Self-Supervised Foundation Model for Template Matching

Anton Hristov *, Dimo Dimov, Maria Nisheva-Pavlova

Posted Date: 7 November 2024

doi: 10.20944/preprints202411.0478.v1

Keywords: self-supervised learning; template matching; foundation model; convolutional neural network; image matching

Article

Self-Supervised Foundation Model for Template Matching

Anton Hristov [1,*], Dimo Dimov [2,*] and Maria Nisheva-Pavlova [1,3,*]

[1] Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", 5 James Bourchier Blvd., Sofia 1164, Bulgaria
[2] Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Acad. G. Bonchev Str., Block 2, Sofia 1113, Bulgaria
[3] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Acad. G. Bonchev Str., Block 8, Sofia 1113, Bulgaria
* Correspondence: anhristov@gmail.com (A.H.); dimo.dimov@iict.bas.bg (D.D.); marian@fmi.uni-sofia.bg (M.N.-P.)

Abstract: Finding a template location in query image is a fundamental problem in many computer vision applications, such as localization of known objects, image registration, image matching and object tracking. Currently available methods fail when not enough training data is available or big variations in the textures, different modalities and weak visual features exist in the images, leading to limited applications on real-world tasks. We introduce Self-Supervised Foundation Model for Template Matching (Self-TM), a novel end-to-end approach for self-supervised learning template matching. The idea behind Self-TM is to learn hierarchical features incorporating localization properties from images without any annotations. As going deeper in the convolutional neural network (CNN) layers, their filters begin to react to more complex structures and their receptive fields increase. This leads to loss of localization information in contrast to the early layers. The hierarchical propagation of the last layers activations back to the first layer results in precise template localization. Due to its zero-shot generalization capabilities on tasks such as image retrieval, dense template matching and sparse image matching our pre-trained model can be classified as a foundation one.

Keywords: self-supervised learning; template matching; foundation model; convolutional neural network; image matching

1. Introduction

In computer vision, visual template detection is a technique used to identify and locate a pattern in a larger image. This usually involves comparing the image of the template on an input image at different positions to find its best match. This technique can be used for detection of familiar objects, image registration and object tracking tasks.

Despite the significant progress in the development of this technique, the solutions developed so far do not emphasize a sufficiently wide range of essential properties important for practice, such as generalization capability, real-time execution and easy retraining that does not require annotated data, etc.

Self-supervised models [1–3] are artificial neural networks designed to extract knowledge from visual data without relying on annotations. The lack of necessity for a large amount of annotated data is the reason for its increased popularity in recent years. As self-supervised models evolve, they continuously improve their generalization capabilities while being more data efficient.

This work presents a novel approach for self-supervised learning of a CNN designed for visual template detection. Our Self-TM models provide a high degree of generalization capabilities across various tasks, which classifies them as so-called "foundation" models dealing with high accuracy on

 doi:10.20944/preprints202411.0478.v1

data of a different modality, i.e., highly different from those used for training. If accuracy increase is needed, these foundation models can be easily and quickly fine-tuned on a small set of real data.

The applied approach in Self-TM can be attributed to "Siamese network-based trackers" [4–13] where typically the same model (encoder) is used to extract both image features of the search object and image features in which it is searched. Then, the features from the last layers are correlated using learnable cross-correlation [12,13], which result is then processed by one or multiple decoders (convolutional neural networks or transformers) [14–19]. Most often, the result obtained by the decoders is one or several regression and classification maps [12,13].

Based on this framework, the present work proposes highly efficient architecture, in addition to a simple yet accurate correlation approach. Combined with the intuitive training method, Self-TM can be easily fine-tuned on any type of images.

The novelty in the proposed Self-TM may be summarized by the following:

- The high degree of generalization eliminates the need to retrain the model with real data;
- If further training (fine-tuning) is still necessary, very few images of the real data are needed until the desired accuracy is reached;
- The specially designed encoder is trained on the task of locating a searched object, what differs from the standard approaches of using a network trained on some other tasks, most often classification or a more general one;
- The integrated correlation operator we use provides real information about the location of the searched object, rather than being a separate layer of the network [12,13] which has to be further trained and the result to be decoded by one or more neural networks. This approach has not been found in the literature so far;
- Hierarchical propagation of activations from the last to the first layer leads to precise localization while precluding the need to use an additional decoder that needs to be trained. The result is an extremely simplified and lightweight architecture, while providing high accuracy. This approach has not been found in the literature so far;
- Two step self-supervised training, involving two types of data augmentations: color augmentation, and color and geometric augmentations;
- Self-TM is rotationally invariant. In this work, the Self-TM models' family is trained over the entire interval from -90 to +90 degrees.

2. Related Work

2.1. Hand-Crafted Methods

Template matching has a long history in the field of computer vision with many hand-crafted methods, mostly involving correlation-based techniques in which the searched template is slid over the target image, computing various similarity scores.

Traditional methods applied directly on pixel intensities, such as SSD (sum of squared differences) [20] and SAD (sum of absolute differences) [21] are very sensitive to noise, deformations and visual transformations of the object or region of interest, making them inapplicable to real tasks. Papageorgiou and Poggio (in 1998) introduced a novel method [22] that uses normalized cross-correlation (NCC), allowing a certain degree of robustness against illumination and contrast variations. Their approach has spawned later advances, for example with zero-mean NCC (ZNCC) [23] in pattern matching, emphasizing the importance of normalization.

Another type of methods, based on keypoints' visual features such as SIFT [24], SURF [25] and ORB [26], which have emerged as an alternative to traditional template matching techniques, have wide popularity, because they can efficiently handle variations in scale, rotation, and perspective. Despite their accuracy fluctuations, they are used in various systems since they have high performance, good generalization capabilities on new data, and do not need high computational resources.

 doi:10.20944/preprints202411.0478.v1

2.2. Learnable Methods

Driven by the idea of increasing the accuracy of template matching, in recent years the focus of development has mainly shifted to the development of two types of learnable methods: sparse and dense.

Sparse methods have gained popularity for their efficiency due to the use of keypoints features, resulting in reduced computational cost and improved robustness. For example, SuperPoint [27] is a keypoints extraction algorithm that introduces a trained CNN to detect keypoints and generate their features. SuperGlue [28] is focused on feature matching detection. It evaluates and aligns each pair of keypoints by constructing an association matrix based on their features and applies Sinkhorn [29] to determine the optimal match between them. LightGlue [30] shows significant progress with its efficient matching determination. It uses a small architecture, combining the ability to early stop inference when matches are already found, and exclude for later processing points that cannot be compared. OmniGlue [31] presents a framework using SuperPoint [27] to keypoints detection that attempts to improve the generalization properties by exploiting features from foundation model [32].

Dense methods [33–36] have demonstrated remarkable improvements in accuracy for similarity search between images, especially with recent advances introduced by the transformers [15,17–19]. Unlike sparse methods, they aim to establish correspondences for each pixel of the processed images, resulting in dense information flow. This approach is particularly useful in applications where understanding the entire image is critical. However, recent methods mainly emphasize accuracy, thereby increasing computational requirements to unacceptable levels, challenging even systems with significant GPU resources.

Self-TM offers a learnable dense approach for localizing a search template, taking advantage of the dense method's high accuracy, while being characterized by a very efficient architecture that does not require high computational resources. Self-TM can also be integrated into a sparse-based framework such as OmniGlue [31] (described in section 4.3), increasing its accuracy and preserving the characteristic properties of this type of methods.

Although learnable methods have proven to perform well on their training data, they are characterized by a limitation in the similarity of the tasks in which they can be applied due to their limited generalization that is characteristic for the foundation models.

2.3. Foundation Models

Foundation models are pre-trained, data-rich, highly generalizable models designed to perform more than one task with sufficiently high accuracy. These models serve as a foundation for a variety of applications, allowing for fine-tuning or adaptation to specific tasks with less data and computational resources.

Key features of these models are: pre-training with large datasets covering a wide range of visual concepts, the ability to fine-tune on a small but specific corpus of data focused on a particular task, the ability to adapt to different tasks or subtasks, high robustness and generalization ensuring robust performance in different domains and tasks.

For example, DINO [32] and DINOv2 [37] are transformers trained by a self-supervised learning approach that allow semantic segmentation without relying on annotated data, making them invaluable for training on large unannotated image sets. DINOv2, as a successor, extracts higher quality features, providing improvements that allow it to solve more complex tasks with greater accuracy. Both methods use a self-distillation approach, where a student model learns from a teacher model of the same architecture. The teacher model is updated based on the student parameters using the exponential moving average technique. DINO and DINOv2 use "contrastive" learning [38], which encourages the model to produce similar embeddings under variations of the same image, by applying different color and geometric augmentations, while discriminating between different images.

SAM [39] and SAM2 [40] with their accuracy and generalization, revolutionized semantic image and video segmentation by requiring minimal annotations compared to traditional methods. In addition to images, the input for these frameworks also includes prompts in the form of point

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 November 2024 doi:10.20944/preprints202411.0478.v1

positions, a binary mask, rectangle coordinates, and even a textual description, allowing them to focus on a specific region to extract desired features and output.

