Efficient Training of Visual Transformers with Small-Size Datasets

Yahui Liu
University of Trento
Fondazione Bruno Kessler
yahui.liu@unitn.it

Enver Sangineto
University of Trento
enver.sangineto@unitn.it

Wei Bi
Tencent AI Lab
victoriabi@tencent.com

Nicu Sebe
University of Trento
niculae.sebe@unitn.it

Bruno Lepri
Fondazione Bruno Kessler
lepri@fbk.eu

Marco De Nadai
Fondazione Bruno Kessler
denadai@fbk.eu

Abstract

Visual Transformers (VTs) are emerging as an architectural paradigm alternative to Convolutional networks (CNNs). Differently from CNNs, VTs can capture global relations between image elements and they potentially have a larger representation capacity. However, the lack of the typical convolutional inductive bias makes these models more data-hungry than common CNNs. In fact, some local properties of the visual domain which are embedded in the CNN architectural design, in VTs should be learned from samples. In this paper, we empirically analyse different VTs, comparing their robustness in a small training-set regime, and we show that, despite having a comparable accuracy when trained on ImageNet, their performance on smaller datasets can be largely different. Moreover, we propose a self-supervised task which can extract additional information from images with only a negligible computational overhead. This task encourages the VTs to learn spatial relations within an image and makes the VT training much more robust when training data are scarce. Our task is used jointly with the standard (supervised) training and it does not depend on specific architectural choices, thus it can be easily plugged in the existing VTs. Using an extensive evaluation with different VTs and datasets, we show that our method can improve (sometimes dramatically) the final accuracy of the VTs. The code will be available upon acceptance.

1 Introduction

Visual Transformers (VTs) are progressively emerging architectures in computer vision as an alternative to standard Convolutional Neural Networks (CNNs), and they have already been applied to many tasks, such as image classification [18, 57, 66, 38, 62, 65, 36, 64], object detection [5, 69, 15], segmentation [53], tracking [40] and image generation [33, 31], to mention a few. These architectures are inspired by the well-known Transformer [59], which is the de-facto standard in Natural Language Processing (NLP) [16, 49], and one of their appealing properties is the possibility to develop a unified information-processing paradigm for both visual and textual domains. A pioneering work in this direction is ViT [18], in which an image is split using a grid of non-overlapping patches, and each patch is linearly projected in the input embedding space, so obtaining a "token". After that, all the tokens are processed by a series of multi-head attention and feed-forward layers, similarly to how (word) tokens are processed in NLP Transformers.

A clear advantage of VTs is the possibility for the network to use the attention layers to model global relations between tokens, and this is the main difference with respect to CNNs, where the

Preprint. Under review.

receptive field of the convolutional kernels locally limits the type of relations which can be learned. However, the increased representation capacity of the VTs comes at a price, which is the lack of the typical CNN inductive biases, based on exploiting the locality, the translation invariance and the hierarchical structure of visual information [38, 62, 65]. As a result, VTs need a lot of data for training, usually more than what is necessary to standard CNNs [18]. For instance, ViT is trained with JFT-300M [18], a (proprietary) huge dataset of 303 million (weakly) labeled high-resolution images, and performs worse than ResNets [27] with similar capacity when trained on ImageNet-1K ($\sim$ 1.3 million samples [51]). This is likely due to the fact that ViT needs to learn some local proprieties of the visual data using more samples than a CNN, while the latter embeds these properties in its architectural design.

In order to alleviate this problem, very recently a second generation of VTs has been independently proposed by different groups [66, 38, 62, 65, 64, 36, 31]. A common idea behind these works is to mix convolutional layers with attention layers, in such a way providing a local inductive bias to the VT. These hybrid architectures enjoy the advantages of both paradigms: attention layers model long-range dependencies, while convolutional operations can emphasise the local properties of the image content. The empirical results shown in most of these works demonstrate that this second-generation VTs can be trained on ImageNet outperforming similar-size ResNets on this dataset [66, 38, 62, 65, 64, 36]. However, it is still not clear what is the behaviour of these networks when trained on medium-small size datasets. In fact, from an application point of view, most of the vision tasks cannot rely on (supervised) datasets whose size is comparable with (or larger than) ImageNet.

In this paper, we compare to each other different second-generation VTs by either training them from scratch or fine-tuning them on medium-small size datasets, and we empirically show that, despite their ImageNet results are basically on par with each other, their classification accuracy with smaller datasets largely varies. Moreover, we propose to use an additional self-supervised *pretext* task and a corresponding loss function in order to accelerate training in a small training-set or few-epochs regime. Specifically, the proposed task is based on (unsupervised) learning the spatial relations between the output-token embeddings. Given an image, we *densely* sample random pairs from the final embedding grid, and, for each pair, we ask the network to guess their relative translation offsets. To solve this task, the network needs to encode both local and contextual information in each embedding. In fact, without local information, embeddings representing different input image patches cannot be distinguished the one from the others, while, without contextual information (aggregated using the attention layers), the task may be ambiguous.

Our task is inspired by ELECTRA [12], an NLP model in which the pretext task is densely defined for each output embedding (Sec. 2). Clark et al. [12] show that their task is more *sample-efficient* than commonly used NLP pretext tasks, and this gain is particularly strong with small-capacity models or relatively smaller training sets. Similarly, we exploit the fact that an image is represented by a VT using multiple token embeddings, and we use their relative distances to define a localization task over a subset of all the possible embedding pairs. In this way, *for a single image forward*, we can compare to each other many embedding pairs and average our localization loss over all of them. Thus, our task is drastically different from those multi-crop strategies proposed, for instance, in SwAV [7], which need to independently forward each input patch through the network.

Since our additional task is self-supervised, our *dense relative localization* loss ($\mathcal{L}_{drloc}$) does not require additional annotation, and we use it jointly with the standard (supervised) cross-entropy as a regularization of the VT training. $\mathcal{L}_{drloc}$ is very easy-to-be-reproduced and, despite this simplicity, it can largely boost the accuracy of the VTs, especially when the VT is either trained from scratch on a small-size dataset, or fine-tuned on a dataset with a large domain-shift with respect to the pretraining ImageNet dataset. In our empirical analysis, based on different training scenarios, a variable amount of training data and three different second-generation VTs, $\mathcal{L}_{drloc}$ has *always* improved the results of the tested baselines, sometimes boosting the final accuracy of tens of points (and up to 45 points).

In summary, the contributions of this paper are the following:

- We empirically compare to each other different very recent VTs, and we show that their behaviour can largely differ when trained with small-size datasets or few training epochs.
- We propose a new, straightforward self-supervised relative localization loss which is used as an additional task for VT training. Using an extensive empirical analysis, we show that our loss is beneficial to speed-up training and increase the generalization ability of different VTs, independently of their specific architectural design or application task.

2 Related work

In this section, we briefly review previous work related to both VTs and self-supervised learning.

