. They are reported in Tab. 10.

4.6.1 Geowarp and GRL

We now combine Geowarp and GRL, and see the results. To do so, we consider Efficientnet V2 Small with GRL and $\alpha = 0.3$ and we apply GeoWarp with all the settings combinations discussed in this subsection. The most interesting result are gathered in Tab. 5. We notice relevant boosts when reranking with the following configurations:

1. Resnet50/Netvlad/5 (+1.3% on R@1,R@5 and +2.4% overall)
2. VGG16/gem/5 (+1.6% on R@1,R@5 and +2.8% overall)

The full results are available in Tab. 18

4.7. Different Backbones

We train the model with different pretrained ResNet-18:

- **Places365** [26], a dataset with different scenes
- **Google Landmarks Dataset V2** [23], a dataset with different human made and natural landmarks

In both models we don't obtain interesting results, in fact the original model performs better in all of the testing datasets as shown in Tab. 6.
Then we train the model with different backbone networks in order to analyze the effect of the backbone on the results, the networks that we decide to use are:

- **EfficientNet** B0, B1, B2 [19], networks developed to obtain a good trade-off between performance and computational complexity.
- **EfficentNet V2** S [20], new version of the previous ones, with an higher complexity.

Re-rank aux network	Pooling	n	Time (min)	SF-XS			Tokyo XS			Tokyo Night		
				R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
no re-rank	-	-	-	64.6	75.4	80	81.6	92.7	94	68.6	84.8	85.7
resnet50	netvlad	5	5:58	66.1	75.1	78.1	83.8	92.7	94.0	75.2	84.8	85.7
resnet50	netvlad	10	11:48	64.3	74.6	78.1	84.4	93.0	94.0	75.2	83.8	85.7
vgg16	gem	5	16:04	66.3	75.1	78.1	85.7	92.7	94.0	75.2	84.8	85.7
vgg16	netvlad	5	16:04	62.8	75.1	78.1	85.1	92.7	94.0	76.2	84.8	85.7

Table 5. Test results (summarized) **with GRL** and GeoWarp's trained homography regression models.

Backbone	Parameters (M)	Training Time (h)	SF-XS		Tokyo XS		Tokyo Night	
			R@1	R@5	R@1	R@5	R@1	R@5
ResNet18	11.7	**2.49**	52.5	66.2	69.5	83.5	51.4	73.3
EfficientNet V2 S	21.4	4.26	**64.7**	75.1	83.5	92.1	74.3	85.7
EfficientNet V2 S + GeoWarp (resnet50+netvlad+10)	21.4	4.26	**64.8**	**76.6**	**85.1**	**92.7**	**80.0**	**87.6**
Swin Tiny	28.3	5.76	53.0	67.2	66.3	85.4	48.6	71.4

Table 6. Test results of models with different backbones. Backbones paremeters are in millions(M), Training Time is in hours(h). Complete results at Tab. 15 and Tab. 14

- **MobileNet V3** Small, Large [10], next generation of MobileNets tuned to mobile phone CPUs.
- **ConvNext** Tiny [14], a 2022 ResNet that can compete with Transformers [21].
- **SwinTransformer** Tiny [13], a vision transformer model.

Backbone freeze and truncation. Each backbone has a different number of layers and of output channels, so starting from the CosPlace implementation, we understand which part of the backbone we have to freeze and which part we have to truncate. We freeze different number of the last layers w.r.t. the different backbones.
We set AMP ON, ImageNet for pretrained weights and the use of Cosface loss for all the models, unless where it's written differently.

4.7.1 Evaluation

We report backbones' parameters, training time[1] and recalls for all the testing datasets in Tab. 6
Training time. MobileNet V3 Small is the **fastest one**, but it's the one with worst recalls, then there are MobileNet V3 Large and ResNet-18 (with AMP ON) that have the same times, with little difference in the recalls. All other networks are slower, but we have to consider also that networks like EfficientNet V2 S, ConvNext Tiny and Swin Tiny have a complexity that is comparable to ResNet-50 that we can't test because of our computational resources.
Recalls. Our EfficientNet V2 S model **outperforms** all of the other networks. This backbone provides a general boost in all the recalls expecially for R@1 and R@5. It's interesting to see how much the performances improve in Tokyo-Night that is the most difficult one because query images are taken in the night and the database images are taken in the day. Anyway the number of Tokyo Night queries is **very low** so we should consider this result with caution and focus instead on Tokyo XS. Talking about SF-XS where the dataset is more populated, we can see the same trend of the other networks, keeping also a good improvement with respect to the default backbone. **GeoWarp.** We apply GeoWarp also to MobileNet model where the results are similar to the Resnet-18 ones. When we apply GeoWarp to EfficientNet model we obtain better results only when using the configuration with ResNet-50, so we can confirm that this setup typically provides good recalls, despite the additional time.

