SPATIAL ENTROPY AS AN INDUCTIVE BIAS FOR VISION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work showed that the attention maps of Vision Transformers (VTs), when trained with self-supervision, can contain a semantic segmentation structure which does not spontaneously emerge when training is supervised. In this paper, we *explicitly* encourage the emergence of this spatial clustering as a form of training regularization, this way including a self-supervised *pretext task* into the standard supervised learning. In more detail, we exploit the assumption that, in a given image, objects usually correspond to few connected regions, and we propose a spatial formulation of the information entropy to quantify this *object-based inductive bias*. By minimizing the proposed spatial entropy, we include an additional self-supervised signal during training. Using extensive experiments, we show that the proposed regularization is beneficial with different training scenarios, datasets, downstream tasks and VT architectures. The code will be available upon acceptance.

1 INTRODUCTION

There is a growing interest in the computer vision community on Vision Transformers (VTs), as a computational paradigm alternative to standard Convolutional Neural Networks (CNNs). VTs are inspired by the Transformer network (Vaswani et al., 2017), which is the *de facto* standard in Natural Language Processing (NLP) (Devlin et al., 2019; Radford & Narasimhan, 2018) and it is based on multi-head attention layers transforming the input *tokens* (e.g., language words) into a set of final embedding tokens. Dosovitskiy et al. (2021) recently proposed an analogous processing paradigm, where word tokens are replaced by image patches, and self-attention layers are used to model global pairwise dependencies over all the input tokens. As a consequence, differently from CNNs, where the convolutional kernels have a spatially limited receptive field, ViT (Dosovitskiy et al., 2021) has a dynamic receptive field, which is given by its attention maps (Naseer et al., 2021). However, ViT heavily relies on huge training datasets (e.g., JFT-300M (Dosovitskiy et al., 2021), a proprietary dataset of 303 million images), and underperforms CNNs when trained on ImageNet-1K ($\sim$ 1.3 million images (Russakovsky et al., 2015)) or using smaller datasets (Dosovitskiy et al., 2021; Raghu et al., 2021). To mitigate the need for a huge quantity of training data, a recent line of research is exploring the possibility of reintroducing typical CNN mechanisms in VTs (Yuan et al., 2021b; Liu et al., 2021b; Wu et al., 2021; Yuan et al., 2021a; Xu et al., 2021; Li et al., 2021b; Hudson & Zitnick, 2021). The main idea behind these works is that convolutional layers, mixed with the VT self-attention layers, help to embed a *local inductive bias* in the VT architecture, i.e., to encourage the network to focus on local properties of the image domain. In this paper, we follow an orthogonal (and relatively simpler) direction: rather than changing the VT architecture, we propose to include a local inductive bias using an additional *pretext task* during training which can be easily plugged into existing VTs without significant structural modifications. Specifically, we maximize the probability of producing attention maps which focus on local regions (of *variable* size), based on the idea that,

(a) (b) (c)

Figure 1: ViT attention maps obtained using the `[CLS]` token query and thresholded to keep 60% of the mass (Caron et al., 2021). (a) Standard supervised learning. (b) DINO. (c) Training using SAR.

most of the time, an object is represented by one or very few spatially connected regions in the input image. This pretext task exploits a locality principle, characteristic of the natural images, and extracts additional (self-supervised) information from images without the need of architectural changes.

Our work is inspired by the findings presented in Caron et al. (2021); Bao et al. (2021); Naseer et al. (2021), in which the authors show that VTs, trained using self-supervision (Caron et al., 2021; Bao et al., 2021) or shape-distillation (Naseer et al., 2021), can spontaneously develop attention maps with a semantic segmentation structure. For instance, Caron et al. (2021) show that the last-layer attention maps of ViT, when this is trained with their self-supervised DINO algorithm, can be thresholded and used to segment the most important foreground objects of the input image *without any pixel-level annotation* during training (see Fig. 1 (b)). Similar findings are shown in Bao et al. (2021); Naseer et al. (2021). Interestingly, however, Caron et al. (2021) show that the same ViT architectures, when trained with supervised methods, produce much more spatially disordered attention maps (Fig. 1 (a)). This is confirmed by Naseer et al. (2021), who observed that the attention maps of ViT, trained with supervised protocols, have a widely spread structure over the whole image. The reason why "blob"-like attention maps *spontaneously* emerge, when VTs are trained with some algorithms but not with others, is still unclear. However, in this paper we build on top of these findings and we propose a *spatial entropy* loss function which *explicitly* encourages the emergence of locally structured attention maps (Fig. 1 (c)), independently of the main algorithm used for training. Importantly, our goal is *not* to extract segmentation-like structures from the attention maps. Instead, we use the proposed spatial entropy loss to introduce an *object-based local prior* in VTs: since real life objects usually correspond to one or very few connected image regions, then also the corresponding attention maps of a VT head should focus most of their largest values on spatially clustered regions. The possible discrepancy between this inductive bias (the semantic content in a given image has a low spatial entropy) and the actual spatial entropy measured in each VT head, provides a *self-supervised* signal which is independent of the possible image label and it alleviates the need for huge supervised training datasets, without changing the VT architecture.

The second contribution of this paper is based on the empirical results recently presented by Raghu et al. (2021), who showed that VTs are more influenced by the skip connections than CNNs and, specifically, that in the last blocks of ViT, the *patch tokens* (see §3) representations are mostly influenced by the skip connection path. This means that, in the last blocks of ViT, the self-attention layers have a relatively small influence on the final token embeddings. Since our spatial entropy is measured on the last-block attention maps, we propose to remove the skip connections in the last layer (only). We empirically show that this minor architectural change is beneficial for ViT, both when used jointly with our spatial entropy loss, and when used with a standard training procedure.

Our regularization method, which we call SAR (Spatial Attention-based Regularization), can be easily plugged into existing VTs without drastic architectural changes and it can be applied to different scenarios, jointly with a main-task loss function. For instance, when used in a supervised classification task, the main loss is the (standard) cross entropy, used jointly with our spatial entropy loss. The goal of SAR is to use the spatial layout of the attention map values as additional unsupervised information which alleviates the need of large supervised data when training a VT. In summary, our main contributions are the followings: (1) We propose a spatial entropy loss which exploits the spatial clustering of the attention maps to extract an additional self-supervised signal during training. (2) We propose to remove the last-block skip connections, empirically showing that this is beneficial for the patch token representations. (3) Using extensive experiments, we show that SAR improves the accuracy of different VT architectures, leading to a very large boost when trained from scratch with small-medium datasets.