Visual Transformers. Despite some previous work in which attention is used inside the convolutional layers of a CNN [61, 29], the first fully-transformer architectures for vision are iGPT [8] and ViT [18]. The former is trained using a "masked-pixel" self-supervised approach, similar in spirit to the common masked-word task used, for instance, in BERT [16] and in GPT [49] (see below). On the other hand, ViT is trained in a supervised way, using a special "class token" and a classification head attached to the final embedding of this token. Both methods are computationally expensive and, despite their very good results when trained on huge datasets, they underperform ResNet architectures when trained from scratch using only ImageNet-1K [18, 8]. VideoBERT [54] is conceptually similar to iGPT, but, rather than using pixels as tokens, each frame of a video is holistically represented by a feature vector, which is quantized using an off-the-shelf pretrained video classification model. DeiT [57] trains ViT using distillation information provided by a pretrained CNN.

The success of ViT has attracted a lot of interest in the vision community, and different variants of this architecture have been recently used in many tasks [57, 53, 33, 11]. However, as mentioned in Sec. 1, the lack of the typical CNN inductive biases in ViT, makes this model difficult to train without using (very) large-size datasets. For this reason, very recently, a second-generation of VTs has focused on hybrid architectures, in which convolutions are used jointly with long-range attention layers [66, 38, 62, 65, 64, 36, 31]. The common idea behind all these works is that the sequence of the individual token embeddings can be shaped/reshaped in a geometric grid, in which the position of each embedding vector corresponds to a fixed location in the input image. Given this geometric layout of the embeddings, convolutional layers can be applied to neighboring embeddings, so encouraging the network to focus on local properties of the image. The main difference among these works concerns *where* the convolutional operation is applied (e.g., only in the initial representations [66] or in all the layers [38, 62, 65, 64, 36], in the token to query/key/value projections [62] or in the forward-layers [65, 36, 31], etc.). In this paper we do not use ViT, being the original ViT architectures too big as number of parameters, and because we focus on training/fine-tuning on datasets whose size is smaller than ImageNet-1K. Conversely, we use three state-of-the-art second-generation VTs (T2T [66], Swin [38] and CvT [62]), for which there is a public implementation. For each of them, we select the model whose number of parameters is comparable with a ResNet-50 [27] (more details in Sec. 3). We do not modify the native architectures because the goal of this work is to propose a pretext task and a loss function which can be easily plugged in existing VTs.

Similarly to the original Transformer [59], in ViT, an (absolute) *positional embedding* is added to the representation of the input tokens. In Transformer networks, positional embedding is used to provide information about the token order, since both the attention and the (individual token based) feed-forward layers are permutation invariant. In [38, 64], *relative* positional embedding [52] is used, where the position of each token is represented relatively to the others. Generally speaking, positional embedding is a representation of the token position which is *provided as input* to the network. Conversely, our relative localization loss exploits the relative positions (of the final VT embeddings) as a *pretext task* to extract additional information without manual supervision.

Self-supervised learning. Reviewing the vast self-supervised learning literature is out of the scope of this paper. However, we briefly mention that self-supervised learning was first successfully applied in NLP, as a means to get supervision from text by replacing costly manual annotations with *pretext* tasks [41, 42]. A typical NLP pretext task consists in masking a word in an input sentence and asking the network to guess which is the masked token [41, 42, 16, 49]. ELECTRA [12] is a *sample-efficient* language model in which the masked-token pretext task is replaced by a discriminative task defined over all the tokens of the input sentence. Our work is inspired by this method, since we propose a pretext task which can be efficiently computed by densely sampling the final VT embeddings.

使用代理任务

In vision, common pretext tasks with still images are based on extracting two different views from the same image (e.g., two different crops) and then considering these as a pair of *positive* images, likely sharing the same semantic content [9]. Current self-supervised vision approaches can be broadly categorised in contrastive learning [58, 28, 9, 25, 56, 60, 20], clustering methods [3, 70, 32, 6, 1, 22, 7], asymmetric networks [24, 10] and feature-decorrelation methods [21, 67, 2, 30]. While the aforementioned approaches are all based on ResNets, very recently, Chen et al. [11] have empirically tested some of these methods with a ViT architecture [18].

One important difference of our proposal with respect to previous work, is that we do not propose a fully-self-supervised method, but we rather use self-supervision jointly with standard supervision (i.e., image labels) in order to regularize VT training, hence our framework is a *multi-task learning* approach [14]. Moreover, our dense relative localization loss is not based on positive pairs, and we do *not* use multiple views of the same image in the current batch, thus our method can be used with standard (supervised) data-augmentation techniques. Specifically, our pretext task is based on predicting the relative positions of pairs of tokens extracted from the same image.

Previous work using localization for self-supervision is based on predicting the input image rotation [23] or the relative position of *adjacent patches* extracted from the same image [17, 46, 47, 43]. For instance, in [46], the network should predict the correct permutation of a grid of 3×3 patches (in NLP, a similar, permutation based pretext task, is *deshuffling* [50]). In contrast, we do not need to extract multiple patches from the same input image, since we can efficiently use the final token embeddings (thus, we need a *single* forward and backward pass per image). Moreover, differently from previous work based on localization pretext tasks, our loss is *densely* computed between many random pairs of (non necessarily adjacent) token embeddings. Note that one of the reasons for which we can use non-adjacent image positions, is that the attention layers of the VT include contextual information in each token representation, thus making the prediction task easier. Finally, in [15], the position of a random query patch is used for self-supervised training a transformer-based object detector [5]. However, the localization loss used in [15] is specific for the final task (object localization) and the specific DETR architecture [5], while our loss is generic and can be plugged in any VT.

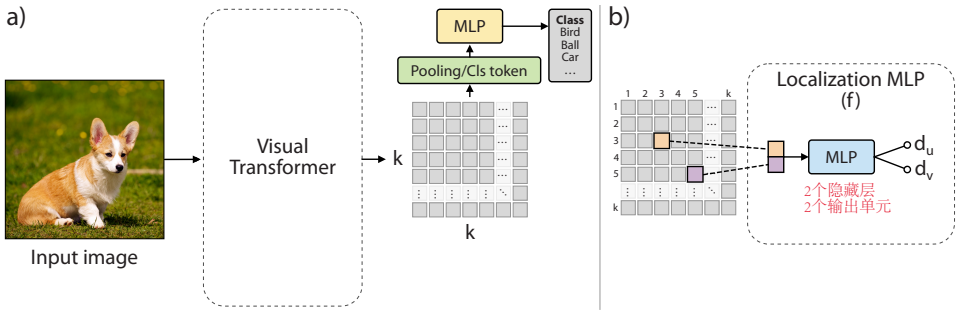

Figure 1: A schematic representation of the VT architecture. (a) A typical second-generation VT. (b) Our localization MLP which takes as input (concatenated) pairs of final token embeddings.