4.8. Test Time Query Processing

Query images have different sizes w.r.t. the database images. In this step we apply different data augmentations techniques [5] in order to observe the behaviour of our models. Thanks to queries resize, we can stack more than one image together and **increase the batch size**. We report our tests on models with ResNet-18 and EfficientNet V2 S as backbones in Tab. 7. In ResNet-18 tests we can see that methods with post processing generally have better results, with different percentage of improvement depending on the test dataset and the specific method. In EfficientNet V2-S tests results are tipically worse, but for both the models it's interesting to see that **Central Crop** is a light-weight

[1] As training time, we consider the time needed to train the model for the given epoch and number of iterations plus the time needed to validate the model. In this way we consider both training and inference time.

Backbone	Query Method	Batch Size	SF-XS			Tokyo XS			Tokyo Night		
			E. Time (s)	R@1	R@5	E. Time (s)	R@1	R@5	E. Time (s)	R@1	R@5
ResNet-18	-	1	19	52.2	66.3	6	69.5	84.8	2	50.5	72.4
ResNet-18	central crop	16	11	47.0	61.8	5	69.8	84.1	2	51.4	71.4
	nearest crop	16	47	51.0	65.7	17	70.8	86.7	7	54.3	73.3
EfficientNet V2 S	-	1	44	64.7	75.1	14	83.5	92.1	5	74.3	85.7
EfficientNet V2 S	central crop	8	17	57.6	70.2	7	80.0	88.9	3	68.6	82.9
	nearest crop	8	87	64.0	75.8	28	82.2	90.8	10	73.3	83.8
	maj voting	8	97	63.9	75.8	27	82.5	90.8	10	73.3	83.8

Table 7. Test results with different query processing method. Extraction time is in seconds (s). Complete results at Tab. 17

Backbone	Criteria	Ingredients	SF-XS		Tokyo XS		Tokyo Night	
			R@1	R@5	R@1	R@5	R@1	R@5
ResNet-18	sf-xs val R@1	cosplace loss	47.0	61.8	69.8	84.1	51.4	71.4
	sf-xs test R@1	cosplace loss + hflip	48.2	62.1	68.9	86.3	47.6	73.3
	tokyo night R@5	cosplace loss + hflip; sphereface	46.9	60.6	68.9	85.7	52.4	72.4
EfficientNet V2 S	tokyo xs R@1	cosplace loss	57.6	70.2	80.0	88.9	68.6	82.9
	tokyo xs R@5	sphereface loss; cosplace loss	57.5	68.4	79.4	90.2	67.6	84.8

Table 8. Test results of model soups. Criteria defines the starting ingredient of the soup (model with best R@k in dataset d). Soup ingredients are the effective number of model in the soup. Models in ingredients column are separeted by ';'. Complete results at Tab. 16

method that allows us to parallelize query images with a good trade off between performance and inference time (expecially in EfficientNet V2-S model).

4.9. Horizontal Flip

We train the model applying this data augmentation technique using a random factor of 0.5. As shown in the second row of Tab. 8 results are similar to the default model; we have lower recalls in Tokyo Datasets, so we don't use this model in other configurations than Model Soup context.

4.10. Model Ensemble

We now work on ensembling models to obtain better performance. We ensemble models with different backbones, doing the mean of their descriptors, we can't test all our models together, so we try only a greedy test with our ResNet, MobileNet and EfficientNet models and we don't find an ensembled model that increases performance, so we skip this method to focus on others.
Talking about Model Soup method, we decide to adopt the ***greedy soup*** variant in which model's weight are added only if they improve recalls provided by previous "ingredients". The limitation of Model Soup approach is that **we can't ensemble model with different backbones**, but only ensemble models with the same layers configuration. The pro of using Model Soup is that we can obtain mixed descriptors without increase time of execution. For this procedure, we consider **six different models** with ResNet-18 as backbone and two different version of the EfficientNetV2 S one. In order to let the algorithm works, we evaluate in different ways our soup, considering recalls at any of our test datasets, including the sf-xs validation dataset as reported in Tab. 8.
Generally for all the criteria no other model is added to the initial soup and in the cases in which this happens, the new model improves only slightly in the criteria's dataset and even lower if we consider only R@1 and R@5.