SegGPT [41] is a segmentation model built on results from [42] that allows the model to adapt to different tasks using a minimum of examples. The model is useful for all segmentation tasks such as instance, object, and semantic segmentation. During training, it performs contextual coloring using a random coloring scheme (rather than specific colors) to identify segments by learning their contextual information, resulting in improved generalization.

Self-TM is a foundation model oriented towards learning high-quality hierarchical features incorporating localization properties. Similar to DINO [32] and DINOv2 [37], the Self-TM models' family is trained using a self-supervised learning approach, not relying on annotated data, making them invaluable for training on large unannotated image sets. The localization properties provide high accuracy in tasks such as object detection, image registration, object tracking, sparse and/or dense image matching.

3. Theoretical Formulation of the Proposed Method

In this section, the setup of our proposed **Sel**f-Supervised **F**oundation Model for **T**emplate **M**atching (Self-TM) method, illustrated in Figure 1, is discussed.

Technical details such as the choice of architecture and layers, data, and training steps involved are explained in detail, as well as essential properties of the hierarchical features that are the overall goal of the present work, including properties for image templates localization without any annotations.

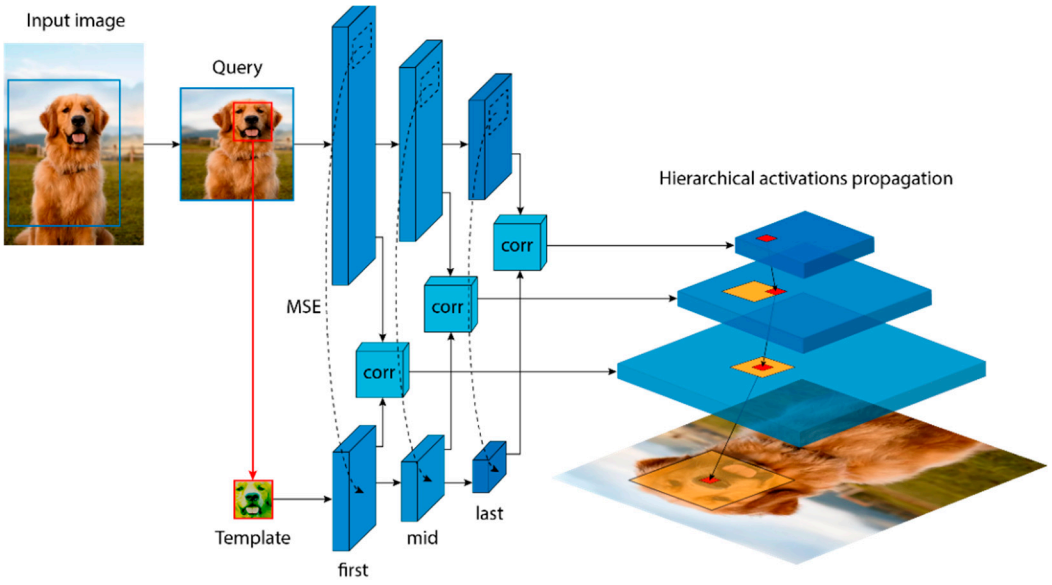


Figure 1. Illustration of Self-TM.

The next section (Section 4) presents the experimental results of Self-TM on data with different modalities (HPatches [43], MegaDepth [44] and ScanNet [45]) and a variety of tasks including template localization, patch verification, patch retrieval and image matching.

Section 5 presents details related to the development of the approach: the iterative learning approach; a comparison of Self-TM against the architecture on which it is based.

3.1. Model

A convolutional neural network is chosen since the proposed method involves hierarchical activations from the final to the first layer of the network where the deeper the layer in which the neuron is located, the larger receptive field it has, i.e. a larger region of the input pixels is covered.

The proposed method is independent of the CNN architecture and can be applied to models with different properties. Our chosen architecture is based on the recently introduced ConvNeXt

 doi:10.20944/preprints202411.0478.v1

architecture [14], which is similar to ResNet [16] but designed to compete in performance with the state-of-the-art vision transformers [18,19]. In addition, this architecture is one of the few effectively used in self-supervised learning.

Our model Self-TM uses the same blocks of ConvNeXt [14], but with modified filter size and stride, see Table 1. Down sampling blocks now use 3 × 3 filters and stride 3, which in terms of receptive field gives a neuron from layer *N* field of view with size 3 by 3 of layer *N*-1.

Table 1. Detailed description of the Self-TM architecture: (**a**) Available model sizes; (**b**) Self-TM architecture. It contains 3 layers, namely for simplicity "first", "middle", "last". X_{first}, X_{mid} and X_{last} denote the number of parameters in the first, middle, and last layers for each model size.

(**a**)

Size	X_{first}	X_{mid}	X_{last}	Num of params
Self-TM Small	128	256	512	13M
Self-TM Base	128	384	1024	40M
Self-TM Large	128	512	2048	130M

(**b**)

Input size	Layer name	Layer components	Output size
3 × 189 × 189	Down sampling	$\begin{bmatrix}\text{Conv2D } 3\times 3, [X_{first}], \text{stride } 3\\ \text{Norm}\end{bmatrix}$	X_{first} × 63 × 63
X_{first} × 63 × 63	ConvNeXt block	$\begin{bmatrix}\text{Conv2D } 7\times 7, [X_{first}], \text{stride } 1, \text{pad } 3\\ \text{Norm, Linear } [X_{first}, 512]\\ \text{GELU, Linear } [512, X_{first}]\end{bmatrix}\times 3$	X_{first} × 63 × 63
X_{first} × 63 × 63	Normalization	Norm	X_{first} × 63 × 63
X_{first} × 63 × 63	Down sampling	$\begin{bmatrix}\text{Norm}\\ \text{Conv2D } 3\times 3, [X_{mid}] \text{ stride } 3\end{bmatrix}$	X_{mid} × 21 × 21
X_{mid} × 21 × 21	ConvNeXt block	$\begin{bmatrix}\text{Conv2D } 7\times 7, [X_{mid}], \text{stride } 1, \text{pad } 3\\ \text{Norm, Linear } [X_{mid}, 1024]\\ \text{GELU, Linear } [1024, X_{mid}]\end{bmatrix}\times 9$	X_{mid} × 21 × 21
X_{mid} × 21 × 21	Normalization	Norm	X_{mid} × 21 × 21
X_{mid} × 21 × 21	Down sampling	$\begin{bmatrix}\text{Norm}\\ \text{Conv2D } 3\times 3, [X_{last}], \text{stride } 3\end{bmatrix}$	X_{last} × 7 × 7
X_{last} × 7 × 7	ConvNeXt block	$\begin{bmatrix}\text{Conv2D } 7\times 7, [X_{last}], \text{stride } 1, \text{pad } 3\\ \text{Norm, Linear } [X_{last}, 2048]\\ \text{GELU, Linear } [2048, X_{last}]\end{bmatrix}\times 3$	X_{last} × 7 × 7
X_{last} × 7 × 7	Normalization	Norm	X_{last} × 7 × 7

The layers used, following ConvNeXt [14], are a standard two-dimensional convolution (Conv2D), a linear layer (Linear), a normalization layer (Norm) [46] and a non-linear GELU activation function [47], see also Section 5.2.

The size of Self-TM Small is significantly compressed, containing only 13 million learnable parameters, which is significantly smaller than standard architectures (Table 2) used for encoders and decoders in template matching or tracking methods. This is given by the network design entirely focused on this type of tasks.

 doi:10.20944/preprints202411.0478.v1

Table 2. Size comparison of Self-TM.

Model	Num of params
Self-TM Small	**13M**
DeiT-S [19], ViT-S [15], Swin-T [48]	22-28M
ConvNeXt-T [14]	29M
Self-TM Base	**40M**
ConvNeXt-S [14], Swin-S [48]	50M
EffNet-B7 [49], RegNetY-16G [50], DeiT-B [19], ViT-B [15], Swin-B [48]	66-88M
ConvNeXt-B [14]	89M
EffNetV2-L [51]	120M
Self-TM Large	**130M**
ConvNeXt-L [14]	198M
ViT-L [15]	304M
ConvNeXt-XL [14]	350M
R-101x3 [52], R-152x4 [52]	388-937M

3.2. Data

ImageNet-1K Train [53] is used for training the Self-TM models because it has established as a standard for training image processing models, as well as because of the high diversity of classes (1000 classes) covering enough different visual concepts for achieving high generalization properties.

3.3. Training

The often-preferred Invariance-based [54] approach has been used to the training, where the idea is to learn similar features for compatible images and discriminative features for incompatible images. These compatible images are produced by applying random image augmentations modifying the input image such as: changing the contrast, blurring, etc. Using this learning approach, it is possible to obtain the so-called representation collapse, where the network's output is constant regardless of the various input data applied.