3 Preliminaries

As mentioned in Sec. 1-2, in this paper we focus on second-generation discriminative VTs [66, 38, 62, 65, 64, 36] which are hybrid architectures mixing Transformer-like multi-attention layers with convolutional operations. Without loss of generality, these networks take as input an image which is split in a grid of (possibly overlapping) $K \times K$ patches. Each patch is projected in the input embedding space, obtaining a set of $K \times K$ input *tokens*, fed to the VT. The latter is based on the typical Transformer multi-attention layers [59], which model pairwise relations over the token intermediate representations. Importantly, the attention layers gradually introduce contextual information in each token representation. Differently from a pure Transformer [59], hybrid architectures usually shape or re-shape the sequence of these token embeddings in a spatial grid, which makes it possible to apply convolutional operations over a small set of neighboring token embeddings. Using convolutions with a stride greater than 1 and/or pooling operations, the resolution of the initial $K \times K$ token grid can possibly be reduced, thus simulating the hierarchical structure of a CNN. We assume that the final embedding grid has a resolution of $k \times k$ (where, usually, $k \leq K$), see Fig. 1 (a).

The final $k \times k$ grid of embeddings represents the input image and it is used for the discriminative task. For instance, some methods include an additional "class token" which collects contextual information over the whole grid [66, 62, 65, 64, 36], while others [38] apply an average global pooling over the final grid to get a compact representation of the whole image. Finally, a standard, small MLP head takes as input the whole image representation and it outputs a posterior distribution over the set of the target classes (Fig. 1 (a)). The VT is trained using a standard cross-entropy loss ($\mathcal{L}_{ce}$), computed using these posteriors and the image ground-truth labels.

When we plug our relative localization loss (Sec. 4) in an existing VT, we always use the native VT architecture of each tested method, without any change apart from the dedicated localization MLP (see Sec. 4). For instance, we use the class token when available, or the average pooling layer when it is not, and on top of these we use the cross-entropy loss. We also keep the positional embedding (Sec. 2) for those VTs which add this information to the tokens (see Sec. 4.1 for a discussion about this choice). The only architectural change we do is to downsample the final embedding grid of T2T [66] and CvT [62] to make them of the same size as that used in Swin [7]. Specifically, in Swin, the final grid has a resolution of 7×7 ($k = 7$), while, in T2T and in CvT, it is 14×14. Thus, in T2T and in CvT, we use a 2×2 average pooling (*without* learnable parameters) and we get a final 7×7 grid for all the three tested architectures. This pooling operation is motivated in Sec. 4.1, and it is used only together with our localization task (it does not affect the posterior computed by the classification MLP). Finally, note that T2T uses convolutional operations only in the input stage, and it outputs a sequence of $14 \times 14 = 196$ embeddings, corresponding to its 14×14 input grid. In this case, we first reshape the sequence and then we use pooling.

4 Dense Relative Localization task

The goal of our regularization loss is to encourage the VT to learn spatial information without using additional manual annotations. We achieve this by *densely* sampling *multiple* embedding pairs *for each image* and asking the network to guess their relative positions. In more detail, given an image x, we denote its corresponding $k \times k$ grid of final embeddings (Sec. 3), as $G_x = \{\mathbf{e}_{i,j}\}_{1 \leq i,j \leq k}$, where $\mathbf{e}_{i,j} \in \mathbb{R}^D$, and D is the dimension of the embedding space. For each G_x, we randomly sample multiple pairs of embeddings and, for each pair $(\mathbf{e}_{i,j}, \mathbf{e}_{p,h})$, we compute the 2D normalized target translation offset $(t_u, t_v)^T$, where:

$$t_u = \frac{|i-p|}{k}, \quad t_v = \frac{|j-h|}{k}, \quad (t_u, t_v)^T \in [0,1]^2. \tag{1}$$

The selected embedding vectors $\mathbf{e}_{i,j}$ and $\mathbf{e}_{p,h}$ are concatenated and input to a small MLP (f), with two hidden layers and two output neurons, one per spatial dimension (Fig. 1 (b)), which predicts the relative distance between position (i,j) and position (p,h) on the grid. Let $(d_u, d_v)^T = f(\mathbf{e}_{i,j}, \mathbf{e}_{p,h})^T$. Given a mini-batch B of n images, our *dense relative localization loss* is:

$$\mathcal{L}_{drloc} = \sum_{x \in B} \mathbb{E}_{(\mathbf{e}_{i,j}, \mathbf{e}_{p,h}) \sim G_x} [|(t_u, t_v)^T - (d_u, d_v)^T|_1]. \tag{2}$$

In Eq. (2), for each image x, the expectation is computed by sampling uniformly at random m embedding pairs $(\mathbf{e}_{i,j}, \mathbf{e}_{p,h})$ in G_x, and averaging the L_1 loss between the corresponding $(t_u, t_v)^T$ and $(d_u, d_v)^T$.

$\mathcal{L}_{drloc}$ is added to the standard cross-entropy loss ($\mathcal{L}_{ce}$) of each native VT (Sec. 3). The final loss is: $\mathcal{L}_{tot} = \mathcal{L}_{ce} + \lambda \mathcal{L}_{drloc}$. We use $\lambda = 0.1$ in all the experiments with both T2T and CvT, and $\lambda = 0.5$ in case of Swin.

4.1 Discussion

Intuitively, $\mathcal{L}_{drloc}$ transforms the relative positional embedding (Sec. 2), used, for instance, in Swin [38], in a pretext task, asking the network to guess which is the relative distance of a random subset of all the possible token pairs. Thus a question may arise: is the relative positional embedding used in some VTs sufficient for the localization MLP (f) to solve the localization task? The experiments presented in Sec. 5.2-5.3 show that, when we plug $\mathcal{L}_{drloc}$ on CvT, in which *no kind* of positional embedding is used [62], the relative accuracy boost is usually *smaller* than in case of Swin, confirming that the relative positional embedding, used in the latter, is not sufficient to make our task trivial.

In Sec. 3, we mentioned that, in case of Swin, the final embedding grid has a 7×7 resolution, while for the other two VTs we consider here (T2T and CvT), we average-pool their 14×14 grids and we obtain a final 7×7 grid G_x. In fact, in preliminary experiments with both T2T and CvT at their original 14×14 resolution, we observed a very slow convergence of $\mathcal{L}_{drloc}$. We presume this is due to the fact that, with a finer grid, the localization task is harder. This makes more difficult the convergence of f, and it likely generates noisy gradients which are backpropagated through the whole VT (see also Sec. 5.1). We leave this for future investigation and, in the rest of this article, we always assume that our pretext task is computed with a 7×7 grid G_x.

4.2 Loss variants

In this section, we present different variants of the relative localization loss which will be empirically analyzed in Sec. 5.1.