5. Conclusions

In this work, we study different methods to improve CosPlace. We look into various fields to increase the performances, from domain adaptation to perspective changes handling. Through our approaches we find that both GeoWarp and the applying of different (and newer) backbones bring a **significant performance boost**, whereas our domain adaptation techniques, except for GRL which in some cases had decent results, aren't so helpful in our situation, typically it looks like it's difficult to improve the model as it is without increase training or testing time.

In case of real application, we think that parallelization at inference time is **crucial**, so a better study of query processing techniques could help not also to reduce the total inference time, but also the robustness to domain shift.

However, in future studies we could further investigate on ToDayGAN (by using a self-trained version) and FDA (by utilizing it at train time). Our work and our results are **limited by our setup restrictions and the low number of epoches in training**. We should test our approaches with proper hardware and different setups to study if our conclusions are globally valid.

References

[1] Asha Anoosheh, Torsten Sattler, Radu Timofte, Marc Pollefeys, and Luc Van Gool. Night-to-day image translation for retrieval-based localization, 2018. 1, 3

[2] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition, 2015. 1, 5

[3] 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] Gabriele Berton, Carlo Masone, Valerio Paolicelli, and Barbara Caputo. Viewpoint invariant dense matching for visual geolocalization. In *ICCV*, pages 12169–12178, October 2021. 1, 2, 3

[5] Gabriele Berton, Riccardo Mereu, Gabriele Trivigno, Carlo Masone, Gabriela Csurka, Torsten Sattler, and Barbara Caputo. Deep visual geo-localization benchmark. In *CVPR*, June 2022. 2, 3, 6

[6] Gabriele Moreno Berton, Valerio Paolicelli, Carlo Masone, and Barbara Caputo. Adaptive-attentive geolocalization from few queries: a hybrid approach. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, jan 2021. 2

[7] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Cotsia, and Stefanos P Zafeiriou. ArcFace: Additive angular margin loss for deep face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. 2

[8] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation, 2014. 1, 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 3, 4, 5

[10] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019. 4, 6

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. 5

[12] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017. 2

[13] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. 6

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

[15] Carlo Masone and Barbara Caputo. A survey on deep visual place recognition. *IEEE Access*, 9:19516–19547, 2021. 2

[16] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation, 2017. 5

[17] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2014. 3

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. 5

[19] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019. 5

[20] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. 2021. 4, 5

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 6

[22] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5265–5274, 2018. 2

[23] Tobias Weyand, Andre Araujo, Bingyi Cao, and Jack Sim. Google landmarks dataset v2 – a large-scale benchmark for instance-level recognition and retrieval, 2020. 5

[24] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 17–23 Jul 2022. 3

[25] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation, 2020. 1, 2

[26] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 5

6. Appendix

In this section, we report the full results of our methods, which we display a subset of in the previous sections to facilitate reading.