The training is performed in two successive steps, which differ in the type of augmentations applied to the input images: color augmentations and color and geometric augmentations. This is necessary because geometric augmentations add increased variation to the data that the model cannot initially overcome. To overcome this, Self-TM is initially trained only on data with color augmentations applied. Once this training step is complete, the model continues its training, now adding geometric augmentations to the color augmentations, which are: perspective transform with factor up to 0.5, rotation from -90 to +90 degrees, and rescale with tolerance from -0.3 to +0.3.

Training starts with 15 epochs with color augmentations and continues with 30 epochs with color and geometric augmentations. 4 NVIDIA A5000 24GB graphics cards are used for the training, and an epoch of Self-TM Large takes about 6.5 hours.

The training steps are as follows:

1. An input image I, is taken from an unannotated image database, to which a random crop R is applied, and then resized by S, to 189 × 189 pixels yielding a "query" image, $S(R(I)) = Q$, see Figure 1;
2. Another random crop R, is performed on Q, and then random image augmentations (color and/or geometric) A, modifying the input image to obtain a "template" image, $A(R(Q)) = T$, see Figure 1. This step can produce one or many different templates. In the Self-TM training, only two "template" numbers are used;
3. The position gt_p (ground truth position), of the resulting "template" on the "query" image is stored (the coordinates of the red rectangle's center on "query", see Figure 1);

 doi:10.20944/preprints202411.0478.v1

4. "Query" and the two templates are fed as input to the Self-TM network, $f_\theta: (Q,T) \rightarrow (y,y')$, and the resulting feature maps from all layers are stored, $y = f_\theta(Q)$, $y' = f_\theta(T)$, $(y,y') = \{(y_n,y'_n)|n = 1,\dots,N\}$, where N is the number of the network's layers. In the current architecture the number of layers is 3, denoted above by: first, mid and last;
5. A correlation operator $CORR(y,y')$, is applied to every two corresponding feature maps sequentially starting from the deepest layer. Then the position of the maximum value (predicted position), $pred\_p_n = softmax\big(CORR(y_n,y'_n)\big)$, is found on its output (see "hierarchical activations propagation" in Figure 1);
6. A mean squared error is calculated on every two corresponding feature maps, $MSE_n = MSE(y_n,y'_n) = \frac{1}{|D|}\sum_D (R(y)_n - y'_n)^2$, $D = def(y_n) \cap def(y'_n)$, where D is the region of summation, i.e. the intersection of the two definition domains of the "template" feature map y'_n and the "query" feature map $R(y)_n$, $(y,y') = \{(y_n,y'_n)|n = 1,\dots,N\}$. Multiplication is scalar, i.e. elementwise in D;
7. Parameter optimization (gradient descent): by minimizing the errors obtained from $MSE_N,\dots,MSE_1$, and the offset of the positions of $pred\_p_N,\dots,pred\_p_1$, relative to the real positions of the template $gt\_p_N,\dots,gt\_p_1$, which is possible due to compatibility of positions (in pixels).

3.4. Augmentation

Because geometric augmentations add increased variation to the data that the model cannot initially overcome, training is performed in two steps: training begins on data with applied color augmentations, and then it continues on data with both color and geometric augmentations.

3.4.1. Color augmentations

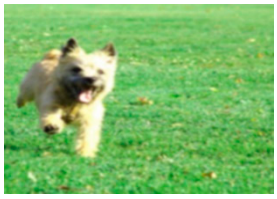
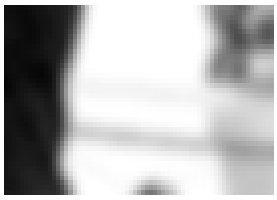
The applied color augmentations are based on the approach in [55], where:

- To obtain the "query" image are applied sequentially:
  - Random crop: scale: from 0.1 to 0.9
  - Rescale: 189 × 189 pixels
  - Normalization: mean = [0.485, 0.456, 0.406], std = [0.228, 0.224, 0.225], computed on ImageNet
- To obtain the two "templates" on the "query" are applied sequentially:
  - Random color jitter with independent probability* of 80%: brightness = 0.4, contrast = 0.4, saturation = 0.2, hue = 0.1
  - Random grayscale with independent probability* of 20%
  - For „template" 1:
    - Random Gaussian blur: radius from 0.1 to 0.2
  - For „template" 2:
    - Random Gaussian blur with independent probability* of 10%: radius from 0.1 to 2.0
    - Random invert of all pixel values above a given threshold (Solarization) with independent probability* of 20%: threshold = 128
  - Normalization: same as in "query"
  - Random crop: scale from 0.14 to 0.85, ratio from 0.2 to 5.0

* Applying random operation with independent probability following the law of uniform distribution

 doi:10.20944/preprints202411.0478.v1

Table 3. Visualization of random color augmentations on randomly cropped images from ImageNet-1K.

Image crop	Color augmentation
	
	
	
	
	
	

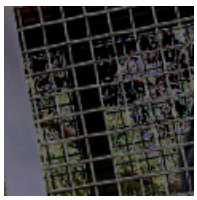
3.4.2. Geometric Augmentations

Geometric augmentations (aligned by the center of the resulting cropping square) are applied on the "template" images only:

- Random square crop: scale from 0.14 to 0.45
- A randomly selected geometric augmentation is applied with a 50% probability:
  - Random perspective transformation: distortion scale = 0.5
  - Random rotation: degrees from -90 to +90
  - Random rescale: factor from -0.7 to 1.3

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 November 2024 doi:10.20944/preprints202411.0478.v1

Table 4. Visualization of random color and random geometric augmentations on random images from ImageNet-1K. For better visualization, the images are cropped in a 1:1 height-to-width ratio.

Image crop	Color and geometric augmentation
	
	
	
	
	
	

3.5. Hierarchical Activations Propagation

One of the main novelty in the proposed framework is the hierarchical propagation of activations from the last to the first layer $pred\_p_N, \ldots, pred\_p_1$ (currently $pred\_p_{last}$, $pred\_p_{mid}$, $pred\_p_{first}$). This results in precise localization although not training an additional decoder is required ("hierarchical activations propagation" in Figure 1).

The localization of the "template" T, in a selected image Q is performed by finding the maximum value (the highest value activation position visualized by a red square in Figure 1 and Figure 2) in the correlation operator's output produced between any two corresponding layers starting from the deepest to the first, i.e. $pred\_p_{N,\ldots,1} = softmax\left(CORR\left(y_{N,\ldots,1}, y'_{N,\ldots 1}\right)\right)$.

We start by finding the position of the maximum value ($pred\_p_N$) in the last layer N, which is propagated to the previous layer N-1 as the position $pred\_p_{N-1}$. This new position ($pred\_p_{N-1}$) is again maximum value but here selected from a limited region denoted as RF (receptive field), of the

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 November 2024 doi:10.20944/preprints202411.0478.v1

maximum value of N, $pred\_p_{N-1} = softmax\left(RF_{pred\_p_N}(CORR(y_{N-1}, y'_{N-1}))\right)$, see Figure 2. In this way, each subsequent layer refines the assumed position of T.

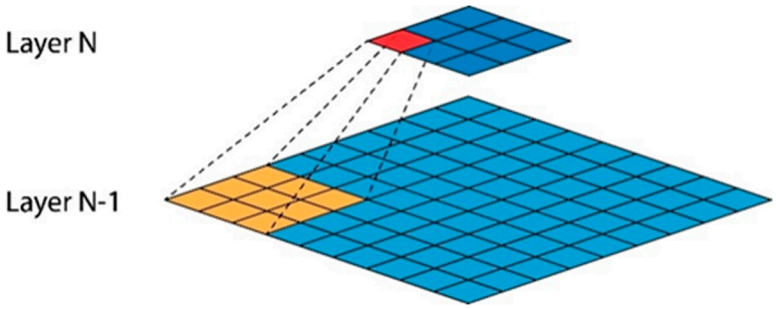


Figure 2. Illustration of a receptive field, $RF_{pred\_p_N}$, in layer $N-1$ (in orange) of a detected maximum value, $pred\_p_N$, in layer N (in red).

To detect the position of the maximum value, a softmax function [56] is used. It is a differentiable argmax function allowing the gradient to flow freely from the end to the beginning of the network during training:

$$softmax(z)_i = \frac{e^{-\beta z_i}}{\sum_{j=1}^{K} e^{-\beta z_i}} \text{ for } i = 1, \dots, K, \text{and } z = (z_1, \dots, z_K) \in \mathbb{R}^K. \quad (1)$$

3.6. Correlation

Another novelty in the proposed framework is the integration of a static correlation operator in the network, the result of which provides information about the location of the searched object, instead of being a separate layer [12,13] that needs to be further trained and then its result decoded by one or more neural networks.

The correlation operator used here is normalized cross-correlation representing a dense measure of a pixel-wise similarity metric:

$$CORR(x, y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x', y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2} \sqrt{\sum_{x',y'} I(x+x', y+y')^2}}, \quad (2)$$

where $(x', y') \in def(Q) \cap def(T)$, i.e. the summation is (only) over the intersection of the domains, of the input I and the template T positioned at the point (x, y) of I.