2 Related work

Vision Transformers. One of the very first fully-Transformer architectures for computer vision is iGPT (Chen et al., 2020a), in which each image pixel is represented as a token. However, due to the quadratic computational complexity of Transformer networks (Vaswani et al., 2017), iGPT can only operate with very small resolution images. This problem has been largely alleviated by ViT (Dosovitskiy et al., 2021), where the input tokens are $p \times p$ image patches (§3). The success of ViT has inspired several similar Vision Transformer (VT) architectures in different application domains, such as image classification (Dosovitskiy et al., 2021; Touvron et al., 2020; Yuan et al., 2021b; Liu

et al., 2021b; Wu et al., 2021; Yuan et al., 2021a; Li et al., 2021b; Xu et al., 2021; d'Ascoli et al., 2021), object detection (Carion et al., 2020; Zhu et al., 2021; Dai et al., 2021), segmentation (Strudel et al., 2021; Rao et al., 2021), human pose estimation (Zheng et al., 2021), object tracking (Meinhardt et al., 2021), video processing (Neimark et al., 2021; Li et al., 2021a), image generation (Jiang et al., 2021; Hudson & Zitnick, 2021; Ramesh et al., 2022; Chang et al., 2022), point cloud processing (Guo et al., 2021; Zhao et al., 2020), and many others. However, the lack of the typical CNN local inductive biases makes VTs to need more data for training (Dosovitskiy et al., 2021; Raghu et al., 2021). For this reason, many recent works are addressing this problem by proposing hybrid architectures, which reintroduce typical convolutional mechanisms into the VT design (Yuan et al., 2021b; Liu et al., 2021b; Wu et al., 2021; Yuan et al., 2021a; Xu et al., 2021; Li et al., 2021b; d'Ascoli et al., 2021; Hudson & Zitnick, 2021; Li et al., 2021a). In contrast, we propose a different and simpler solution, in which, rather than changing the VT architecture, we introduce a local inductive bias (§1) by means of a pretext task based on the spatial entropy minimization.

Self-supervised learning. Most of the self-supervised approaches with still images impose a semantic consistency between different views of the same image, where the views are obtained with data-augmentation techniques. So far, most of the research in this field has been based on ResNet (He et al., 2016) backbones, and can be roughly grouped in contrastive learning (van den Oord et al., 2018; Hjelm et al., 2019; Chen et al., 2020b; He et al., 2020; Tian et al., 2020; Wang & Isola, 2020; Dwibedi et al., 2021), clustering methods (Bautista et al., 2016; Zhuang et al., 2019; Ji et al., 2019; Caron et al., 2018; Asano et al., 2020; Gansbeke et al., 2020; Caron et al., 2020; 2021), asymmetric networks (Grill et al., 2020; Chen & He, 2021) and feature-decorrelation methods (Ermolov et al., 2021; Zbontar et al., 2021; Bardes et al., 2021; Hua et al., 2021). Recently, many articles have appeared which use VTs for self-supervised learning. For instance, Chen et al. (2021) have empirically tested different representatives of the above categories using VTs, and they also proposed MoCo-v3, a contrastive approach based on MoCo (He et al., 2020) but without the queue of the past-samples. DINO (Caron et al., 2021) is an on-line clustering method which is one of the current state-of-the-art self-supervised approaches using VTs. BEiT (Bao et al., 2021) adopts the typical "masked-word" NLP pretext task (Devlin et al., 2019), but it needs to pre-extract a vocabulary of visual words using the discrete VAE pre-trained in (Ramesh et al., 2021). Other recent works which use a "masked-patch" pretext task are (He et al., 2021; Xie et al., 2021; Wei et al., 2021; Dong et al., 2021; Hua et al., 2022; Chen et al., 2022; Bachmann et al., 2022; El-Nouby et al., 2021; Zhou et al., 2021; Kakogeorgiou et al., 2022).

In this paper, we do not propose a new self-supervised algorithm, but we rather use self-supervision (we extract information from samples without additional manual annotation) to speed-up the convergence in a supervised scenario. In the Appendix, we also show that SAR can be plugged on top of both MoCo-v3 and DINO, boosting the accuracy of both of them. Similarly to this paper, Liu et al. (2021a) propose a VT regularization approach based on predicting the geometric distance between patch tokens. In contrast, we use the largest value connected regions in the VT attention maps to extract additional unsupervised information from images and the two regularization methods can potentially be used jointly. Li et al. (2020) compute the gradients of a ResNet with respect to the image pixels to get an attention (saliency) map. This map is thresholded and used to mask-out the most salient pixels. Minimizing the classification loss on this masked image encourages the attention on the non-masked image to include most of the useful information. Our approach is radically different and much simpler, because we do not need to manually set the thresholding value and we require only one forward and one backward pass per image.

Spatial entropy. There are many definitions of spatial entropy (Razlighi & Kehtarnavaz, 2009; Altieri et al., 2018). For instance, Batty (1974) normalizes the probability of an event occurring in a given zone by the area of that zone, this way accounting for unequal space partitions. In (Tupin et al., 2000), spatial entropy is defined over a Markov Random Field describing the image content, but its computation is very expensive (Razlighi & Kehtarnavaz, 2009). In contrast, our spatial entropy loss can be efficiently computed and it is differentiable, thus it can be easily used as an auxiliary regularization task in existing VTs.

3 Background

Given an input image I, Dosovitskiy et al. (2021) split I in a grid of $K \times K$ non-overlapping patches, and each patch is linearly projected into a (learned) input embedding space. The input of ViT is

this set of $n = K^2$ *patch tokens*, jointly with a special token, called `[CLS]` token, which is used to represent the whole image. Following a standard Transformer network (Vaswani et al., 2017), ViT (Dosovitskiy et al., 2021) transforms these $n + 1$ tokens in corresponding final $n + 1$ token embeddings using a sequence of L Transformer blocks. Each block is composed of LayerNorm (LN), Multiheaded Self Attention (MSA) and MLP layers, plus skip connections. Specifically, if the token embedding sequence at the $(l-1)$-th layer is $\boldsymbol{z}^{l-1} = [\boldsymbol{z}_{CLS}; \boldsymbol{z}_1; ...\boldsymbol{z}_n]$, then:

$$\boldsymbol{z}' = \text{MSA}(\text{LN}(\boldsymbol{z}^{l-1})) + \boldsymbol{z}^{l-1}, \qquad l = 1, \dots, L \tag{1}$$

$$\boldsymbol{z}^{l} = \text{MLP}(\text{LN}(\boldsymbol{z}')) + \boldsymbol{z}', \qquad l = 1, \dots, L \tag{2}$$

where the addition (+) denotes a skip (or "identity") connection, which is used both in the MSA (Eq. 1) and in the MLP (Eq. 2) layer. The MSA layer is composed of H different heads, and in the h-th head ($1 \leq h \leq H$), each token embedding $\boldsymbol{z}_i \in \mathbb{R}^d$ is projected into a query ($\boldsymbol{q}_i^h$), a key ($\boldsymbol{k}_i^h$) and a value ($\boldsymbol{v}_i^h$). Given query (Q^h), key (K^h) and value (V^h) matrices containing the corresponding elements, the h-th self-attention matrix (A^h) is:

$$A^h = \text{softmax}\left(\frac{Q^h (K^h)^T}{\sqrt{d}}\right). \tag{3}$$

Using A^h, each head outputs a weighted sum of the values in V^h. The final MSA layer output is obtained by concatenating all the head outputs and then projecting each token embedding into a d-dimensional space. Finally, the last-layer (L) class token embedding $\boldsymbol{z}_{CLS}^L$ is fed to an MLP head, which computes a posterior distribution over the set of the target classes and the whole network is trained using a standard cross-entropy loss ($\mathcal{L}_{ce}$). Some hybrid VTs (see §2) such as CvT (Wu et al., 2021) and PVT (Wang et al., 2021), progressively subsample the number of patch tokens, leading to a final $k \times k$ patch token grid ($k \leq K$). In the rest of this paper, we generally refer to a spatially arranged grid of final patch token embeddings with a $k \times k$ resolution.

4 METHOD

Generally speaking, an object usually corresponds to one or very few connected regions of a given image (Fig. 1). Our goal is to exploit this natural image inductive bias and penalize those attention maps which do not lead to a spatial clustering of their largest values. Intuitively, if we compare Fig. 1 (a) with Fig. 1 (b), we observe that, in the latter case, the attention maps are more "spatially ordered", i.e. there are *less* and *bigger* "blobs" (obtained after thresholding the map values (Caron et al., 2021)). Since an image is usually composed of a few main objects, each of which most of the times is represented as a connected region of tokens, during training we penalize those attention maps which produce a large number of small blobs. We use this as an auxiliary pretext task which extracts information from images without additional annotation, by exploiting the assumption that spatially close tokens should preferably belong to the same cluster.

4.1 SPATIAL ENTROPY LOSS

For each head of the last Transformer block, we compute a similarity map S^h ($1 \leq h \leq H$, see §3) by comparing the `[CLS]` token query ($\boldsymbol{q}_{CLS}^h$) with all the patch token keys ($\boldsymbol{k}_{x,y}^h$, where $(x, y) \in \{1, ..., k\}^2$):

$$S_{x,y}^h = < \boldsymbol{q}_{CLS}^h, \boldsymbol{k}_{x,y}^h > / \sqrt{d}, \quad (x, y) \in \{1, ..., k\}^2, \tag{4}$$

where $< \boldsymbol{a}, \boldsymbol{b} >$ is the dot product between $\boldsymbol{a}$ and $\boldsymbol{b}$. S^h is extracted from the self-attention map A^h by selecting the `[CLS]` token as the only query and before applying the softmax (see §4.3 for a discussion about this choice). S^h is a $k \times k$ matrix corresponding to the final $k \times k$ spatial grid of patches (§3), and (x, y) corresponds to the "coordinates" of a patch token in this grid.

In order to extract a set of connected components containing the largest values in S^h, we zero-out those elements of S^h which are smaller than the mean value $m = 1/n \sum_{(x,y) \in \{1,...,k\}^2} S_{x,y}^h$:

$$B_{x,y}^h = \text{ReLU}(S_{x,y}^h - m), \quad (x, y) \in \{1, ..., k\}^2, \tag{5}$$

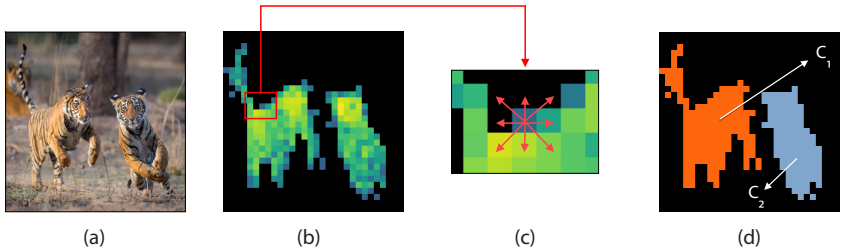

Figure 2: A schematic illustration of the spatial entropy. (a) The original image. (b) The thresholded similarity map B^h (zero values shown in black). (c) The 8-connectivity relation used to group non-zero elements in B^h. (d) The resulting two connected components (C_1 and C_2).

where thresholding using m corresponds to retain half of the total "mass" of Eq. 4. We can now use a standard algorithm (Grana et al., 2010) to extract the connected components from B^h, obtained using an 8-connectivity relation between non-zero elements in B^h (see Fig. 2):

$$C^h = \{C_1, ..., C_{h_r}\} = \text{ConnectedComponents}(B^h). \tag{6}$$

C_j ($1 \leq j \leq h_r$) in C^h is the set of coordinates ($C_j = \{(x_1, y_1), ..., (x_{n_j}, y_{n_j})\}$) of the j-th connected component, whose cardinality (n_j) is variable, and such is the total number of components (h_r). Given C^h, we define the spatial entropy as:

$$\mathcal{H}(S^h) = -\sum_{j=1}^{h_r} P^h(C_j) \log P^h(C_j), \tag{7}$$

$$P^h(C_j) = \frac{1}{|B^h|} \sum_{(x,y) \in C_j} B^h_{x,y}, \tag{8}$$

where $|B^h| = \sum_{(x,y) \in \{1,...,k\}^2} B^h_{x,y}$. Importantly, in Eq. 8, the probability of each region ($P^h(C_j)$) is computed using all its elements, and this makes the difference with respect to a non-spatial entropy which is directly computed over all the elements in S^h, without considering the adjacency relation. Note that the less the number of components h_r or the less uniformly distributed the probability values $P^h(C_1), ...P^h(C_{h_r})$, the lower $\mathcal{H}(S^h)$. Using Eq. 7, the spatial entropy loss is defined as:

$$\mathcal{L}_{se} = \frac{1}{H} \sum_{h=1}^{H} \mathcal{H}(S^h). \tag{9}$$

$\mathcal{L}_{se}$ is used jointly with the main task loss. For instance, in case of supervised training, we use: $\mathcal{L}_{tot} = \mathcal{L}_{ce} + \lambda \mathcal{L}_{se}$, where λ is the weight given to $\mathcal{L}_{se}$.