Re-rank aux network	Pooling	n	SF-XS			Tokyo XS			Tokyo Night		
			R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
no re-rank	-	-	52.2	66.3	71.8	69.5	84.8	89.2	50.5	72.4	79.0
alexnet	gem	5	55.7	66.3	71.8	77.5	84.8	89.2	61.0	72.4	79.0
alexnet	gem	10	56.6	69.1	71.8	76.5	87.3	89.2	57.1	75.2	81.9
alexnet	gem	15	56.2	69.7	73.3	74.9	87.3	91.1	53.3	75.2	81.9
alexnet	netvlad	5	53.6	66.3	71.8	75.2	84.8	89.2	61.9	72.4	79.0
alexnet	netvlad	10	53.4	67.5	71.8	74.9	86.3	89.2	60.0	76.2	79.0
alexnet	netvlad	15	52.1	68.1	72.6	75.9	85.7	91.4	60.0	74.3	82.9
resnet50	gem	5	56.0	66.3	71.8	76.2	84.8	89.2	62.9	72.4	79.0
resnet50	gem	10	55.7	68.6	71.8	76.8	86.7	89.2	62.9	76.2	79.0
resnet50	gem	15	54.7	69.4	73.5	75.9	87.0	91.4	61.9	77.1	83.8
resnet50	netvlad	5	57.7	66.3	71.8	76.8	84.8	89.2	64.8	72.4	79.0
resnet50	netvlad	10	59.0	68.8	71.8	75.9	88.3	89.2	65.7	77.1	79.0
resnet50	netvlad	15	58.6	68.8	73.2	77.1	88.9	91.4	65.7	79.0	82.9
vgg16	gem	5	57.7	66.3	71.8	79.4	84.8	89.2	66.7	72.4	79.0
vgg16	netvlad	5	54.9	66.3	71.8	78.4	84.8	89.2	67.6	72.4	79.0

Table 9. Test results with GeoWarp's trained homography regression models.

Re-rank aux network	Pooling	n	SF-XS			Tokyo XS			Tokyo Night		
			R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
no re-rank	-	-	52.2	66.3	71.8	69.5	84.8	89.2	50.5	72.4	79.0
alexnet	gem	5	54.5	66.3	71.8	74.6	84.8	89.2	56.2	72.4	79.0
alexnet	gem	10	54.5	68.5	71.8	72.7	85.4	89.2	53.3	71.4	79.0
alexnet	gem	15	53.5	68.2	73	70.8	86.0	90.5	50.5	71.4	81.0
alexnet	netvlad	5	53	66.3	71.8	70.8	84.8	89.2	51.4	72.4	79.0
alexnet	netvlad	10	52.2	67.7	71.8	70.2	84.8	89.2	50.5	71.4	79.0
alexnet	netvlad	15	50.6	67.2	72.4	69.5	85.1	89.5	49.5	72.4	80.0
resnet50	gem	5	52.8	66.3	71.8	73.3	84.8	89.2	57.1	72.4	79.0
resnet50	gem	10	51.5	67.7	71.8	72.1	87.0	89.2	55.2	77.1	79.0
resnet50	gem	15	49.6	66.2	72.3	70.2	86.7	90.5	54.2	76.2	81.9

Table 10. Test results with homography regression models trained on sf-xs. Overall worst result than the GeoWarp's default models.

Re-rank aux network	Pooling	n	SF-XS			Tokyo XS			Tokyo Night		
			R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
no rerank	-	-	52.4	65.5	70.3	69.8	86.3	90.5	57.1	77.1	85.7
alexnet	gem	5	54.9	65.5	70.3	74.0	86.3	90.5	57.1	77.1	85.7
alexnet	gem	10	55.6	67.3	70.3	75.6	87.6	90.5	61.9	80.0	85.7
alexnet	gem	15	55.8	68.2	72.1	74.9	89.5	91.1	58.1	81.0	84.8
alexnet	netvlad	5	54.7	65.5	70.3	72.7	86.3	90.5	60.0	77.1	85.7
alexnet	netvlad	10	53.7	66.0	70.3	74.3	87.6	90.5	61.9	81.9	85.7
alexnet	netvlad	15	52.5	66.3	71.5	72.4	88.3	90.8	58.1	81.0	85.7
resnet50	gem	5	56.0	65.5	70.3	75.2	86.3	90.5	62.9	77.1	85.7
resnet50	gem	10	56.0	67.9	70.3	75.2	87.0	90.5	63.8	81.0	85.7
resnet50	gem	15	54.7	68.4	72.5	75.9	88.3	91.1	63.8	80.0	86.7
resnet50	netvlad	5	57.2	65.5	70.3	76.5	86.3	90.5	65.7	77.1	85.7
resnet50	netvlad	10	59.0	68.1	70.3	79.0	86.7	90.5	68.6	79.0	85.7
resnet50	netvlad	15	59.0	69.4	72.6	79.7	89.5	92.1	68.6	81.9	86.7
vgg16	gem	5	58.4	65.5	70.3	80.0	86.3	90.5	65.7	77.1	85.7
vgg16	netvlad	5	55.7	65.5	70.3	77.1	86.3	90.5	65.7	77.1	85.7