The result of the correlation operator is a grayscale image, where the position of the template is the pixel with the highest intensity. In this case, the operator is not applied to pixel values, but to the activation values of the layers (feature maps).

3.7. Loss

Two terms MSE and $CORR$ are used for computing the loss, $\mathcal{L}$ while training the network, f_θ:

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{n=1}^{N} \sqrt{MSE(y_n, y'_n)}, \; y_n \in \mathbb{R}, y'_n \in \mathbb{R} \quad (3)$$

$$\mathcal{L}_{CORR} = \frac{1}{N} \sum_{n=1}^{N} \sqrt{(pred\_p_n - gt\_p_n)^2}, \; pred\_p_n \in Z^2, \; gt\_p_n \in \mathbb{R}^2 \quad (4)$$

$$\mathcal{L} = \frac{\mathcal{L}_{MSE} + \mathcal{L}_{CORR}}{2}, \quad (5)$$

where $\mathcal{L}_{MSE}$ is the average of the mean squared error (MSE) results for each layer, and $\mathcal{L}_{CORR}$ is the average of the Euclidean distances between the detected maximum values position $pred\_p$, and the real template positions, gt_p, for each layer.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 November 2024 doi:10.20944/preprints202411.0478.v1

3.8. Hyperparameters/Optimization

An AdamW optimizer [57] with a $learning\_rate = 0.001$ and a regularization technique $weight\_decay = 10^{-6}$ is used in the loss calculation:

$$\mathcal{L} = \mathcal{L} + \underline{weight\_decay} * \underline{L2\ norm\ of\ the\ weights}. \quad (6)$$

3.9. Prediction

A correlation operator $CORR(y, y')$ is applied on every two corresponding feature maps obtained from all layers of the network for an input image $y = f_\theta(I_Q)$ and a selected template $y' = f_\theta(I_T)$. Starting with $CORR$ result of the deepest layer by using "hierarchical activations propagation" the maximum activation value of the first layer is reached. The position of this activation is the coordinates of the template I_T in the input image I_Q. But, since the receptive field in the down sampling blocks for layer *N* is 3 by 3, this resulting position is applicable to the input image I_Q downsized 3 times.

4. Experiments

This section presents experimental results and a demonstration of our proposed Self-TM method on data having the same and different modality as the one used in the training (ImageNet-1K Test [53], HPatches [43], MegaDepth [44] and ScanNet [45]) as well as a variety of tasks:

- ImageNet-1K Test [53], showing the accuracy of template localization on data having the same modality as the training data;
- HPatches [43], evaluating the properties of feature maps as local descriptors for finding matches between different image patches in corresponding images;
- MegaDepth [44], evaluating image matching accuracy on outdoor scenes having different modality from the training data;
- ScanNet [45], evaluating image matching accuracy on indoor scenes having different modality from the training data.

4.1. ImageNet-1K Test

ImageNet-1K test set [53] was used to compare the trained Self-TM model family (Small, Base and Large) based on template localization accuracy.

Since the training is carried out in two successive steps, which differ in the type of applied augmentations of the input images: color augmentations; color and geometric augmentations, this test is performed separately for each of them.

The result is presented in Table 5, where for all Self-TM models (small, base and large) the displacement in pixels of the three layers $(D\_L_{last}, D\_L_{mid}, D\_L_{first})$ of $pred\_p_N, \dots, pred\_p_1$ is calculated, relative to the real positions of the template $gt\_p_N, \dots, gt\_p_1$.

Table 5. ImageNet-1K Test results of Self-TM models trained on ImageNet-1K Train by color or color and geometric augmentations. The calculated displacements (in pixels) D_L_{last}, D_L_{mid}, D_L_{first} are calculated from the last, middle and first layers of the network, respectively.

Model size	Applied augmentation	D_L_{first} pixels	D_L_{mid} pixels	D_L_{last} pixels
Self-TM Small	color	0.579	0.176	0.156
Self-TM Base	color	0.577	0.173	0.156
Self-TM Large	color	0.572	0.171	0.153
Self-TM Small	color and geometric	2.214	0.767	0.409
Self-TM Base	color and geometric	1.752	0.602	0.338
Self-TM Large	color and geometric	1.331	0.452	0.273

The test steps are similar to training steps and are as follows:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 November 2024 doi:10.20944/preprints202411.0478.v1

1. An input image is taken from an unannotated dataset (ImageNet-1K Test), from which a "template" is obtained. On each two corresponding feature maps positions of the maximum values of the correlation operator's result are then calculated (see training steps 1 to 5 in Section 3.3).
2. For each two corresponding feature maps (i.e., for each layer), the position displacement measured in pixels ($D\_L_{last}, D\_L_{mid}, D\_L_{first}$) of $pred\_p_N, \dots, pred\_p_1$ is computed, relative to the actual template positions $gt\_p_N, \dots, gt\_p_1$, using Euclidean distance:

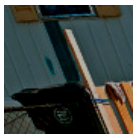
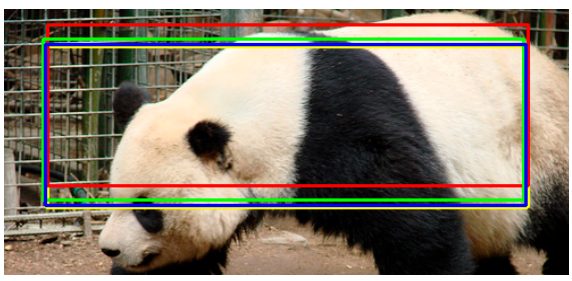
$$D\_L_n = |gt\_p_n - pred\_p_n| = \frac{1}{M}\sum_{i=1}^{M}\sqrt{(gt\_p_n(i) - pred\_p_n(i))^2}, \quad (7)$$

where M is the number of images in the dataset, $gt\_p_n(i)$ and $pred\_p_n(i)$ are the two-dimensional vectors of each image $i = 1, \dots, M$, and n is the currently selected layer of the network $\{D\_L_n | n = 1, \dots, N\}$. In this case, $N = 3$ and for clarity instead of $n = 1, 2, 3$ we are using $first, mid, last$ indexation. D_L_N or D_L_{last} is the average Euclidean distance for the deepest layer.

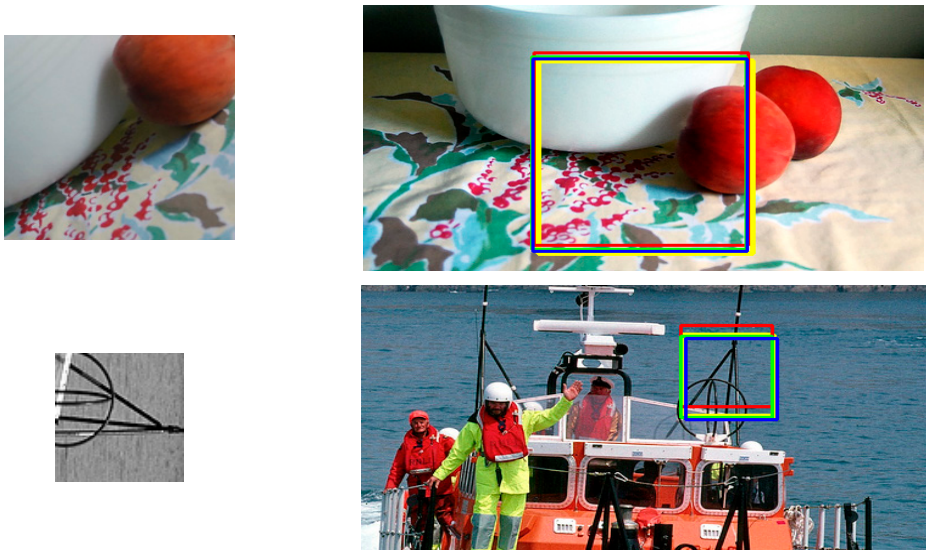
Table 5 clearly shows that the models trained and tested using only color augmentations despite their size difference manage to achieve the same localization accuracy, respectively an average displacement of 0.15 pixels for the deepest layer, followed by an average displacement of 0.17 pixels for the middle layer and an average displacement of 0.57 pixels for the first layer. This shows that even the smallest Self-TM model (13M) has sufficient parameters capable of reaching successful training, competing with the largest Self-TM (130M). However, this is not the case when geometric augmentations are added to the color augmentations, where we observe almost twice the displacement ($D\_L_{first} = 2.214, D\_L_{mid} = 0.767, D\_L_{last} = 0.409$), compared to Self-TM Large (130M) ($D\_L_{first} = 1.331, D\_L_{mid} = 0.452, D\_L_{last} = 0.273$). The reason is the increased variation in the training data by the applied geometric augmentations, which require bigger models and more training time.

Table 6 shows the performance of the Self-TM Large model on random images from the ImageNet-1K Test set. A randomly selected region on which random color and geometric augmentations were applied to obtain a template. From the examples, it can be clearly seen that the selected model successfully manages to cope with the applied complex color and geometry augmentations.