4.2 Removing the skip connections

Raghu et al. (2021) empirically showed that, *in the last blocks* of ViT, the patch token representations are mostly propagated from the previous layers using the skip connections (§1). We presume this is (partially) due to the fact that only the `[CLS]` token is used as input to the classification MLP head (§3), thus the last-block patch token embeddings are usually neglected. Moreover, Raghu et al. (2021) show that the effective receptive field (Luo et al., 2017) of each block, when computed *after* the MSA skip connections, is much smaller than the effective receptive field computed *before* the MSA skip connections. Both empirical observations lead to the conclusion that the MSA skip connections in the last blocks may be detrimental for the *representation* capacity of the final patch token embeddings. This problem is emphasized when using our spatial entropy loss, since it is computed using the attention maps of the last-block MSA (§4.1). For these reasons, we propose to remove the MSA skip connections in the last block (L). Specifically, in the L-th block, we replace Eq. 1-2 with:

$$\boldsymbol{z}' = \text{MSA}(\text{LN}(\boldsymbol{z}^{L-1})), \tag{10}$$

$$\boldsymbol{z}^L = \text{MLP}(\boldsymbol{z}') + \boldsymbol{z}'. \tag{11}$$

Note that, in addition to removing the MSA skip connections (Eq. 10), we also remove the subsequent LN (Eq. 11), because we empirically observed that this further improves the VT accuracy (see §5.1).

4.3 DISCUSSION

In this section, we discuss and motivate the choices made in §4.1 and §4.2. First, we use S^h, extracted before the softmax (Eq. 3), because, using the softmax, the network can "cheat", by increasing the norm of the vectors $\boldsymbol{q}_{CLS}$ and $\boldsymbol{k}_{x,y}$ ($(x,y) \in \{1,...,k\}^2$). As a result, the dot product $< \boldsymbol{q}_{CLS}, \boldsymbol{k}_{x,y} >$ also largely increases, and the softmax operation (based on the exponential function) enormously exaggerates the difference between the elements in S^h, generating a very peaked distribution, which zeros-out non-maxima (x,y) elements. We observed that, when using the softmax, the VT is able to minimize Eq. 9 by producing single-peak similarity maps which have a 0 entropy, each being composed of only one connected component with only one single token (i.e., $h_r = 1$ and $n_j = 1$).

Second, the spatial entropy (Eq. 7) is computed for each head separately and then averaged (Eq. 9) to allow each head to focus on *different* image regions. Note that, although computing the connected components (Eq. 6) is a non-differentiable operation, C^h is only used to "pool" the values of B^h (Eq. 8), and each C_j can be implemented as a binary mask (more details in the Appendix).

Finally, we remove the MSA skip connections only in the last block (Eq. 10-11) because, according to the results reported in (Raghu et al., 2021), removing the skip connections in the ViT intermediate blocks, brings to an accuracy drop. In contrast, in §5.1 we show that our strategy, which keeps the ViT architecture unchanged apart from the last block, is beneficial even when used *without* our spatial entropy loss. In the rest of this paper, we refer to our full method SAR as composed of the spatial entropy loss (§4.1) and the last-block MSA skip connection and LN removal (§4.2).

5 EXPERIMENTS

In §5.1 we analyse the contribution of the spatial entropy loss and the skip connection removal. In §5.2 we show that, using SAR and different VT architectures, we can improve VT training in different scenarios: (1) training from scratch on ImageNet-1K and small/medium datasets, (2) transfer learning on small datasets, (3) out-of-distribution testing. In §5.3 we analyse the properties of the attention maps generated using SAR. In the Appendix, we provide additional experiments in which we compare with the regularization approach of Liu et al. (2021a) and we use segmentation downstream tasks and self-supervised learning approaches. Each model was trained using 8 NVIDIA V100 32GB GPUs.

5.1 ABLATION STUDY

In this section, we analyse the influence of the λ value (§4.1), the removal of the skip connections and the LN in the last VT block (§4.2), and the use of the spatial entropy loss (§4.1). In all the ablation experiments, we use ImageNet-100 (IN-100) (Tian et al., 2020; Wang & Isola, 2020), which is a subset of 100 classes of ImageNet and ViT-S/16, a 22 million parameter ViT (Dosovitskiy et al., 2021), trained with 224×224 resolution images and 14×14 patches tokens ($k = 14$) with a patch resolution of 16×16 (Touvron et al., 2020). Moreover, in all the experiments in this section, we adopt the training protocol and the data-augmentations described in Liu et al. (2021b).
In Tab. 1 (a), we train from scratch all the models using 100 epochs and we show the impact on the test set accuracy using different values of λ. In the experiments of this table, we use our loss function ($\mathcal{L}_{tot} = \mathcal{L}_{ce} + \lambda \mathcal{L}_{se}$) and we remove both the skip connections and the LN in the last block (Eq. 10-11), thus the column $\lambda = 0$ corresponds to the result reported in Tab. 1 (b), Row "C" (see below). In the rest of the paper, we use the results obtained with this setting (IN-100, 100 epochs, etc.) and the best λ value ($\lambda = 0.01$) for **all** the other datasets, training scenarios (e.g., training from scratch or fine-tuning) and VT architectures (e.g., ViT, CvT, PVT, etc.). In fact, although a higher accuracy can very likely be obtained by tuning λ, our goal is to show that SAR is an easy-to-use regularization approach, even without tuning its only hyperparameter.