The first variant consists in including negative target offsets:

$$t'_u = \frac{i-p}{k}, \quad t'_v = \frac{j-h}{k}, \quad (t'_u, t'_v)^T \in [-1,1]^2. \tag{3}$$

Replacing $(t_u, t_v)^T$ in Eq. (2) with $(t'_u, t'_v)^T$ computed as in Eq. (3), and keeping all the rest unchanged, we obtain the first variant, which we call $\mathcal{L}^*_{drloc}$.

In the second variant, we transform the regression task in Eq. (2) in a classification task, and we replace the L_1 loss with the cross-entropy loss. In more detail, we use as target offsets:

$$c_u = i - p, \quad c_v = j - h, \quad (c_u, c_v)^T \in \{-k, ..., k\}^2, \tag{4}$$

and we associate each of the $2k+1$ discrete elements in $C = \{-k, ..., k\}$ with a "class". Accordingly, the localization MLP f is modified by replacing the 2 output neurons with 2 different sets of neurons, one per spatial dimension (u and v). Each set of neurons represents a discrete offset prediction over the $2k+1$ "classes" in C. Softmax is applied *separately* to each set of $2k+1$ neurons, and the output of f is composed of two posterior distributions over C: $(\mathbf{p}_u, \mathbf{p}_v)^T = f(\mathbf{e}_{i,j}, \mathbf{e}_{p,h})^T$, where $\mathbf{p}_u, \mathbf{p}_v \in [0,1]^{2k+1}$. Eq. (2) is then replaced by:

$$\mathcal{L}^{ce}_{drloc} = -\sum_{x \in B} \mathbb{E}_{(\mathbf{e}_{i,j}, \mathbf{e}_{p,h}) \sim G_x} [log(\mathbf{p}_u[c_u] + log(\mathbf{p}_v[c_v]], \tag{5}$$

where $\mathbf{p}_u[c_u]$ indicates the c_u-th element of $\mathbf{p}_u$ (and similarly for $\mathbf{p}_v[c_v]$).

Note that, using the cross-entropy loss in Eq. (5), corresponds to considering C an unordered set of "categories". This implies that prediction errors in $\mathbf{p}_u$ (and $\mathbf{p}_v$) are independent of the "distance" with respect to the ground-truth c_u (respectively, c_v). In order to alleviate this problem, and inspired by [19], the third variant we propose imposes a Gaussian prior on $\mathbf{p}_u$ and $\mathbf{p}_v$, and minimizes the normalized squared distance between the expectation of $\mathbf{p}_u$ and the ground-truth c_u (respectively, $\mathbf{p}_v$ and c_v). In more detail, let $\mu_u = \sum_{c \in C} \mathbf{p}_u[c] * c$ and $\sigma_u^2 = \sum_{c \in C} \mathbf{p}_u[c] * (c - \mu_u)^2$ (and similarly for μ_v and σ_v^2). Then, Eq. (5) is replaced by:

$$\mathcal{L}^{reg}_{drloc} = \sum_{x \in B} \mathbb{E}_{(\mathbf{e}_{i,j}, \mathbf{e}_{p,h}) \sim G_x} \left[\frac{(c_u - \mu_u)^2}{\sigma_u^2} + \alpha log(\sigma_u) + \frac{(c_v - \mu_v)^2}{\sigma_v^2} + \alpha log(\sigma_v) \right], \tag{6}$$

where the terms $log(\sigma_u)$ and $log(\sigma_v)$ are used for variance regularization [19].

The last variant we propose is based on a "very-dense" localization loss, where $\mathcal{L}_{drloc}$ is computed *for every transformer block* of VT. Specifically, let G_x^l be the $k_l \times k_l$ grid of token embeddings output by the l-th block of VT, and let L be the total number of these blocks. Then, Eq. (2) is replaced by:

$$\mathcal{L}^{all}_{drloc} = \sum_{x \in B} \sum_{l=1}^{L} \mathbb{E}_{(\mathbf{e}_{i,j}, \mathbf{e}_{p,h}) \sim G_x^l} [|(t_u^l, t_v^l)^T - (d_u^l, d_v^l)^T|_1], \tag{7}$$

where $(t_u^l, t_v^l)^T$ and $(d_u^l, d_v^l)^T$ are, respectively, the target and the prediction offsets computed at block l using the randomly sampled pair $(\mathbf{e}_{i,j}, \mathbf{e}_{p,h}) \in G_x^l$. For each block, we use a block-specific MLP f^l to compute $(d_u^l, d_v^l)^T$. Note that, using Eq. (7), the initial layers of VT receive more "signal", because each block l accumulates the gradients produced by all the blocks $l' \geq l$.

All the proposed variants but the last ($\mathcal{L}^{all}_{drloc}$) are very computationally efficient, because they involve only one forward and one backward pass per image, and m forward passes through f. In Sec. 5.1 we use the final VT accuracy to empirically compare to each other the proposed localization losses.

5 Experiments

All the experiments presented in this section are based on image classification tasks, while in the Appendix we also show object detection, instance segmentation and semantic segmentation tasks.

We use 11 different datasets: ImageNet-100 (IN-100) [56, 60], which is a subset of 100 classes of ImageNet; CIFAR-10 [34], CIFAR-100 [34], Oxford Flowers102 [45], and SVHN [44], which are four widely used vision datasets; and the six datasets of DomainNet [48], a benchmark commonly used for domain adaptation tasks. We chose the latter because of the large domain-shift between some of its datasets and ImageNet, which mekes the fine-tuning experiments non-trivial. Tab. 1 (a) shows the number of samples for each of these 11 datasets.

We used, when available, the official VT code (for T2T [66] and Swin [38]) and a publicly available implementation of CvT [62][1]. In the fine-tuning experiments (Sec. 5.3), we use only T2T and Swin because of the lack of publicly available ImageNet pre-trained CvT networks. For each of the three baselines, we chose a model of comparable size to ResNet-50 (25M parameters): see Tab. 2 (b) for more details. When we plug our loss on one of these baselines, we follow Sec. 4, *keeping unchanged* the VT architecture apart from our localization MLP (f). Moreover, in all the experiments, we train the baselines, both with and without our localization loss, using the same data-augmentation protocol for all the models, and we use the VT-specific hyper-parameter configuration suggested by the authors of each VT. We train each model using 8 Nvidia V100 32GB GPUs.

5.1 Ablation study

In Tab. 1 (b) we compare the loss variants presented in Sec. 4.2. For these experiments, we use IN-100, we train all the models for 100 epochs, and we show the top-1 classification accuracy on the test set. For all the variants, the baseline model is Swin [38] (row (A) of Tab. 1 (b)).