Table 6. Visualization of template localization in random images from ImageNet-1K Test. Random color and geometric augmentations are applied to all the templates. The color legend of the rectangles is as follows: yellow: template position, red: calculated position obtained from the deepest layer, green: calculated from the middle layer, blue: calculated from the first layer.

Template	Result
	
	

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 November 2024 doi:10.20944/preprints202411.0478.v1



4.2. HPatches

HPatches [43] is a dataset for finding matches between different patches, which is used to evaluate the generalization capabilities of Self-TM models.

This dataset has an out-of-domain modality than the one used for training, containing scenes (116 groups of scenes with 6 images in each) with different illumination and geometric augmentations. Patch matches are detected by feature points located by HoG, Hessian and Harris detectors between the first and the other images. The standard metric mAP (mean average precision) was used for evaluation on three different tasks: patch verification, image matching and patch retrieval.

Patch Verification evaluates how well positive pairs are separated from negative pairs. There are two subtasks: SameSeq containing pairs of patches from the same set of scenes and DiffSeq containing pairs of patches from different sets of scenes. The SameSeq is considered more challenging because the textures in different parts of the image are often similar.

Image Matching evaluates how accurately images from the same scene are detected. Here, the overall mAP results are much lower than in Patch Verification, since the ratio of positive to negative examples is significantly lower here and all the negatives come from the same group of scenes. Another interesting observation is that results obtained from images with photometric changes (illum) are significantly lower than those with viewpoint changes (viewpt). This is because the "illum" images include extreme changes in illumination.

Patch Retrieval evaluates how accurately patches correspondences are detected in a large set of patches. Here, a single patch is compared to a large collection containing mainly distractors obtained randomly from the remaining image sets.

In the Self-TM experiments on HPatches, the feature maps from the last layer of the network are used to obtain the patch descriptors. The input data is resized to 32x32 yielding descriptors of 1x512 (Self-TM Small), 1x1024 (Self-TM Base) and 1x2048 (Self-TM Large).

Benchmark of Self-TM with state-of-the-art methods on all three HPatches tasks [43] is shown in Figure 3.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 November 2024 doi:10.20944/preprints202411.0478.v1

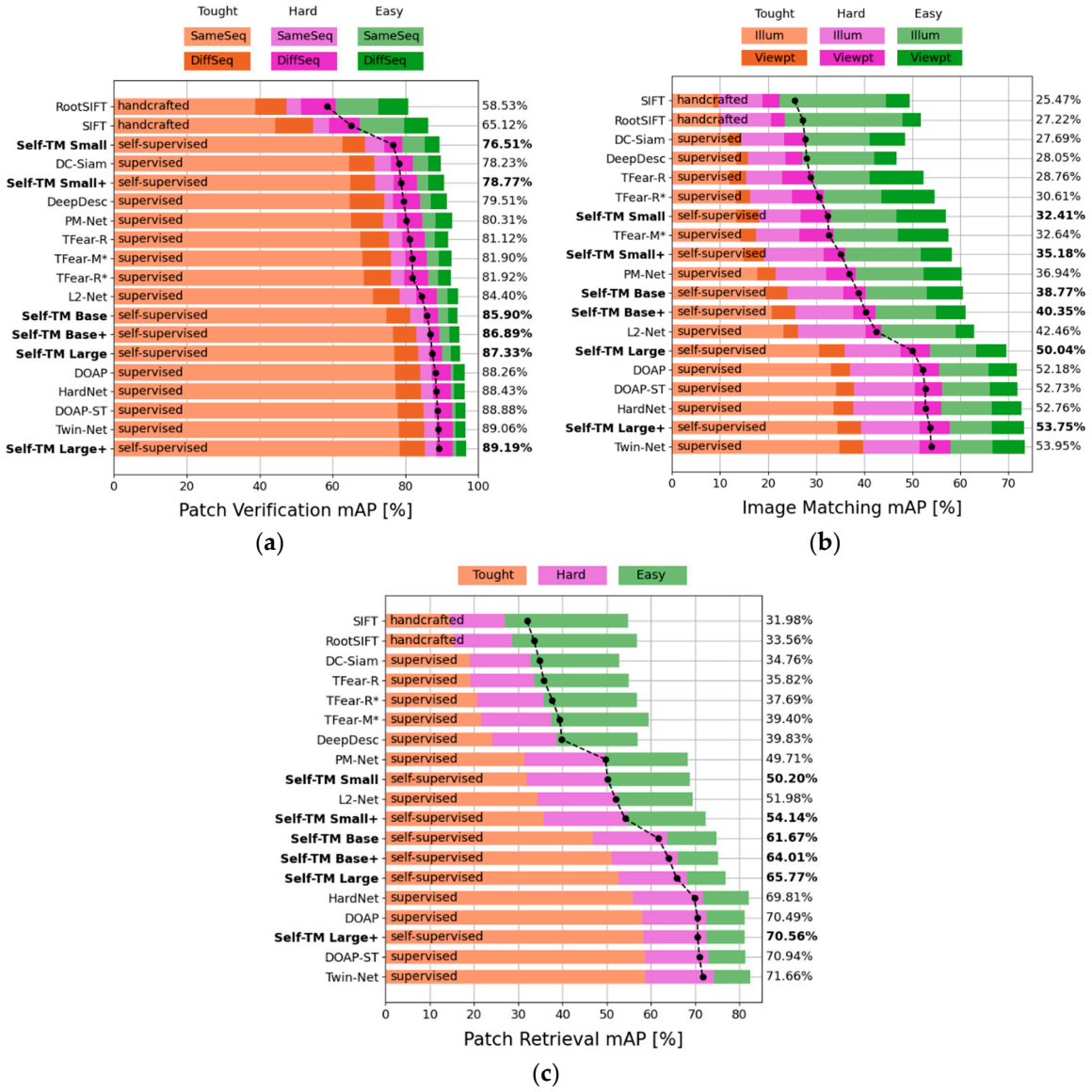


Figure 3. Visual representation of results on HPatches (values, excluding those for Self-TM, are taken from Twin-Net[58]): (**a**) Patch Verification task; (**b**) Image Matching task; (c) Patch Retrieval task; The methods are grouped into the following groups: 'handcrafted', which were manually created by their authors; 'supervised', which used annotated data for their training; 'self-supervised', which did not use any annotations. A plus (+) denotes Self-TM models that are finetuned on the HPatches dataset.

In all three tasks the color of the markers indicates the amount of "geometric noise" $Easy$, $Hard$, $Tough$ contained in the images used. The accuracy percentages represent the average values of the selected subtask.

The experiment used all the Self-TM models: Self-TM Small, Self-TM Base, Self-TM Large, trained on ImageNet1K Train [53] using color and geometric augmentations, where (+) denotes models that are finetuned on the HPatches dataset.

These results demonstrate robust generalization of all the Self-TM models, where Self-TM Large exhibits impressive performance on all three tasks - competitive with previous results of models trained with annotated data (supervised training), approaching the best.

The performance of the Self-TM Large+ finetuned on HPatches even outperforms the previous results on the Patch Verification task, and on Image Matching it gives the first place with a difference of only 0.2% mAP.

 doi:10.20944/preprints202411.0478.v1

4.3. Image Matching

Image matching by sparse methods is a suitable task for verifying the high degree of generalization capabilities of Self-TM models, also being very close to template matching. Here, matching is performed by finding sparse keypoints matches in a set of images (Figure 4).

To demonstrate the effectiveness of Self-TM, we use two well-known databases, namely:

- MegaDepth [44] is a large-scale outdoor image dataset. Ground truth matches between images are computed using Structure-From-Motion [59] algorithms. Test data used contains 1500 predefined image pairs, following previous work [34].
- ScanNet [45] which contains many indoor images composed of challenging scenes with low texture and large perspective changes. Here again, test data used contains 1500 predefined image pairs, following [34].

Both datasets contain only out-of-domain images different from the Self-TM training data.

We perform estimation of the matching accuracy between each pair of images by finding the camera pose estimation and computing the essential matrix with the RANSAC [60] algorithm, then decomposing it into rotation and translation. Finally, the angular error between the calculated and real rotation between all pairs of images is reported, represented by AUC (area under the ROC Curve) at thresholds of 5°, 10° and 20°, see [44,45]. AUC summarizes the performance of classification models with different classification thresholds using the ratio of True Positive Rate (TPR) to False Positive Rate (FPR).

Since Self-TM itself is a convolutional neural network model, it cannot be applied alone to solve the image matching tasks. Therefore, it is integrated into the OmniGlue [31] framework, designed and focused on a high degree of generalization using DINOv2 foundation model [37].

(**a**) (**b**)

Figure 4. Comparison of (**a**) OmniGlue [31] and (**b**) OmniGlue + Self-TM Base in finding keypoints matches in an image with out-of-training-domain modality. For the purpose of visualization, matches with high "confidence" are not visualized to make the errors visible.

More generally, OmniGlue [31] uses frozen weights of DINOv2 [37] and SuperPoint [27] to extract sparse keypoints and their features, which are used to construct an internal graph between each image keypoints and an external graph between all the potential correspondences. Both graphs are fed to a learning module containing multiple blocks with two attention layers ending with a layer giving the potential corresponding keypoints.