In Tab. 1 (b), we train from scratch all the models using 100 epochs and Row "A" corresponds to our run of the original ViT-S/16 (Eq. 1-2). When we remove the MSA skip connections (Row "B"), we observe a $+0.42$ points improvement, which becomes $+1.5$ if we also remove the LN (Row "C"). This experiment confirms that the last block patch tokens can learn more useful representations if we inhibit the MSA identity path (Eq. 10-11). However, if we also remove the skip connections in the subsequent MLP layer (Row "D"), the results are inferior to the baseline. Finally, when we use the spatial entropy loss with the original architecture (Row "E"), the improvement is marginal, but using $\mathcal{L}_{se}$ jointly with Eq. 10-11 (full model, Row "F"), the accuracy boost with respect to the baseline is

Table 1: ImageNet 100. (a) Influence of the spatial entropy loss weight λ (100 epochs). (b) Analysis of the different components of SAR (100 epochs). (c) Influence of the number of epochs.

λ	Top-1 Acc.
0	75.72
0.001	75.82
0.005	76.16
0.01	**76.72**
0.05	76.22
0.1	75.88

(a)

Model	Top-1 Acc.
A: Baseline	74.22
B: A + no MSA skip connections	74.64 (+0.42)
C: B + no LN	75.72 (+1.5)
D: C + no MLP skip connections	73.76 (-0.46)
E: A + spatial entropy loss	74.78 (+0.56)
F: A + SAR	76.72 (+2.5)

(b)

	Top-1 Acc.	
Model	100 ep.	300 ep.
ViT-S/16	74.22	80.82
ViT-S/16+SAR	**76.72** (+2.5)	**85.24** (+4.42)

(c)

much stronger. Tab. 1 (c) compares training with 100 and 300 epochs and shows that, in the latter case, SAR can reach a much higher relative improvement with respect to the baseline (+4.42).

5.2 Main results

Training from scratch on ImageNet. We start with a set of experiments on ImageNet-1K (IN-1K), in which we plug SAR into different VT architectures: ViT (Dosovitskiy et al., 2021), T2T (Yuan et al., 2021b), PVT (Wang et al., 2021) and CvT (Wu et al., 2021). We omit other common frameworks such as, for instance, Swin (Liu et al., 2021b) because of the lack of a `[CLS]` token in their architecture. Although the `[CLS]` token used, e.g, in §4.1 to compute S^h, can potentially be replaced by a vector obtained by average-pooling all the patch embeddings, we leave this for future investigations.

Moreover, for computational reasons, we focus on small-medium capacity VTs (see Tab. 2 for details on the number of parameters of each VT). Importantly, for each tested method, we use the original training protocol developed by the corresponding authors, including the learning rate schedule, the batch size, the VT-specific hyperparameter values and the data-augmentation type used to obtain the corresponding published results (see column "Training Protocol" in Tab. 2). Finally, as mentioned in §5.1, we keep fixed the only SAR hyperparameter ($\lambda = 0.01$). Although better results can likely be obtained by hyperparameter tuning (including the VT-specific hyperparameters), our goal is to show that SAR can be easily used in different VTs increasing their final testing accuracy. Thus, differently from a common practice, we have **not** tuned the hyperparameters on the IN-1K validation set. The results reported in Tab. 2 show that SAR improves *all* the tested VTs, independently of their specific architecture, model capacity or training protocol. Notably, SAR leads to almost 1 point difference with respect to ViT-S/16 (Touvron et al., 2020), which is obtained *without any additional learnable parameters*. Note that both PVT and CvT have a final grid resolution of 7×7, which is smaller than the 14×14 grid used in ViT and T2T, and this probably has a negative impact on our spatial based entropy loss. In Fig. 3, we show that, using ViT-S/16, SAR can largely speed-up training. For instance, ViT-S/16 + SAR, with 100 epochs, achieves almost the same accuracy as the baseline trained with 150 epochs, while we surpass the final baseline accuracy (79.8% at epoch 300) with only 250 training epochs (79.9% at epoch 250).

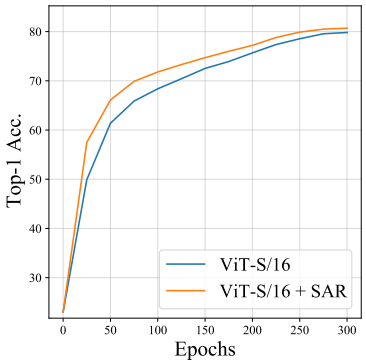

Figure 3: IN-1K, validation set accuracy with respect to the number of training epochs.

Training from scratch on small-medium datasets. Comparing the improvement obtained using ViT-S/16 + SAR on IN-1K (+0.9) with the corresponding improvement obtained on IN-100 (+4.42, Tab. 1 (c)), we observe that SAR is relatively more effective when the dataset size is smaller. This is likely due to the fact that, usually, regularization techniques are most effective with small(er) datasets (Liu et al., 2021a; Balestriero et al., 2022). For instance, Balestriero et al. (2022) empirically show that common data-augmentation techniques are more relatively effective with a smaller training dataset, which is quite intuitive, being data-augmentation used to *increase the diversity* of a dataset. Similarly, the goal of SAR is to extract unsupervised information from images to alleviate the need of large labeled datasets (§1), thus, this additional (self-)supervision is relatively more effective when there is less (manual) supervision data. To further validate this, we present another set of experiments in which we follow a recent trend of works (Liu et al., 2021a; El-Nouby et al., 2021; Cao & Wu,

Table 2: IN-1K experiments with different VTs. For each tested VT, we plugged SAR on the publicly available code of the corresponding baseline and we used the suggested hyperparameter values for training. All results but ours are reported from the corresponding paper indicated in the column "Training Protocol". All the results have been obtained using 300 training epochs.

Architecture	Training Protocol	Params (M)	FLOPs (G)	Top-1 Acc.
ViT-T/16	DEiT (Touvron et al., 2020)	5	1.6	72.2
ViT-T/16 + SAR	DEiT (Touvron et al., 2020)	5	1.6	**72.4** (+0.2)
ViT-S/16	DEiT (Touvron et al., 2020)	22	4.7	79.8
ViT-S/16 + SAR	DEiT (Touvron et al., 2020)	22	4.7	**80.7** (+0.9)
T2T-ViT-14	T2T (Yuan et al., 2021b)	21.5	6.1	81.5
T2T-ViT-14 + SAR	T2T (Yuan et al., 2021b)	21.5	6.1	**81.9** (+0.4)
PVT-Small	PVT (Wang et al., 2021)	24.5	3.8	79.8
PVT-Small + SAR	PVT (Wang et al., 2021)	24.5	3.8	79.84 (+0.04)
CvT-13	CvT (Wu et al., 2021)	20	4.5	81.6
CvT-13 + SAR	CvT (Wu et al., 2021)	20	4.5	**81.8** (+0.2)

Table 3: Training from scratch on small-medium datasets.