Table 11. Test results with GeoWarp's trained homography regression models using MobilenetV3 Large instead of Resnet18

Re-rank aux network	Pooling	n	SF-XS			Tokyo XS			Tokyo Night		
			R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
no rerank	-	-	52.4	65.5	70.3	69.8	86.3	90.5	57.1	77.1	85.7
alexnet	gem	5	53.5	65.6	70.3	73.7	86.3	90.5	58.1	77.1	85.7
alexnet	gem	10	52.7	66.3	70.3	74.3	86.7	90.5	61.0	78.1	85.7
alexnet	gem	15	51.8	67.5	71.7	72.4	88.9	90.8	56.2	80.0	83.8
alexnet	netvlad	5	52.9	65.5	70.3	72.4	86.3	90.5	56.2	77.1	85.7
alexnet	netvlad	10	52.4	66.3	70.3	72.1	87.0	90.5	56.2	80.0	85.7
alexnet	netvlad	15	51.3	65.6	71.4	71.4	86.0	91.4	54.3	76.2	85.7

Table 12. Test results with our self-trained homography regression models using MobilenetV3 Large instead of Resnet18

Re-rank aux network	Pooling	n	SF-XS			Tokyo XS			Tokyo Night		
			R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
no rerank	-	-	64.7	75.1	79.6	83.5	92.1	94.3	74.3	85.7	90.5
alexnet	gem	5	66.0	75.1	79.6	85.4	92.1	94.3	77.1	85.7	90.5
alexnet	gem	10	63.6	76.1	79.6	81.9	92.7	94.3	71.4	85.7	90.5
alexnet	gem	15	62.4	75.8	79.8	81.0	93.0	94.6	69.5	86.7	89.5
alexnet	netvlad	5	63.8	75.1	79.6	81.6	92.1	94.3	71.4	85.7	90.5
alexnet	netvlad	10	60.8	75.1	79.6	78.4	91.4	94.3	65.7	85.7	90.5
alexnet	netvlad	15	59.4	73.9	79.0	77.1	91.7	94.6	63.8	85.7	90.5
resnet50	gem	5	65.3	75.1	79.6	85.1	92.1	94.3	79.0	85.7	90.5
resnet50	gem	10	63.1	76.2	79.6	79.4	92.4	94.3	73.3	87.6	90.5
resnet50	gem	15	60.5	75.5	79.6	78.4	92.4	95.2	71.4	86.7	92.4
resnet50	netvlad	5	66.4	75.1	79.6	84.4	92.1	94.3	75.2	85.7	90.5
resnet50	netvlad	10	64.8	76.6	79.6	85.1	92.7	94.3	80.0	87.6	90.5
resnet50	netvlad	15	65.4	76.0	79.9	83.2	93.3	94.9	79.0	88.6	91.4
vgg16	gem	5	66.5	75.1	79.6	84.8	92.1	94.3	74.3	85.7	90.5
vgg16	netvlad	5	64.2	75.1	79.6	83.5	92.1	94.3	74.3	85.7	90.5

Table 13. Test results with GeoWarp's trained homography regression models using EfficientnetV2S instead of Resnet18

Re-rank aux network	Pooling	n	SF-XS			Tokyo XS			Tokyo Night		
			R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
no rerank	-	-	64.7	75.1	79.6	83.5	92.1	94.3	74.3	85.7	90.5
alexnet	gem	5	62.8	75.1	79.6	84.8	92.1	94.3	74.3	85.7	90.5
alexnet	gem	10	60.0	75.3	79.6	79.0	91.4	94.3	66.7	84.8	90.5
alexnet	gem	15	57.4	74.8	79.5	77.5	92.4	94.6	62.9	86.7	90.5
alexnet	netvlad	5	62.4	75.1	79.6	80.3	92.1	94.3	71.4	85.7	90.5
alexnet	netvlad	10	58.6	75.3	79.6	76.5	90.2	94.3	66.7	84.8	90.5
alexnet	netvlad	15	57.1	72.7	78.6	74.3	90.5	93.7	61.0	83.8	89.5

Table 14. Test results with our self-trained homography regression models using EfficientNetV2 instead of Resnet18