Self-TM is included in the OmniGlue framework [31] replacing DINOv2 [37] without any further training. Thus, a direct comparison between Self-TM and DINOv2 [37] features in terms of quality and generalization is performed.

Table 7 presents a comparison between the novel OmniGlue + Self-TM framework against the original OmniGlue [31] and various competing methods (requiring training) following the experiments performed in [31]. These experiments also include traditional approaches like SIFT [24]

 doi:10.20944/preprints202411.0478.v1

and SuperPoint [27] still used in tasks where training is not possible. In these hand-crafted rules a Mutual Nearest Neighbor is used to find a match between keypoints.

Table 7. Comparison between the presented novel framework OmniGlue + Self-TM against OmniGlue [31] and various competing methods (requiring training). Results (excluding those of OmniGlue + Self-TM Small/Base/Large) are taken from [31].

	Method	MegaDepth-1500 AUC@5° / 10° / 20°	ScanNet AUC@5° / 10° / 20°
Descriptors with hand-crafter rules	SIFT [24] + MNN	25.8 / 41.5 / 54.2	1.7 / 4.8 / 10.3
	SuperPoint [27] + MNN	31.7 / 46.8 / 60.1	7.7 / 17.8 / 30.6
Sparse methods	SuperGlue [28]	42.2 / 61.2 / 76.0	10.4 / 22.9 / 37.2
	LightGlue [30]	47.6 / 64.8 / 77.9	15.1 / 32.6 / 50.3
	OmniGlue [31]	47.4 / 65.0 / 77.8	14.0 / 28.9 / 44.3
	OmniGlue + Self-TM Small Relative gain (in %) over OmniGlue	48.2 / 64.7 / 73.8 +1.8 / -0.4 / -5.1	15.8 / 29.4 / 43.4 +13.0 / +1.8 / -2.0
	OmniGlue + Self-TM Base Relative gain (in %) over OmniGlue	56.7 / 69.4 / 78.1 +19.6 / +6.7 / +0.3	22.0 / 34.8 / 47.0 +57.1 / +20.5 / +6.2
	OmniGlue + Self-TM Large Relative gain (in %) over OmniGlue	**59.8 / 70.6 / 78.4** +26.2 / +8.7 / +0.8	**26.6 / 37.7 / 48.4** +90.1 / +30.3 / +9.2

Table 7 shows a significant advantage of OmniGlue + Self-TM Large/Base over OmniGlue [31] on both MegaDepth [44] and ScanNet [45] datasets. OmniGlue + Self-TM Small improves accuracy at AUC @5° but lags behind at AUC @20°.

In addition to the accuracy evaluation the comparison of DINOv2 [37] and Self-TM should also consider their architectures size, which directly affects training time and inference speed. The DINOv2 [37] architecture used is ViT-14-base [18] having 87M parameters compared to Self-TM Base having 40M and Self-TM Large having 130M parameters.

Considering the results of Table 7 and the size of the architectures used, Self-TM gives a higher degree of generalization than DINOv2 [37] (ViT-14-base) and at the same time provides a more optimized architecture - in number of trainable parameters and speed (Table 8).

Table 8. Inference speed and architecture size comparison of DINOv2 and Self-TM. The experiment is performed with input images of various resolutions (multiples of 14 due to DINOv2 limitation) on an Intel Xeon Gold 5222 processor (3.80GHz) and no graphics card is used.

Model	Num of parameters	Inference speed at various input resolution		
		364 × 238 pixels	742 × 490 pixels	1498 × 994 pixels
Self-TM (Small)	13M	212 ms	659 ms	2 481 ms
Self-TM (Base)	40M	244 ms	914 ms	3 432 ms
DINOv2 (ViT-14-base)	87M	445 ms	3 065 ms	38 709 ms
Self-TM (Large)	130M	377 ms	1 268 ms	4 706 ms

 doi:10.20944/preprints202411.0478.v1

5. Implementation Details

5.1. Two Step Training

Table 9 aims to present our experiments based on which the iterative training approach is developed (two successive steps with different image augmentations: color, color and geometric). HPathces dataset [43] is used, which effectively evaluates feature maps properties as local descriptors for finding matches between different image patches in corresponding images.

Table 9. Results of trained Self-TM Base models on HPatches [43] using different combinations of training data (ImageNet-1k [53], HPatches [43]) and applied augmentations (color, color and geometric).

Model	#	Initial weights	Dataset	Augmentations	Patch Verification mAP %	Image Matching mAP %	Patch Retrieval mAP %
Self-TM Base	1	Random init	HPatches	color	64.15	8.32	25.74
	2	Random init	HPatches	color and geometric	65.04	9.95	28.86
	3	HPatches (color)	HPatches	color and geometric	70.07	11.14	30.42
	4	Random init	ImageNet	color	65.19	21.33	37.01
	5	ImageNet (color)	ImageNet	color and geometric	85.90	38.77	61.67
	6	ImageNet (color)	HPatches	color	66.09	21.97	37.79
	7	ImageNet (color)	HPatches	color and geometric	78.97	29.85	50.30
	8	ImageNet (color and geometric)	HPatches	color and geometric	86.89	40.35	64.01

Our experiment uses Self-TM Base model trained from scratch on ImageNet or HPatches dataset using color or color and geometric augmentations, and the same one finetuned on the same data but using weights of already trained models on: HPatches (color); ImageNet (color); ImageNet (color and geometric)

The iterative training approach (Table 9 row 3), where the model is first trained on HPatches with color augmentations and then finetuned with color and geometric augmentations increases the accuracy in: Patch Verification +7.73%; Image Matching +11.96%; Patch Retrieval +5.41% over the model trained directly on HPatches with color and geometric augmentations (Table 9 row 2). The reason for this lies in the fact that geometric augmentations introduce additional variability in the data, which the model cannot initially overcome.

HPatches contain scenes (116 groups of scenes with 6 images in each) with different illumination and geometric augmentations, where models trained with color and geometric augmentations, respectively, have increased accuracy than those using only color augmentations:

- ImageNet training (Table 8 row 5 vs. row 4) results on increased model accuracy such as: Patch Verification +31.77%; Image Matching +81.76%; Patch Retrieval +66.63%;
- HPatches training (Table 9 row 3 vs. row 1) results on increased model accuracy such as: Patch Verification +9.23%; Image Matching +33.89%; Patch Retrieval +18.18%.

 doi:10.20944/preprints202411.0478.v1

High degree of generalization can be explained by the high number of images and the high diversity of classes (1000 classes) covering a sufficient number of visual concepts in ImageNet. Comparing ImageNet and HPatches training (Table 9 row 5 vs. row 3), a significant advantage is observed for the model trained on ImageNet: Patch Verification +22.59%; Image Matching +248.03%; Patch Retrieval +102.73%, despite out-of-domain training data compared to the HPatches data used for test evaluation.

When accuracy increase is required, Self-TM allows for easy finetuning with a small number of images to bring in precise information about the objects and visual concepts to be detected. Comparing ImageNet training result and the same model finetuning with HPatches (Table 9 row 8 vs. row 5) shows an increase in accuracy: Patch Verification +1.15%; Image Matching +4.08%; Patch Retrieval +3.79%.

5.2. Model Architecture

Computer vision field has seen increased model refinement based on improved architectures. They all focus on creating general-purpose visual features that perform well on images of out-of-domain modalities and various tasks. Such models are designed to cover different types of tasks, which usually results in increased size and complexity due to incorporation of additional layers for each type of task (classification, segmentation, tracking, etc.).

The current work shows that designing an architecture targeting a domain of tasks (object detection, image registration, object tracking, sparse and dense image matching) significantly reduces the network size, simplifies the training and inference, increases accuracy and retains its generalization capability, performing well on out-of-domain images.

Our proposed architecture (described in Section 3) uses ConvNeXt [14] blocks with a modified filter size and stride. Down sampling blocks now use 3 × 3 filters and stride 3, which in terms of a receptive field gives a layer N neuron field of view of size 3 by 3 on layer $N-1$. This results in smoother layer-to-layer hierarchical propagation of activations and precise centering due to the size oddness.

The layers used, following ConvNeXt [14], are:

- Standard two-dimensional convolution (Conv2D) using a certain number of filters performing scalar multiplication on their input data;
- Linear layer (Linear), representing a fully connected layer where each neuron is connected to each neuron from the previous layer;
- Normalization layer (Norm) [46], a faster alternative to Batch Normalization [61], used to normalize network weights, in order to reduce the training time;
- Non-linear activation function GELU [47], used in all modern transformers;

A template positioning accuracy comparison of the proposed Self-TM architecture versus ConvNeXt, is shown in Table 10 via the pixel displacements D_L_{First} and D_L_{Last} between the calculated positions $pred\_p_1$ и $pred\_p_N$, relative to the ground truth positions gt_p_1 and gt_p_N of the template. Here D_L_{Fisrt} represents the average displacement using the first layer feature maps and D_L_{Last} – using the last (the deepest) layer feature maps.