When we plug $\mathcal{L}_{drloc}$ on top of Swin (Sec. 4), the final accuracy increases by 1.26 points (B). All the other dense localization loss variants underperform $\mathcal{L}_{drloc}$ (C-F). A bit surprisingly, the very-dense localization loss $\mathcal{L}^{all}_{drloc}$ is significantly outperformed by the much simpler (and computationally more efficient) $\mathcal{L}_{drloc}$. Moreover, $\mathcal{L}^{all}_{drloc}$ is the only variant which underperforms the baseline. We presume that this is due to the fact that most of the Swin intermediate blocks have resolution grids G^l_x finer than the last grid G^L_x ($l < L$, $k_l > k_L$, Sec. 4.2), and this makes the localization task harder, slowing down the convergence of f^l, and likely providing noisy gradients to the VT (see Sec. 4.1). In the rest of this paper and in all the other experiments, we always use $\mathcal{L}_{drloc}$ as the relative localization loss.

Finally, we analyze the impact of different values of m (the total number of embedding pairs used per image, see Sec. 4). Since we use the same grid resolution for all the VTs (i.e., 7×7, Sec. 3), also the number of embeddings per image is the same for all the VTs ($k^2 = 49$). Hence, following the results of Tab. 2 (a), obtained with CIFAR-100 and Swin, we use $m = 64$ for all the VTs and all the datasets.

Table 1: (a) The size of the datasets used in our empirical analysis. (b) IN-100, 100 epoch training: a comparison between the different variants of our proposed loss.

(a)

	Dataset	Train size	Test size	Classes
	ImageNet-100 [56]	126,689	5,000	100
	CIFAR-10 [35]	50,000	10,000	10
	CIFAR-100 [35]	50,000	10,000	100
	Oxford Flowers102 [45]	2,040	6,149	102
	SVHN [44]	73,257	26,032	10
DomainNet [48]	ClipArt	33,525	14,604	345
DomainNet [48]	Infograph	36,023	15,582	345
DomainNet [48]	Painting	50,416	21,850	345
DomainNet [48]	Quickdraw	120,750	51,750	345
DomainNet [48]	Real	120,906	52,041	345
DomainNet [48]	Sketch	48,212	20,916	345

(b)

	Model	Top-1 Acc.
A:	Swin-T [38]	82.76
B:	A + $\mathcal{L}_{drloc}$	84.02 (+1.26)
C:	A + $\mathcal{L}^{*}_{drloc}$	83.14 (+0.38)
D:	A + $\mathcal{L}^{ce}_{drloc}$	83.86 (+1.10)
E:	A + $\mathcal{L}^{reg}_{drloc}$	83.24 (+0.48)
F:	A + $\mathcal{L}^{all}_{drloc}$	81.88 (-0.88)

[1] https://github.com/lucidrains/vit-pytorch

Table 2: (a) CIFAR-100, 100 training epochs: an analysis of the impact of the number of pair samples (m) in L_{drloc}. (b) Accuracy results on IN-100 with different number of training epochs.

(a)

Model	Top-1 Acc.
A: Swin-T [38]	53.28
B: A + $\mathcal{L}_{drloc}$, m=32	63.70
C: A + $\mathcal{L}_{drloc}$, m=64	**66.23**
D: A + $\mathcal{L}_{drloc}$, m=128	65.16
E: A + $\mathcal{L}_{drloc}$, m=256	64.87

(b)

	Model	#Param (M)	Top-1 Acc. 100 epochs	Top-1 Acc. 300 epochs
CvT	CvT-13	20	85.62	90.16
	CvT-13+$\mathcal{L}_{drloc}$	20	**85.98** (+0.36)	**90.28** (+0.12)
Swin	Swin-T	29	82.76	89.68
	Swin-T+$\mathcal{L}_{drloc}$	29	**84.02** (+1.26)	**90.32** (+0.64)
T2T	T2T-ViT-14	22	82.74	87.76
	T2T-ViT-14+$\mathcal{L}_{drloc}$	22	**83.90** (+1.16)	**88.16** (+0.4)

5.2 Training from scratch

In this section, we analyze the performance of both the VT baselines and our regularization loss using small-medium size datasets and different number of training epochs. The goal is to simulate a scenario with limited computational resources and/or limited training data.

We start by analyzing the impact of the number of training epochs on IN-100. Tab. 2 (b) shows that, using $\mathcal{L}_{drloc}$, *all the tested VTs* show an accuracy improvement, and this boost is larger with fewer epochs. As expected, our loss acts as a regularizer, whose effects are more pronounced in a shorter training regime. We believe this result is particularly significant considering the larger computational times which are necessary to train typical VTs with respect to ResNets.

In Tab. 3, we use all the other datasets and we train from scratch with 100 epochs. First, we note that the accuracy of the VT baselines varies a lot depending on the dataset (which is expected), but also depending on the specific VT architecture. As a reference, when these VTs are trained on ImageNet-1K (for 300 epochs), the differences of their respective top-1 accuracy is much smaller: Swin-T, 81.3 [38]; T2T-ViT-14, 81.5 [66]; CvT-13, 81.6 [62]. Conversely, Tab. 3 shows that, for instance, the accuracy difference between CvT and Swin is about 45-46 points in Quickdraw and Sketch, 30 points on CIFAR-10, and about 20 points on many other datasets. Analogously, the difference between CvT and T2T is between 20 and 25 points in Sketch, Painting, Flowers102, and quite significant in the other datasets. This comparison shows that CvT is usually much more robust in a small training-set regime with respect to the other two VTs. We believe that these results are interesting especially for those tasks in which fine-tuning a model pre-trained on large datasets is not possible. This is the case, for instance, when there is a large domain-shift with respect to the application dataset (e.g., medical images, etc.) or when the VT architecture should be drastically modified and adapted to the specific task (e.g., processing 3D data, etc.). In these scenarios, choosing an architecture which can quickly learn from small-medium size datasets may be crucial.

In Tab. 3, we also show the accuracy of these three VTs when training is done using $\mathcal{L}_{drloc}$ as a regularizer. Similarly to the IN-100 results, also in this case our loss *improves the accuracy of all the tested VTs in all the datasets*. Most of the time, this improvement is quite significant (e.g., almost 4 points on SVHN with CvT), and sometimes dramatic (e.g., more than 45 points on Quickdraw with Swin). These results show that a self-supervised side task can provide a significant "signal" to the VT when the training set is limited, and, specifically, that our loss can be very effective in boosting the accuracy of a VT trained from scratch in this scenario.

5.3 Fine-tuning

In this section, we analyze a typical fine-tuning scenario, in which a model is pre-trained on a big dataset (e.g., ImageNet), and then fine-tuned on the target domain. Specifically, in *all* the experiments, we use VT models pre-trained by the corresponding VT authors on ImageNet-1K *without* our localization loss. The difference between the baselines and ours concerns *only* the fine-tuning stage, which is done in the standard way for the former and using our $\mathcal{L}_{drloc}$ regularizer for the latter. Starting from standard pre-trained models and using our loss only in the fine-tuning stage, emphasises the easy to use of our proposal in pratical scenarios, in which fine-tuning can be

Table 3: Top-1 accuracy of the VTs, trained from scratch on different datasets (100 epochs).