Backbone	Parameters (M)	Training Time (h)	SF-XS			Tokyo XS			Tokyo Night		
			R@1	R@5	R10	R@1	R@5	R10	R@1	R@5	R10
ResNet18 AMP OFF	11.7	4.0	52.2	66.3	71.8	69.5	84.8	89.2	50.5	72.4	79.0
ResNet18 AMP OFF + Place365	11.7	4.0	50.3	64.3	70.5	62.9	80.0	86.7	41.0	59.0	70.5
ResNet18 AMP OFF + GLDV 2	11.7	4.0	48.8	62.2	67.9	64.4	83.5	91.4	43.8	64.8	80.0
ResNet18	11.7	2.49	52.5	66.2	70.4	69.5	83.5	88.9	51.4	73.3	80.0
EfficientNet B0	5.3	2.70	43.8	59.6	64.9	63.8	80.6	87.0	41.0	64.8	75.2
EfficientNet B1	7.8	3.00	39.5	54.5	59.2	59.0	78.7	83.2	40.0	63.8	70.5
EfficientNet B2	9.1	3.00	43.3	55.8	60.2	66.7	83.5	87.3	54.3	71.4	78.1
EfficientNet V2 S	21.4	4.26	64.7	75.1	79.6	83.5	92.1	94.3	74.3	85.7	90.5
EfficientNet V2 S + SphereFace	21.4	4.26	63.1	73.5	77.3	82.9	92.1	94.3	73.3	86.7	90.5
MobileNet V3 S	2.5	2.40	33.2	45.3	49.6	56.8	74.3	79.0	37.1	60.0	64.8
MobileNet V3 L	5.5	2.49	50.4	64.8	69.1	68.6	85.4	89.5	55.2	75.2	82.9
MobileNet V3 L (4 training epoches)	5.5	3.32	52.4	65.5	70.3	69.8	86.3	90.5	57.1	77.1	85.7
ConvNext Tiny	28.6	4.20	51.7	65.4	70.9	65.4	84.1	87.9	41.0	66.7	72.4
Swin Tiny	28.3	5.76	53.0	67.2	72.8	66.3	85.4	88.3	48.6	71.4	75.2

Table 15. Test results of models with different backbones. Backbones paremeters are in millions(M), Training Time is in hours(h).

Backbone	Criteria	Soup Ings. / Pot Ings.	Ingredients	SF-XS		Tokyo XS		Tokyo Night	
				R@1	R@5	R@1	R@5	R@1	R@5
ResNet-18	sf-xs val R@1	1/6	cosface (cosplace)	47.0	61.8	69.8	84.1	51.4	71.4
	sf-xs val R@5	2/6	cosplace loss; cosplace loss + hflip	48.0	63.0	68.9	83.2	49.5	69.5
	sf-xs test R@1	1/6	cosplace loss + hflip	48.2	62.1	68.9	86.3	47.6	73.3
	sf-xs test R@5	1/6	cosplace loss + hflip	48.2	62.1	68.9	86.3	47.6	73.3
	tokyo xs R@1	1/6	cosplace loss	47.0	61.8	69.8	84.1	51.4	71.4
	tokyo xs R@5	1/6	cosplace loss + hflip	48.2	62.1	68.9	86.3	47.6	73.3
	tokyo night R@1	1/6	arcface loss	43.4	55.6	69.2	82.5	54.3	69.5
	tokyo night R@5	2/6	cosplace loss + hflip; sphereface	46.9	60.6	68.9	85.7	52.4	72.4
EfficientNet V2 S	sf-xs val R@1	1/2	cosplace loss	57.6	70.2	80.0	88.9	68.6	82.9
	sf-xs val R@5	1/2	cosplace loss	57.6	70.2	80.0	88.9	68.6	82.9
	sf-xs test R@1	1/2	cosplace loss	57.6	70.2	80.0	88.9	68.6	82.9
	sf-xs test R@5	1/2	cosplace loss	57.6	70.2	80.0	88.9	68.6	82.9
	tokyo xs R@1	1/2	cosplace loss	57.6	70.2	80.0	88.9	68.6	82.9
	tokyo xs R@5	2/2	sphereface loss; cosplace loss	57.5	68.4	79.4	90.2	67.6	84.8
	tokyo night R@1	1/2	cosplace loss	57.6	70.2	80.0	88.9	68.6	82.9
	tokyo night R@5	1/2	sphereface loss	55.9	67.9	78.4	89.5	68.6	83.8

Table 16. Test results of model soups. Criteria defines the starting ingredient of the soup (model with best R@k in dataset d). Soup ingredients are the effective number of model in the soup. Models in ingredients column are separeted by ';'.