Table 10. Comparison of Self-TM Base and ConvNeXt-S [14] trained on ImageNet-1k [53], with various augmentations (color, color and geometric) applied. D_L_{First}, D_L_{Last} displacements (in pixels) are calculated using the first and last layers, respectively.

Model	Num of parameters	Dataset	Augmentations	D_L_{First} pixels	D_L_{Last} pixels
ConvNeXt-S [14]	50M	ImageNet	color	1.119	0.418
			color and geometric	2.515	0.542

 doi:10.20944/preprints202411.0478.v1

Self-TM Base Relative gain over ConvNeXt-S [14]	40M -20.00%	ImageNet	color	0.577 -48.44%	0.156 -62.68%
			color and geometric	1.752 -30.39%	0.338 -37.64%

Here Self-TM Base (40 million parameters) and ConvNeXt-S (50 million parameters) are used as they are closest in number of parameters. The result shows that Self-TM Base having 20% less parameters manages to match the templates more accurately by 48.44% using only color augmentations and by 30.39% with geometric augmentations also applied, respectively.

Additional comparison between Self-TM Base and ConvNeXt-S is shown in Table 11, aiming to evaluate the properties of feature maps as local descriptors for detecting matches between different image patches in their corresponding images, using the HPatches dataset [43].

Table 11. Self-TM Base and ConvNeXt-S [14] training results using different combinations of training data (ImageNet-1k [53], HPatches [43]) and applied augmentations (color, color and geometric).

Model	Initial weights	Dataset	Augmentations	Patch Verification mAP %	Image Matching mAP %	Patch Retrieval mAP %
ConvNeXt-S [14]	Random init	ImageNet	color	63.00	16.99	33.49
	ImageNet (color)	ImageNet	color and geometric	83.31	32.94	58.42
	ImageNet (color and geometric)	HPatches	color and geometric	84.39	34.88	60.61
Self-TM Base Relative gain over ConvNeXt-S [14]	Random init	ImageNet	color	65.19 +3.48%	21.33 +25.54%	37.01 +10.51%
	ImageNet (color)	ImageNet	color and geometric	85.90 +3.11%	38.77 +17.70%	61.67 +5.56%
	ImageNet (color and geometric)	HPatches	color and geometric	86.89 +2.96%	40.35 +15.68%	64.01 +5.61%

Results of multiple trainings steps on each model (sequentially with different augmentations) are presented starting with ImageNet training with color augmentation and finetuning it to a model trained on HPatches with color and geometric transformations.

Despite the smaller number of parameters in Self-TM Base, increased accuracy is observed in each of the three tasks: patch verification, image matching, and patch retrieval. The best result using a model trained on HPatches with color and geometric augmentations shows an increase in accuracy by 2.96% (from 84.39 to 86.89) in Patch Verification, by 15.68% (from 34.88 to 40.35) in Image Matching and by 5.61% (from 60.61 to 64.01) on Patch Retrieval.

6. Conclusions

The present work introduces a novel end-to-end approach using a family of foundation models **Sel**f-Supervised **F**oundation Model for **T**emplate **M**atching (Self-TM) for precise template matching, image retrieval, dense template matching and sparse image matching using hierarchical propagation of activations from the last to the first layer in an efficient-sized convolutional neural network.

Trained on ImageNet-1K, without any annotations, Self-TM provides robust generalization capabilities out-of-domain data involving challenging geometric augmentations. On the sparse image matching task, Self-TM Base significantly outperforms DINOv2 [37] by +19.6% / +6.7% / +0.3% (AUC@5° / 10° / 20°) on MegaDepth-1500 [44], and on ScanNet-1500 [45] by +57.1% / +20.5% / +6.2%

 doi:10.20944/preprints202411.0478.v1

(AUC@5° / 10° / 20°). On HPatches [43], Self-TM has competitive performance compared to methods using a supervised learning approach. When an increase in accuracy is required, the model is easily fine-tuned, outperforming the previous supervised results in the patch verification task on HPatches.

Code and Data: We shared source code using the GitHub repository for this project: https://github.com/anhristov/self-tm.

Author Contributions: Conceptualization, A.H.; methodology, A.H.; exploration, A.H., D.D. and M.N.; formal analysis, A.H. and D.D.; writing—original draft preparation, A.H.; writing—review and editing, D.D., M.N. and A.H.; visualization, A.H.; supervision, M.N. and D.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A simple framework for contrastive learning of visual representations. International Conference on Machine Learning 2020, 1, 1597–1607. https://doi.org/10.48550/arXiv.2002.05709.
2. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R. Momentum contrast for unsupervised visual representation learning. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019. https://doi.org/10.48550/arxiv.1911.05722.
3. He, K.; Chen, X.; Xie, S.; Li, Y.; Dollar, P.; Girshick, R. Masked autoencoders are scalable vision learners. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2022. https://doi.org/10.1109/cvpr52688.2022.01553.
4. Bertinetto, L.; Valmadre, J.; Henriques, J. F.; Vedaldi, A.; Torr, P. H. S. Fully-Convolutional Siamese networks for object tracking. In Lecture notes in computer science; 2016; pp 850–865. https://doi.org/10.1007/978-3-319-48881-3_56.
5. Li, B.; Yan, J.; Wu, W.; Zhu, Z.; Hu, X. High performance visual tracking with Siamese Region Proposal network. IEEE Conference on Computer Vision and Pattern Recognition 2018. https://doi.org/10.1109/cvpr.2018.00935.
6. He, A.; Luo, C.; Tian, X.; Zeng, W. A twofold siamese network for real-time object tracking. IEEE Conference on Computer Vision and Pattern Recognition 2018. https://doi.org/10.48550/arxiv.1802.08817.
7. Valmadre, J.; Bertinetto, L.; Henriques, J.; Vedaldi, A.; Torr, P. H. S. End-to-end representation learning for correlation filter based tracking. IEEE Conference on Computer Vision and Pattern Recognition 2017. https://doi.org/10.1109/cvpr.2017.531.
8. Li, B.; Wu, W.; Wang, Q.; Zhang, F.; Xing, J.; Yan, J. SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks. arXiv (Cornell University) 2018. https://doi.org/10.48550/arxiv.1812.11703.
9. Zhu, Z.; Wang, Q.; Li, B.; Wu, W.; Yan, J.; Hu, W. Distractor-Aware Siamese networks for visual object tracking. In Lecture notes in computer science; 2018; pp 103–119. https://doi.org/10.1007/978-3-030-01240-3_7.
10. Fan, H.; Ling, H. Siamese cascaded region proposal networks for real-time visual tracking. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2018. https://doi.org/10.48550/arxiv.1812.06148.
11. Song, Y.; Ma, C.; Gong, L.; Zhang, J.; Lau, R.; Yang, M.-H. CREST: Convolutional Residual Learning for Visual Tracking. IEEE International Conference on Computer Vision 2017. https://doi.org/10.48550/arxiv.1708.00225.
12. Guo, D.; Wang, J.; Cui, Y.; Wang, Z.; Chen, S. SIAMCAR: Siamese Fully Convolutional Classification and Regression for visual tracking. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020. https://doi.org/10.1109/cvpr42600.2020.00630.
13. Hu, W.; Wang, Q.; Zhang, L.; Bertinetto, L.; Torr, P. H. S. SiamMask: a framework for fast online object tracking and segmentation. PubMed 2023, 45 (3), 3072–3089. https://doi.org/10.1109/tpami.2022.3172932.
14. Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; Xie, S. A ConvNet for the 2020s. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2022. https://doi.org/10.1109/cvpr52688.2022.01167.