Model	ViT-S/16	ViT-S/16+SAR
Cars	35.3	**64.65** (+29.35)
Clipart	41.0	**64.95** (+23.95)
Painting	38.4	**57.11** (+18.17)
Sketch	37.2	**62.98** (+30.78)

Table 4: Transfer learning results.The first row corresponds to a standard fine-tuning protocol, while the other configurations include SAR either in the pre-training or in the fine-tuning stage.

SAR pre-training	SAR fine-tuning	CIFAR-10	CIFAR-100	Flowers	Pets
✗	✗	98.59	88.95	95.07	92.21
✓	✗	98.69 (+0.1)	89.19 (+0.24)	96.05 (+0.98)	92.7 (+0.49)
✗	✓	98.72 (+0.13)	88.95	95.1 (+0.03)	92.34 (+0.13)
✓	✓	98.65 (+0.06)	89.21 (+0.26)	96.1 (+1.03)	92.7 (+0.49)

2021) where VTs are trained from scratch on small-medium datasets (without pre-training on IN-1K). Specifically, we strictly follow the training protocol proposed by El-Nouby et al. (2021), where 5,000 epochs are used to train ViT-S/16 directly on each target dataset. The results are shown in Tab. 3, where the accuracy values of the baseline (ViT-S/16 without SAR) are reported from (El-Nouby et al., 2021). Tab. 3 shows that SAR can *drastically* improve the ViT-S/16 accuracy on these small-medium datasets, with a relative improvement ranging in $[+18.17, +30.78]$.

Transfer learning. In this battery of experiments, we evaluate SAR in a transfer learning scenario. We adopt the four datasets used in (Dosovitskiy et al., 2021; Touvron et al., 2020; Chen et al., 2021; Caron et al., 2021): CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009), Oxford Flowers102 (Nilsback & Zisserman, 2008), and Oxford-IIIT-Pets (Parkhi et al., 2012). The standard transfer learning protocol consists in pre-training on IN-1K, and then fine-tuning on each dataset. This corresponds to the first row in Tab. 4, where the IN-1K pre-trained model is ViT-S/16 in Tab. 2. The next three rows show different pre-training/fine-tuning configurations, in which we use SAR in one of the two phases or in both (see the Appendix for more details). All the configurations lead to an overall improvement of the accuracy with respect to the baseline, and show that SAR can be used flexibly. For instance, SAR can be used when fine-tuning a VT trained in a standard way, without the need to re-train it on ImageNet.

Out-of-distribution testing. Finally, we test the robustness of our VTs trained with SAR when the testing distribution is different from the training distribution. Specifically, following Bai et al. (2021), we use two different testing set: (1) ImageNet-A (Hendrycks et al., 2019), which are real-world images but collected from challenging scenarios (e.g., occlusions, fog scenes, etc.), and (2) ImageNet-C (Hendrycks & Dietterich, 2018), which is designed to measure the model robustness against common image corruptions. Note that training is performed on IN-1K. Thus, in Tab. 5, ViT-S/16 and ViT-S/16 + SAR correspond to the models we trained on IN-1K whose results on the IN-1K standard validation set are reported in Tab. 2. ImageNet-A and ImageNet-C are used *only for testing*, hence they are useful to assess the behaviour of a model when evaluated on a distribution different from the training distribution (Bai et al., 2021). The results reported in Tab. 5 show that SAR can greatly improve the robustness of ViT (note that with the mCE metric, the lower is better (Bai et al., 2021)). We presume that this is a side-effect of our spatial entropy loss minimization, which leads to heads usually focusing on the foreground objects and, therefore, reducing the dependence with respect to the background appearance variability distribution.

Table 5: Out-of-distribution testing on ImageNet-A (IN-A) and ImageNet-C (IN-C).

Model	**IN-A** (Acc. ↑)	**IN-C** (mCE ↓)
ViT-S/16	19.2	52.8
ViT-S/16 + SAR	**22.39** (+3.19)	**51.6** (-1.2)

Table 6: A comparison of the segmentation properties of the attention maps on PASCAL VOC-12.

Model	**Jaccard similarity** (↑)
ViT-S/16	19.18
ViT-S/16 + SAR	**31.19** (+12.01)

5.3 Attention map analysis

This section qualitatively and quantitatively analyses the attention maps obtained using SAR. Fig. 4 visually compares the attention maps obtained with ViT-S/16 and ViT-S/16 + SAR. As expected, standard training generates attention maps with a widely spread structure. Conversely, using SAR, a semantic segmentation structure clearly emerges. Note that, similarly to the self-supervised results shown in DINO (Caron et al., 2021), these segmentation masks have been obtained *without any pixel-level annotation*. However, differently from DINO, we have *explicitly* encouraged the network to produce attention maps with low spatial entropy. For a quantitative analysis, we follow the protocol used in (Caron et al., 2021; Naseer et al., 2021), where the Jaccard similarity is used to compare the ground-truth segmentation masks of the objects in PASCAL VOC-12 (Everingham et al., 2010) with the thresholded attention masks of the last ViT block. Specifically, the attention maps of all the heads are thresholded to keep 60% of the mass, and the head with the highest Jaccard similarity with the ground-truth is selected (Caron et al., 2021; Naseer et al., 2021). Tab. 6 shows that SAR significantly improves the segmentation results, quantitatively confirming the qualitative analysis in Fig. 4.

Figure 4: A qualitative comparison between the attention maps generated by ViT-S/16 (left) and ViT-S/16 + SAR (right). For each image, we show all the 6 attention maps (A^h) corresponding to the 6 last-block heads, computed using only the `[CLS]` token query.

6 Conclusions

In this paper we proposed SAR, a VT training regularization method which is based on a new spatial entropy loss. Specifically, the proposed loss is based on the intuitive idea that objects usually correspond to connected regions, and thus it penalizes spatially disordered attention maps. This way, we can extract additional self-supervised information from the training images, alleviating the need of huge labeled datasets to train VTs. Moreover, we also proposed to remove the last-block MSA skip connections and LN layers, a minor architectural change which prevents the propagation of the patch-token representations through the identity path. We empirically showed that this removal is beneficial, with and without our spatial entropy loss. SAR can be very easily plugged into the most common VT architectures, and our experiments show that this training regularization can boost the classification accuracy and speed-up training, independently of the specific VT or target task. SAR can *drastically* improve the accuracy when VTs are trained on small-medium datasets, which is specifically useful in those domains in which pre-training on ImageNet is not possible.