		CIFAR-10	CIFAR-100	Flowers102	SVHN	ClipArt	Infograph	Painting	Quickdraw	Real	Sketch
CvT	CvT-13	89.02	73.50	54.29	91.47	60.34	19.39	54.79	70.10	76.33	56.98
	CvT-13+$\mathcal{L}_{drloc}$	**90.30** (+1.28)	**74.51** (+1.01)	**56.29** (+2.00)	**95.36** (+3.89)	**60.64** (+0.30)	**20.05** (+0.67)	**55.26** (+0.47)	**70.36** (+0.26)	**77.05** (+0.68)	**57.56** (+0.58)
Swin	Swin-T	59.47	53.28	34.51	71.60	38.05	8.20	35.92	24.08	73.47	11.97
	Swin-T+$\mathcal{L}_{drloc}$	**83.89** (+24.42)	**66.23** (+12.95)	**39.37** (+4.86)	**94.23** (+22.63)	**47.47** (+9.42)	**10.16** (+1.96)	**41.86** (+5.94)	**69.41** (+45.33)	**75.59** (+2.12)	**38.55** (+26.58)
T2T	T2T-ViT-14	84.19	65.16	31.73	95.36	43.55	6.89	34.24	69.83	73.93	31.51
	T2T-ViT-14+$\mathcal{L}_{drloc}$	**87.56** (+3.37)	**68.03** (+2.87)	**34.35** (+2.62)	**96.49** (+1.13)	**52.36** (+8.81)	**9.51** (+2.62)	**42.78** (+8.54)	**70.16** (+0.33)	**74.63** (+0.70)	**51.95** (+20.44)

Table 4: VTs pre-trained on ImageNet-1K and then fine-tuned (top-1 accuracy, 100 epochs).

		CIFAR-10	CIFAR-100	Flowers102	SVHN	ClipArt	Infograph	Painting	Quickdraw	Real	Sketch
Swin	Swin-T	97.95	88.22	98.03	96.10	73.51	41.07	72.99	75.81	85.48	72.37
	Swin-T+$\mathcal{L}_{drloc}$	**98.37** (+0.42)	**88.40** (+0.18)	**98.21** (+0.18)	**97.87** (+1.77)	**79.51** (+6.00)	**46.10** (+5.03)	**73.28** (+0.29)	**76.01** (+0.20)	**85.61** (+0.13)	**72.86** (+0.49)
T2T	T2T-ViT-14	98.37	87.33	97.98	97.03	74.59	38.53	72.29	74.16	84.56	72.18
	T2T-ViT-14+$\mathcal{L}_{drloc}$	**98.52** (+0.15)	**87.65** (+0.32)	**98.08** (+0.20)	**98.20** (+1.17)	**78.22** (+0.37)	**45.69** (+7.16)	**72.42** (+0.13)	**74.27** (+0.11)	**84.57** (+0.01)	**72.29** (+0.11)

done without re-training the model on ImageNet. As mentioned in Sec. 5, in this analysis we do not include CvT because of the lack of publicly available ImageNet-1K pre-trained models for this architecture.

The results are presented in Tab. 4. Differently from the results shown in Sec. 5.2, the accuracy difference between T2T and Swin is much less pronounced, and the latter outperforms the former in most of the datasets. Moreover, analogously to all the other experiments, also in this case, using $\mathcal{L}_{drloc}$ leads to an accuracy improvement *with all the tested VTs and in all the datasets*. For instance, on Infograph, Swin with $\mathcal{L}_{drloc}$ improves of more than 5 points, and T2T more than 7 points.

6 Conclusion

In this paper, we have empirically analyzed different VTs, showing that their performance largely varies when trained from scratch with small-medium size datasets, and that CvT is usually much more effective in generalizing with less data. Moreover, we proposed a self-supervised side-task to regularize VT training. Our localization task, inspired by [12], is densely defined for a random subset of final-token embedding pairs, and it encourages the VT to learn spatial information.

In our extensive empirical analysis, with 11 datasets, different training scenarios and three VTs, our dense localization loss *has always improved the corresponding baseline accuracy*, usually by a significant margin, and sometimes dramatically (up to +45 points). We believe that this shows that our proposal is an easy-to-reproduce, yet very effective tool to boost the performance of VTs, especially in training regimes with a limited amount of data/training time. It also paves the way to investigating other forms of self-supervised/multi-task learning which are specific for VTs, and can help VT training without resorting to the use of huge annotated datasets.

Limitations. A deeper analysis on why fine-grained embedding grids have a negative impact on our localization loss (Sec. 4.1 and 5.1) was left as a future work. Moreover, in our analysis, we focused on VT models of approximately the same size as a ResNet-50. We have not considered

bigger Swin/T2T/CvT models because training and fine-tuning very large networks on 11 datasets is too computationally demanding. For the same reason, we have not tested ViT. However, since the goal of this paper is investigating the VT behaviour with medium-small size datasets, most likely these big models are not the best choice in a training scenario with scarcity of data, as witnessed by the fact that ViT underperforms similar-capacity CNNs when trained on ImageNet-1K [18].

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020.

[2] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv:2105.04906*, 2021.

[3] Miguel A Bautista, Artsiom Sanakoyeu, Ekaterina Tikhoncheva, and Bjorn Ommer. CliqueCNN: deep unsupervised exemplar learning. In *NeurIPS*, 2016.

[4] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *CVPR*, 2018.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[6] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

[8] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[10] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv:2011.10566*, 2020.

[11] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv:2104.02057*, 2021.

[12] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

[13] MMCV Contributors. Openmmlab foundational library for computer vision research, 2020.

[14] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv:2009.09796*, 2020.

[15] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. UP-DETR: unsupervised pre-training for object detection with transformers. *arXiv:2011.09094*, 2020.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL,*, 2019.

[17] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[19] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *CVPR*, 2019.

[20] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. *arXiv:2104.14548*, 2021.

[21] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *ICML*, 2021.

[22] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. SCAN: learning to classify images without labels. In *ECCV*, 2020.

[23] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

[24] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv:2006.07733*, 2020.

[25] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *CVPR*, 2020.

[26] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[28] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

[29] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019.

[30] Tianyu Hua, Wenxiao Wang, Zihui Xue, Yue Wang, Sucheng Ren, and Hang Zhao. On feature decorrelation in self-supervised learning. *arXiv:2105.00470*, 2021.

[31] Drew A. Hudson and C. Lawrence Zitnick. Generative Adversarial Transformers. *arXiv:2103.01209*, 2021.

[32] Xu Ji, João F. Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, 2019.

[33] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. TransGAN: Two transformers can make one strong GAN. *arXiv:2102.07074*, 2021.

[34] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[35] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[36] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. LocalViT: Bringing locality to vision transformers. *arXiv:2104.05707*, 2021.

[37] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[38] 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. *arXiv:2103.14030*, 2021.

[39] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017.