 doi:10.20944/preprints202411.0478.v1

15. Steiner, A.; Kolesnikov, A.; Zhai, X.; Wightman, R.; Uszkoreit, J.; Beyer, L. How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers. arXiv (Cornell University) 2021. https://doi.org/10.48550/arxiv.2106.10270.
16. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition 2016. https://doi.org/10.1109/cvpr.2016.90.
17. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. arXiv (Cornell University) 2017, 30, 5998–6008.
18. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv (Cornell University) 2020. https://doi.org/10.48550/arxiv.2010.11929.
19. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. Training data-efficient image transformers & distillation through attention. International Conference on Machine Learning 2020. https://doi.org/10.48550/arxiv.2012.12877.
20. Hisham, M. B.; Yaakob, S. N.; Raof, R. A. A.; Nazren, A. B. A.; Wafi, N. M. Template matching using sum of squared difference and normalized cross correlation. IEEE Student Conference on Research and Development (SCOReD) 2015. https://doi.org/10.1109/scored.2015.7449303.
21. Niitsuma, H.; Maruyama, T. Sum of absolute difference implementations for image processing on FPGAs. International Conference on Field Programmable Logic and Applications 2010, 33, 167–170. https://doi.org/10.1109/fpl.2010.40.
22. Papageorgiou, C. P.; Oren, M.; Poggio, T. A general framework for object detection. Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271) 1998. https://doi.org/10.1109/iccv.1998.710772.
23. Di Stefano, L.; Mattoccia, S.; Tombari, F. ZNCC-based template matching using bounded partial correlation. Pattern Recognition Letters 2005, 26 (14), 2129–2134. https://doi.org/10.1016/j.patrec.2005.03.022.
24. Lowe, D. G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 2004, 60 (2), 91–110. https://doi.org/10.1023/b:visi.0000029664.99615.94.
25. Bay, H.; Tuytelaars, T.; Van Gool, L. SURF: Speeded up robust features. In European Conference on Computer Vision; 2006; pp 404–417. https://doi.org/10.1007/11744023_32.
26. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. International Conference on Computer Vision 2011. https://doi.org/10.1109/iccv.2011.6126544.
27. DeTone, D.; Malisiewicz, T.; Rabinovich, A. SuperPoint: Self-supervised interest point detection and description. IEEE Conference on Computer Vision and Pattern Recognition Workshops 2018. https://doi.org/10.1109/cvprw.2018.00060.
28. Sarlin, P.-E.; DeTone, D.; Malisiewicz, T.; Rabinovich, A. SuperGlue: Learning feature matching with graph neural networks. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020. https://doi.org/10.1109/cvpr42600.2020.00499.
29. Cuturi, M. Sinkhorn Distances: Lightspeed computation of optimal transport. Neural Information Processing Systems 2013, 26, 2292–2300.
30. Lindenberger, P.; Sarlin, P.-E.; Pollefeys, M. LightGlue: Local feature matching at light speed. International Conference on Computer Vision 2023. https://doi.org/10.1109/iccv51070.2023.01616.
31. Jiang, H.; Karpur, A.; Cao, B.; Huang, Q.; Araujo, A. OmniGlue: Generalizable feature matching with foundation model guidance. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2024, abs/2308.08479, 19865–19875. https://doi.org/10.1109/cvpr52733.2024.01878.
32. Caron, M.; Touvron, H.; Misra, I.; Jegou, H.; Mairal, J.; Bojanowski, P.; Joulin, A. Emerging Properties in Self-Supervised Vision Transformers. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) 2021. https://doi.org/10.1109/iccv48922.2021.00951.
33. Chen, H.; Luo, Z.; Zhou, L.; Tian, Y.; Zhen, M.; Fang, T.; McKinnon, D.; Tsin, Y.; Quan, L. ASpanFormer: Detector-Free Image Matching with Adaptive Span Transformer. In Lecture notes in computer science; 2022; pp 20–36. https://doi.org/10.1007/978-3-031-19824-3_2.
34. Sun, J.; Shen, Z.; Wang, Y.; Bao, H.; Zhou, X. LOFTR: Detector-Free local feature matching with transformers. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021. https://doi.org/10.1109/cvpr46437.2021.00881.

35. Edstedt, J.; Athanasiadis, I.; Wadenbäck, M.; Felsberg, M. DKM: Dense Kernelized Feature Matching for geometry estimation. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2023. https://doi.org/10.1109/cvpr52729.2023.01704.
36. Truong, P.; Danelljan, M.; Timofte, R.; Van Gool, L. PDC-NET+: Enhanced Probabilistic Dense Correspondence Network. IEEE Transactions on Pattern Analysis and Machine Intelligence 2023, 45 (8), 10247–10266. https://doi.org/10.1109/tpami.2023.3249225.
37. Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; Assran, M.; Ballas, N.; Galuba, W.; Howes, R.; Huang, P.-Y.; Li, S.-W.; Misra, I.; Rabbat, M.; Sharma, V.; Synnaeve, G.; Xu, H.; Jegou, H.; Mairal, J.; Labatut, P.; Joulin, A.; Bojanowski, P. DINOv2: Learning Robust Visual Features without Supervision. arXiv (Cornell University) 2023. https://doi.org/10.48550/arxiv.2304.07193.
38. Van Den Oord, A.; Li, Y.; Vinyals, O. Representation Learning with Contrastive Predictive Coding. arXiv (Cornell University) 2018. https://doi.org/10.48550/arxiv.1807.03748.
39. Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; Dollár, P.; Girshick, R. Segment anything. IEEE/CVF International Conference on Computer Vision 2023. https://doi.org/10.48550/arxiv.2304.02643.
40. Ravi, N.; Gabeur, V.; Hu, Y.-T.; Hu, R.; Ryali, C.; Ma, T.; Khedr, H.; Rädle, R.; Rolland, C.; Gustafson, L.; Mintun, E.; Pan, J.; Alwala, K. V.; Carion, N.; Wu, C.-Y.; Girshick, R.; Dollár, P.; Feichtenhofer, C. SAM 2: Segment anything in images and videos. arXiv (Cornell University) 2024. https://doi.org/10.48550/arxiv.2408.00714.
41. Wang, X.; Zhang, X.; Cao, Y.; Wang, W.; Shen, C.; Huang, T. SegGPT: towards segmenting everything in context. IEEE/CVF International Conference on Computer Vision 2023. https://doi.org/10.1109/iccv51070.2023.00110.
42. Wang, X.; Wang, W.; Cao, Y.; Shen, C.; Huang, T. Images speak in images: A generalist painter for in-context visual Learning. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2023. https://doi.org/10.1109/cvpr52729.2023.00660.
43. Balntas, V.; Lenc, K.; Vedaldi, A.; Mikolajczyk, K. HPatches: a benchmark and evaluation of handcrafted and learned local descriptors. IEEE Conference on Computer Vision and Pattern Recognition 2017. https://doi.org/10.1109/cvpr.2017.410.
44. Li, Z.; Snavely, N. MegaDepth: Learning single-view depth prediction from internet photos. IEEE Conference on Computer Vision and Pattern Recognition 2018. https://doi.org/10.48550/arxiv.1804.00607.
45. Dai, A.; Chang, A. X.; Savva, M.; Halber, M.; Funkhouser, T.; Niessner, M. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. IEEE Transactions on Information Theory 2017. https://doi.org/10.1109/cvpr.2017.261.
46. Ba, J. L.; Kiros, J. R.; Hinton, G. E. Layer normalization. arXiv (Cornell University) 2016. https://doi.org/10.48550/arxiv.1607.06450.
47. Nair, V.; Hinton, G. E. Rectified linear units improve restricted Boltzmann machines. International Conference on Machine Learning 2010, 807–814.
48. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) 2021. https://doi.org/10.1109/iccv48922.2021.00986.
49. Tan, M.; Le, Q. V. EfficientNet: Rethinking model scaling for convolutional neural networks. International Conference on Machine Learning 2019. https://doi.org/10.48550/arxiv.1905.11946.
50. Radosavovic, I.; Kosaraju, R. P.; Girshick, R.; He, K.; Dollar, P. Designing network design spaces. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020. https://doi.org/10.1109/cvpr42600.2020.01044.
51. Tan, M.; Le, Q. EfficientNetV2: Smaller models and faster training. International Conference on Machine Learning 2021.
52. Kolesnikov, A.; Beyer, L.; Zhai, X.; Puigcerver, J.; Yung, J.; Gelly, S.; Houlsby, N. Big Transfer (BIT): General visual representation learning. In European Conference on Computer Vision; 2020; pp 491–507. https://doi.org/10.1007/978-3-030-58558-7_29.
53. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, N. K.; Fei-Fei, N. L. ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition 2009. https://doi.org/10.1109/cvpr.2009.5206848.

 doi:10.20944/preprints202411.0478.v1

54. Assran, M.; Duval, Q.; Misra, I.; Bojanowski, P.; Vincent, P.; Rabbat, M.; LeCun, Y.; Ballas, N. Self-Supervised learning from images with a joint-embedding predictive architecture. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2023. https://doi.org/10.1109/cvpr52729.2023.01499.
55. Bardes, A.; Ponce, J.; Lecun, Y. VICRegL: Self-supervised learning of local visual features. Advances in Neural Information Processing Systems 2022. https://doi.org/10.48550/arxiv.2210.01571.
56. Bridle, J. S. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. Advances in Neural Information Processing Systems 1989, 2, 211–217.
57. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. arXiv (Cornell University) 2017. https://doi.org/10.48550/arxiv.1711.05101.
58. Irshad, A.; Hafiz, R.; Ali, M.; Faisal, M.; Cho, Y.; Seo, J. Twin-Net descriptor: twin negative mining with quad loss for Patch-Based matching. IEEE Access 2019, 7, 136062–136072. https://doi.org/10.1109/access.2019.2940737.
59. Schonberger, J. L.; Frahm, J.-M. Structure-from-motion revisited. IEEE Conference on Computer Vision and Pattern Recognition 2016. https://doi.org/10.1109/cvpr.2016.445.
60. Fischler, M. A.; Bolles, R. C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In Elsevier eBooks; 1987; pp 726–740. https://doi.org/10.1016/b978-0-08-051581-6.50070-2.
61. Ioffe, S. Batch renormalization: towards reducing minibatch dependence in Batch-Normalized models. Advances in Neural Information Processing Systems 2017. https://doi.org/10.48550/arxiv.1702.03275.