Backbone	Query Method	Batch Size	SF-XS			Tokyo XS			Tokyo Night		
			E. Time (s)	R@1	R@5	E. Time (s)	R@1	R@5	E. Time (s)	R@1	R@5
ResNet-18	-	1	19	52.2	66.3	6	69.5	84.8	2	50.5	72.4
	single query		19	52.5	65.9	6	69.5	84.8	2	50.5	72.4
ResNet-18	central crop	16	11	47.0	61.8	5	69.8	84.1	2	51.4	71.4
	five crops		47	50.4	64.6	18	67.6	83.5	7	47.6	71.4
	nearest crop		47	51.0	65.7	17	70.8	86.7	7	54.3	73.3
	maj voting		48	50.8	65.8	16	70.8	86.7	6	56.2	73.3
ResNet-18	central crop	8	11	47.0	61.8	4	69.8	84.1	2	51.4	71.4
	five crops		43	50.4	64.6	14	67.6	83.5	6	47.6	71.4
	nearest crop		42	51.0	65.7	14	70.8	86.7	6	54.3	73.3
	maj voting		43	50.8	65.8	14	70.8	86.7	6	56.2	73.3
EfficientNet V2 S	-	1	44	64.7	75.1	14	83.5	92.1	5	74.3	85.7
	single query		44	65.1	75.4	14	83.5	92.1	5	74.3	85.7
EfficientNet V2 S	central crop	8	17	57.6	70.2	7	80.0	88.9	3	68.6	82.9
	five crops		88	63.0	74.6	28	79.0	90.5	10	68.6	81.9
	nearest crop		87	64.0	75.8	28	82.2	90.8	10	73.3	83.8
	maj voting		97	63.9	75.8	27	82.5	90.8	10	73.3	83.8

Table 17. Test results with different query processing method. Extraction time is in seconds (s).

Re-rank aux network	Pooling	n	Time (min)	SF-XS			Tokyo XS			Tokyo Night		
				R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
no re-rank	-	-	-	64.6	75.4	80	81.6	92.7	94	68.6	84.8	85.7
alexnet	gem	5	1:31	64.1	75.1	78.1	83.8	92.7	94.0	69.5	84.8	85.7
alexnet	gem	10	3:09	62.1	75.6	78.1	80.3	92.7	94.0	61.9	82.9	85.7
alexnet	gem	15	4:21	62.1	74.8	78.6	79.4	92.4	93.3	61.9	81.9	83.8
alexnet	netvlad	5	1:29	62.3	75.1	78.1	80.6	92.7	94.0	68.6	84.8	85.7
alexnet	netvlad	10	3:08	58.7	73	78.1	79.7	92.4	94.0	65.7	84.8	85.7
alexnet	netvlad	15	4:21	58.7	73	78.1	77.5	91.4	93.3	62.9	81.0	84.8
resnet50	gem	5	5:58	64.0	75.1	78.1	84.1	92.7	94.0	74.3	84.8	85.7
resnet50	gem	10	11:48	60.9	74.7	78.1	81.6	92.7	94.0	70.5	82.9	85.7
resnet50	gem	15	17:36	60.6	74.7	78.6	79.4	91.4	94.3	68.6	81.9	85.7
resnet50	netvlad	5	5:58	66.1	75.1	78.1	83.8	92.7	94.0	75.2	84.8	85.7
resnet50	netvlad	10	11:48	64.3	74.6	78.1	84.4	93.0	94.0	75.2	83.8	85.7
resnet50	netvlad	15	17:37	64.7	74.6	78.9	82.9	92.7	93.7	73.3	82.9	84.8
vgg16	gem	5	16:04	66.3	75.1	78.1	85.7	92.7	94.0	75.2	84.8	85.7
vgg16	netvlad	5	16:04	62.8	75.1	78.1	85.1	92.7	94.0	76.2	84.8	85.7

Table 18. Test results **with GRL** and GeoWarp's trained homography regression models.