Limitations. Since training VTs is very computationally expensive, in our experiments we used only small/medium capacity VTs. We leave the extension of our empirical analysis to larger capacity VTs for the future. For the same computational reasons, we have not tuned hyperparameters on the datasets. However, we believe that the SAR accuracy improvement, obtained in all the tested scenarios without hyperparameter tuning, further shows its robustness and ease to use.

REFERENCES

Linda Altieri, Daniela Cocchi, and Giulia Roli. SpatEntropy: Spatial Entropy Measures in R. *arXiv:1804.05521*, 2018.

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

Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. MultiMAE: Multi-modal Multi-task Masked Autoencoders. *arXiv:2204.01678*, 2022.

Yutong Bai, Jieru Mei, Alan L. Yuille, and Cihang Xie. Are transformers more robust than cnns? In *NeurIPS*, 2021.

Randall Balestriero, Leon Bottou, and Yann LeCun. The effects of regularization and data augmentation are class dependent. *arXiv:2204.03632*, 2022.

Hangbo Bao, Li Dong, and Furu Wei. BEiT: BERT pre-training of image transformers. *arXiv:2106.08254*, 2021.

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

Michael Batty. Spatial entropy. *Geographical analysis*, 6(1):1–31, 1974.

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

Yun-Hao Cao and Jianxin Wu. Rethinking self-supervised learning: Small is beautiful. *arXiv:2103.13559*, 2021.

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

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

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.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv:2104.14294*, 2021.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. *arXiv:2202.04200*, 2022.

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

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

Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning. *arXiv:2202.03026*, 2022.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *ICCV*, 2021.

Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. UP-DETR: unsupervised pre-training for object detection with transformers. In *CVPR*, 2021.

Stéphane d'Ascoli, Hugo Touvron, Matthew L. Leavitt, Ari S. Morcos, Giulio Biroli, and Levent Sagun. ConViT: Improving vision transformers with soft convolutional inductive biases. In *ICML*, 2021.

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

Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, and Nenghai Yu. PeCo: Perceptual Codebook for BERT Pre-training of Vision Transformers. *arXiv:2111.12710*, 2021.

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.

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. In *ICCV*, 2021.

Alaaeldin El-Nouby, Gautier Izacard, Hugo Touvron, Ivan Laptev, Hervé Jégou, and Edouard Grave. Are large-scale datasets necessary for self-supervised pre-training? *arXiv:2112.10740*, 2021.

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

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010.

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

Costantino Grana, Daniele Borghesani, and Rita Cucchiara. Optimized block-based connected components labeling with decision trees. *IEEE Trans. Image Process.*, 19(6):1596–1609, 2010.

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. In *NeurIPS*, 2020.

Meng-Hao Guo, Junxiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. PCT: Point cloud transformer. *Comput. Vis. Media*, 7(2):187–199, 2021.

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

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

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *arXiv:2111.06377*, 2021.

Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2018.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *CVPR*, 2019.

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.

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

Tianyu Hua, Yonglong Tian, Sucheng Ren, Hang Zhao, and Leonid Sigal. Self-supervision through Random Segments with Autoregressive Coding (RandSAC). *arXiv:2203.12054*, 2022.

Drew A. Hudson and C. Lawrence Zitnick. Generative Adversarial Transformers. In *ICML*, 2021.

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

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

Ioannis Kakogeorgiou, Spyros Gidaris, Bill Psomas, Yannis Avrithis, Andrei Bursuc, Konstantinos Karantzalos, and Nikos Komodakis. What to Hide from Your Students: Attention-Guided Masked Image Modeling. *arXiv:2203.12719*, 2022.

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

Kunpeng Li, Ziyan Wu, Kuan-Chuan Peng, Jan Ernst, and Yun Fu. Guided attention inference network. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(12):2996–3010, 2020.

Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Improved multiscale vision transformers for classification and detection. *arXiv:2112.01526*, 2021a.

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

Yahui Liu, Enver Sangineto, Wei Bi, Nicu Sebe, Bruno Lepri, and Marco De Nadai. Efficient training of visual transformers with small datasets. *NeurIPS*, 2021a.

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*, 2021b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard S. Zemel. Understanding the effective receptive field in deep convolutional neural networks. *arXiv:1701.04128*, 2017.

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

Muzammal Naseer, Kanchana Ranasinghe, Salman H. Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. *arXiv:2105.10497*, 2021.

Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. *arXiv:2102.00719*, 2021.

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.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012.

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

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *arXiv:2108.08810*, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv:2204.06125*, 2022.

Yongming Rao, Wenliang Zhao, Guangyi Chen, Yansong Tang, Zheng Zhu, Guan Huang, Jie Zhou, and Jiwen Lu. DenseCLIP: Language-guided dense prediction with context-aware prompting. *arXiv:2112.01518*, 2021.

Q.R. Razlighi and Nasser Kehtarnavaz. A comparison study of image spatial entropy. In *Electronic Imaging*, 2009.

Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

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.

Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021.

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

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.

F. Tupin, M. Sigelle, and H. Maitre. Definition of a spatial entropy and its use for texture discrimination. In *ICIP*, 2000.

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

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

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

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.

Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan L. Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. *arXiv:2112.09133*, 2021.

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.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. SimMIM: A simple framework for masked image modeling. *arXiv:2111.09886*, 2021.

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

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

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. In *ICCV*, 2021b.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. *arXiv:2012.09164*, 2020.

Ce Zheng, Sijie Zhu, Matias Mendieta, Taojiannan Yang, Chen Chen, and Zhengming Ding. 3D Human pose estimation with spatial and temporal transformers. *ICCV*, 2021.

Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *Int. J. Comput. Vis.*, 127(3): 302–321, 2019.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan L. Yuille, and Tao Kong. iBOT: Image BERT Pre-Training with Online Tokenizer. *arXiv:2111.07832*, 2021.

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.