[40] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixé, and Christoph Feichtenhofer. TrackFormer: Multi-object tracking with transformers. *arXiv:2101.02702*, 2021.

[41] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, 2013.

[42] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[43] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020.

[44] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[45] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

[46] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[47] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *ICCV*, 2017.

[48] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *CVPR*, 2019.

[49] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[50] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:140:1–140:67, 2020.

[51] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[52] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL-HLT*, 2018.

[53] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. *arXiv:2105.05633*, 2021.

[54] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. VideoBERT: A joint model for video and language representation learning. In *ICCV*, 2019.

[55] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. *arXiv:2011.12450*, 2020.

[56] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, 2020.

[57] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv:2012.12877*, 2020.

[58] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

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

[60] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, 2020.

[61] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[62] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing convolutions to vision transformers. *arXiv:2103.15808*, 2021.

[63] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018.

[64] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. *arXiv:2104.06399*, 2021.

[65] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. *arXiv:2103.11816*, 2021.

[66] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token ViT: Training vision transformers from scratch on ImageNet. *arXiv:2101.11986*, 2021.

[67] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv:2103.03230*, 2021.

[68] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019.

[69] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.

[70] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *ICCV*, 2019.

A PyTorch-like pseudocode

In Figure 2, we show a PyTorch-like pseudocode of our dense relative localization task with the associated $\mathcal{L}_{drloc}$ loss. In the rest of this Appendix, we show additional results and a different loss variant.

```
# n        : batch size
# m        : number of pairs
# k X k    : resolution of the embedding grid
# D        : dimension of each token embedding
# x        : a tensor of n embedding grids, shape=[n, D, k, k]

def position_sampling(k, m, n):
    pos_1 = torch.randint(k, size=(n, m, 2))
    pos_2 = torch.randint(k, size=(n, m, 2))
    return pos_1, pos_2

def collect_samples(x, pos, n):
    return torch.stack([x[i, :, pos[i][:,0], pos[i][:,1]] for i in range(n)], dim=0)

def dense_relative_localization_loss(x):
    n, D, k, k = x.size()
    pos_1, pos_2 = position_sampling(k, m, n)

    deltaxy = abs((pos_1-pos_2).float()) # [n, m, 2]
    deltaxy /= k

    pts_1 = collect_samples(x, pos_1, n).transpose(1,2) # [n, m, D]
    pts_2 = collect_samples(x, pos_2, n).transpose(1,2) # [n, m, D]
    predxy = MLP(torch.cat([pts_1, pts_2], dim=2))
    return L1Loss(predxy, deltaxy)
```

Figure 2: A PyTorch-like pseudocode of our dense relative localization task and the corresponding $\mathcal{L}_{drloc}$ loss.

B Additional results

In this section, we provide additional fine-tuning experiments using tasks different from classification (i.e., object detection, instance segmentation and semantic segmentation). Moreover, we use a training protocol different form that used in Sec. 5.3. Specifically, the fine-tuning stage is standard (*without our loss*), while in the pre-training stage we either use the standard cross-entropy (only), or we pre-train the VT jointly using the cross-entropy and $\mathcal{L}_{drloc}$. The pre-training dataset is IN-100, and the VT baseline is Swin. Pre-training is performed for 300 epochs, thus the baseline model is fine-tuned starting from the Swin-T model corresponding to Tab. 2 (b) (final accuracy : 89.68), while Swin-T + $\mathcal{L}_{drloc}$ refers to the model trained with our loss in the same table (final accuracy: 90.32).

The goal of these experiments is to show that the image representation obtained using $\mathcal{L}_{drloc}$ for pre-training, can be usefully transferred to other tasks without modifying the task-specific architecture or the fine-tuning protocol. In these experiments, we adopt the framework proposed in [38], where a pre-trained Swin VT is used as the backbone for detection and segmentation tasks. In fact, note that Swin is based on a hierarchy of embedding grids, which can be used by the specific object detection/image segmentation architectures as they were convolutional feature maps [38].

B.1 Object detection and instance segmentation

Setup. We strictly follow the experimental settings used in Swin [38]. Specifically, we use COCO 2017 [37], which contains 118K training, 5K validation and 20K test-dev images. We use two popular object detection architectures: Cascade Mask R-CNN [4] and Mask R-CNN [26], in which the backbone is replaced with the IN-100 pre-trained Swin model [38]. Moreover, we use the standard mmcv [13] framework to train and benchmark the models. We adopt multi-scale training [5, 55] (i.e., we resize the input image such that the shortest side is between 480 and 800 pixels, while the longest

side is at most 1333 pixels), the AdamW [39] optimizer (initial learning rate 0.0001, weight decay 0.05, and batch size 16), and a 3x schedule (36 epochs with the learning rate decayed by $10\times$ at epochs 27 and 33).

Results. Tab. 5 shows that Swin-T, pre-trained with our $\mathcal{L}_{drloc}$ loss, achieves both a higher detection and a higher instance segmentation accuracy with respect to the baselines. Specifically, with both Mask RCNN and Cascade Mask RCNN, and with all the detection/segmentation metrics, our pre-trained model *always* outperforms the baselines.

Table 5: Results on the COCO object detection and instance segmentation tasks. AP^{box}_{x} and AP^{mask}_{x} are the standard object detection and segmentation Average Precision metrics, respectively [37].

Architecture	**Pre-trained backbone**	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
Mask RCNN	Swin-T [38]	41.8	60.3	45.1	36.7	57.4	39.4
	Swin-T + $\mathcal{L}_{drloc}$	**42.7** (+0.9)	**61.3** (+1.0)	**45.9** (+0.8)	**37.2** (+1.0)	**58.4** (+1.0)	**40.0** (+0.6)
Cascade Mask RCNN	Swin-T [38]	36.0	58.2	38.6	33.8	55.2	35.9
	Swin-T + $\mathcal{L}_{drloc}$	**37.2** (+1.2)	**59.4** (+1.2)	**40.3** (+1.7)	**34.5** (+0.7)	**56.2** (+1.0)	**36.6** (+0.7)

B.2 Semantic segmentation

Setup. We again follow the experimental settings adopted in Swin [38]. Specifically, for the semantic segmentation experiments, we use the ADE20K dataset [68], which is composed of 150 semantic categories, and contains 20K training, 2K validation and 3K testing images. Following [38], we use the popular UperNet [63] architecture with a Swin backbone pre-trained on IN-100. We use the implementation released by mmcv [13] to train and benchmark the models.

When fine-tuning, we used the AdamW [39] optimizer with an initial learning rate of 6×10^{-5}, a weight decay of 0.01, a scheduler with linear learning-rate decay, and a linear warmup of 1,500 iterations. We fine-tuned all the models on 8 Nvidia V100 32GB GPUs with 2 images per GPU for 160K iterations. We adopt the default data augumentation techniques used for segmentation, namely random horizontal flipping, random re-scaling with a [0.5, 2.0] ratio range and random photometric distortion. Stochastic depth with ratio 0.2 is applied for all models, which are trained with an input of 512×512 pixels. At inference time, we use a multi-scale testing, with image resolutions which are $\{0.5, 0.75, 1.0, 1.25, 1.5, 1.75\}\times$ of the training resolution.

Results. The results reported in Tab. 6 shows that the models pre-trained with the proposed loss *always* outperform the baselines with respect to all the segmentation metrics.

Table 6: Results on the semantic segmentation task on the ADE20K validation set. mIoU and mAcc refer to mean Intersection over Union and mean class Accuracy, respectively. The base architecture is UperNet [63].

Pre-trained backbone	**mIoU**	**mAcc**
Swin-T [38]	36.9	47.8
Swin-T + $\mathcal{L}_{drloc}$	**37.8** (+0.9)	**48.7** (+0.9)

C An additional loss variant

In this section, we describe an additional loss function which can be used with our dense relative localization task, by slightly changing the sampling procedure. Specifically, in order to backpropagate the loss gradients with respect to the largest possible number of embeddings in each image, we exhaustively (and deterministically) use all the embeddings at least once, while randomly sampling the "partner" token embedding. Using the notation of Sec. 4, this corresponds to iteratively select

$\mathbf{e}_{i,j}$ (for all $1 \leq i, j, \leq k$) and then randomly sampling $\mathbf{e}_{p,h}$:

$$\mathcal{L}_{drloc}^{exh} = \sum_{x \in B} \sum_{\mathbf{e}_{i,j} \in G_x} \mathbb{E}_{\mathbf{e}_{p,h} \sim G_x}[|(t_u, t_v)^T - (d_u, d_v)^T|_1]. \tag{8}$$

$\mathcal{L}_{drloc}^{exh}$ is more coherent with the ELECTRA pretext task [12], which inspired our work, because it is defined for each $\mathbf{e}_{i,j} \in G_x$. In Tab. 7, we use CIFAR-100 and we compare $\mathcal{L}_{drloc}^{exh}$ (row F) with $\mathcal{L}_{drloc}$ with different numbers of pairs (m). Note that, in case of $\mathcal{L}_{drloc}^{exh}$, $m = k^2 = 49$. Despite $\mathcal{L}_{drloc}^{exh}$ is much better than the baseline (Swin), it is outperformed by $\mathcal{L}_{drloc}$ with $m = 64$, which most likely shows that using a larger number of (random) pairs is better than uniformly scattering the gradients through the whole embedding grid.

Table 7: Comparing $\mathcal{L}_{drloc}^{exh}$ with $\mathcal{L}_{drloc}$ on the CIFAR-100 dataset.

Model	**Top-1 Acc.**
A: Swin-T [38]	53.28
B: A + $\mathcal{L}_{drloc}$, m=32	63.70
C: A + $\mathcal{L}_{drloc}$, m=64	**66.23**
D: A + $\mathcal{L}_{drloc}$, m=128	65.16
E: A + $\mathcal{L}_{drloc}$, m=256	64.87
F: A + $\mathcal{L}_{drloc}^{exh}$, m=49	64.40

D Implementation details

Our localization MLP is a simple feed-forward network composed of three fully connected layers. The first layer projects the concatenation of the two token embeddings $\mathbf{e}_{i,j}$ and $\mathbf{e}_{p,h}$ into a 512-dimensional vector and then it applies a `Relu` activation. Next, we use a linear layer of dimension 512 followed by a `Relu` activation. Finally, we use a linear layer dedicated to the prediction, which depends on the specific loss variant, see Sec. 4 and 4.2. For instance, in $\mathcal{L}_{drloc}$, the last layer is composed of two neurons which predict d_u and d_v. The details of the MLP head are shown in Table 8.

Table 8: The details of the localization MLP head. D is the dimension of a token embedding. The number of outputs o and the final nonlinearity (if used) depend on the specific loss. In $\mathcal{L}_{drloc}$, $\mathcal{L}_{drloc}^{*}$, $\mathcal{L}_{drloc}^{all}$ and $\mathcal{L}_{drloc}^{exh}$, we use $o = 2$ without any nonlinearity. Converesely, in both $\mathcal{L}_{drloc}^{ce}$ and $\mathcal{L}_{drloc}^{reg}$, the last layer is split in two branches of $2k + 1$ neurons each, and, on each branch, we separately apply a `SoftMax` layer.

Layer	**Activation**	**Output dimension**
Input	-	D * 2
Linear	ReLU	512
Linear	ReLU	512
Linear	- / SoftMax	o

In our experiments, we used the officially released framework of Swin [38][2], which also provides all the necessary code to train and test VT networks (including the object detection and segmentation tasks of this Appendix). For a fair comparison, we use the official code of T2T-ViT [66][3] and a publicly released code of CvT [62][4] and we insert them in the training framework released by the authors of Swin. At submission time of this paper, the official code of CvT [62] is not publicly available.

When we train the networks from scratch, we use the AdamW [39] optimizer for 100 epochs with a cosine decay learning-rate scheduler and 20 epochs of linear warm-up. We use a batch size of 1024,

[2] `https://github.com/microsoft/Swin-Transformer`
[3] `https://github.com/yitu-opensource/T2T-ViT`
[4] `https://github.com/lucidrains/vit-pytorch`

an initial learning rate of 0.001, and a weight decay of 0.05. When we fine-tune the networks, we use the AdamW [39] optimizer for 100 epochs with a cosine decay learning-rate scheduler and 10 epochs of linear warm-up. We use a batch size of 1024, an initial learning rate of 0.0005, and a weight decay of 0.05.

In all the experiments reported in the main paper, the images of all the datasets are resized to the same fixed resolution (224×224).

The source code to reproduce all our experiments will be released upon the acceptance of this paper.

E Dataset licensing details

CIFAR-10, CIFAR-100 are released to the public with a non-commercial research and/or educational use[5]. Oxford flower102 is released to the public with an unknown license through its website[6], and we assume a non-commercial research and/or educational use. ImageNet annotations have a non-commercial research and educational license[7]. SVHN is released with a non-commercial use only license[8]. The DomainNet dataset is released under a fair use license[9].

The use of COCO 2017 images should abide by the Flickr Terms of Use, while the annotations in this dataset along with this website belong to the COCO Consortium and are licensed under a Creative Commons Attribution 4.0 License[10].

ADE20K images are shared for non-commercial research and/or educational use, while the annotations are licensed under a Creative Commons BSD-3 License Agreement[11].

[5] https://www.cs.toronto.edu/~kriz/cifar.html
[6] https://www.robots.ox.ac.uk/~vgg/data/flowers/102/
[7] https://image-net.org/download
[8] http://ufldl.stanford.edu/housenumbers/
[9] http://ai.bu.edu/M3SDA/
[10] https://cocodataset.org/#termsofuse
[11] https://groups.csail.mit.edu/vision/datasets/ADE20K/terms/