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

Appendix

A Pseudo-code of the Spatial Entropy loss

Fig. 5 shows the pseudo-code for $\mathcal{L}_{se}$ (Eq. (9)). The goal is twofold: to show how easy is to compute $\mathcal{L}_{se}$, and how to make it differentiable. Specifically, Eq. (6) is based on a connected component algorithm which is not differentiable. However, once C^h is computed, each element $C_j \in C^h$ ($C_j = \{(x_1, y_1), ..., (x_{n_j}, y_{n_j})\}$) can be represented as a binary mask M_j, defined as:

$$M_j[x, y] = \begin{cases} 1, & \text{if } (x, y) \in C_j \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Using Eq. (12), we compute the probability $P^h(C_j)$ of the component C_j using matrix multiplication, which is done efficiently on GPUs. In practice, Eq. (8) is computed as:

$$P^h(C_j) = \frac{1}{|B^h|} \sum_{(x,y)} B^h_{x,y} \odot M_j[x, y], \tag{13}$$

where $\odot$ is the element-wise product. This implementation makes it is possible to back-propagate the spatial entropy loss even if the connected component algorithm is not differentiable.

```
# bs    : batch size
# H     : number of heads
# k x k : size of the embedding grid
# S     : defined in Eq. 4 (main paper),
#         tensor with shape [bs, H, k, k]

def SpatialEntropy(S, eps=1e-9):
    bs, H, k, _ = S.size()
    m = torch.mean(S, dim=[2, 3], keepdim=True)
    B = torch.relu(S - m) + eps

    with torch.no_grad():
        batch_idxs, head_idxs, M = connected_components(B)

    p_u = torch.sum(B[batch_idxs, head_idxs] * M, dim=[2, 3])
    p_n = p_u / torch.sum(B, dim=[2, 3])
    SE = -torch.sum((p_n * torch.log(p_n)) / (bs * H)
    return SE
```

Figure 5: PyTorch-like pseudocode for our Spatial Entropy loss.

B COMPARING THE SPATIAL ENTROPY LOSS WITH OTHER SOLUTIONS

In our preliminary experiments, we replaced the spatial entropy loss with a different loss based on a *total variation denoising* (Rudin et al., 1992), criterion, which we formulated as:

$$\mathcal{L}_{tv} = \frac{1}{H}\sum_{h=1}^{H}\sum_{(x,y)\in\{1,\dots,k\}^2} |S^h_{x,y} - S^h_{x,y+1}| + |S^h_{x,y} - S^h_{x+1,y}|. \tag{14}$$

However, $\mathcal{L}_{tv}$ drastically underperforms $\mathcal{L}_{se}$ and tends to produce blurred (more uniform) attention maps. The main difference between $\mathcal{L}_{tv}$ and $\mathcal{L}_{se}$ is that the thresholding (Eq. 5) and the adjacency-based clustering (Eq. 6) operations in $\mathcal{L}_{se}$ group together image regions of variable size and shape which have in common high attention scores for a specific head, and, therefore, they presumably represent the same semantics (e.g., a specific object). In contrast, $\mathcal{L}_{tv}$ compares patch tokens which are adjacent to each other (e.g., $S^h_{x,y}$ and $S^h_{x,y+1}$) but which do not necessarily share the same semantics (e.g., $S^h_{x,y}$ may belong to the background while $S^h_{x,y+1}$ belongs to a foreground object). Thus, the implicit local inductive bias of the two losses is different: in case of $\mathcal{L}_{se}$, the inductive bias is that the image regions corresponding to the highest attention scores (for a specific head) should be spatially grouped in few, big "blobs", while, in the case of $\mathcal{L}_{tv}$, the inductive bias is that generic adjacent regions should have similar attention scores.

Finally, we have empirically tested slightly different similarity metrics. For instance, replacing Eq. (4) with a cosine similarity computed as:

$$\hat{S}^h_{x,y} = \frac{< \boldsymbol{q}^h_{CLS}, \boldsymbol{k}^h_{x,y} >}{||\boldsymbol{q}^h_{CLS}||||\boldsymbol{k}^h_{x,y}||}, \quad (x,y) \in \{1,...,k\}^2, \tag{15}$$

and keeping all the rest unchanged (e.g., Eq. (5)-Eq. (9)), we get a slightly lower top-1 accuracy value when training on IN-100 (76.25 versus 76.72 in Tab. 1 (c)). This is likely due to the fact that $\hat{S}^h$ in Eq. (15) does not correspond to the metric used to compute the attention for the main task loss (Eq. (3)). Thus, merging the main task loss (e.g., the cross entropy loss) with the spatial entropy loss may be more difficult.

C ADDITIONAL EXPERIMENTS

In this section, we present additional experiments following the same setting used in §5 (e.g., same λ value, using the original VT/algorithm hyperparameters, etc.).

C.1 TRAINING FROM SCRATCH ON IMAGENET-100

In this section, we extend the IN-100 experiments shown in §5.1 by including different VT architectures. Tab. 7 shows that SAR improves *all* the tested VTs, consistently with the results shown in Tab. 2. Note that the SAR-based improvements on IN-100 are relatively larger than those obtained on IN-1K, confirming that SAR is particularly beneficial with relatively smaller datasets.

We further analyze the impact of the amount of training data using different subsets of IN-100 with different sampling ratio (ranging from 25% to 75%, with images randomly selected). We use the same training protocol of Tab. 1 (c) (e.g., 100 training epochs, etc.) and we test on the *whole* IN-100 validation set. Tab. 8 shows the results, confirming that, with less data, the accuracy boost obtained using SAR can significantly increase (e.g., with 75% of the data we have a 10.5 points improvement).

C.2 SELF-SUPERVISED EXPERIMENTS ON IMAGENET-100

In this section, we use SAR in a fully self-supervised scenario. Since self-supervised learning algorithms are very time consuming, we use IN-100, which is a medium-size dataset. We plug SAR on top of two state-of-the-art VT-based self-supervised learning algorithms: MoCo-v3 (Chen et al., 2021) and DINO (Caron et al., 2021) (§2). When we use MoCo-v3, in $\mathcal{L}_{tot}$ (§4.1), we replace the cross-entropy loss ($\mathcal{L}_{ce}$) with the contrastive loss used in (Chen et al., 2021). Similarly, when

Table 7: IN-100 experiments with different VTs. For each tested VT, we plug SAR on the publicly available code of the corresponding baseline and we use the suggested hyperparameter values for training. All the results are obtained using 100 training epochs.

Model	**Top-1 Acc.**
ViT-S/16 (Dosovitskiy et al., 2021)	74.22
ViT-S/16 + SAR	**76.72** (+2.5)
T2T-ViT-14 (Yuan et al., 2021b)	82.42
T2T-ViT-14 + SAR	**83.96** (+1.54)
PVT-Small (Wang et al., 2021)	76.57
PVT-Small + SAR	**77.78** (+1.21)
CvT-13 (Wu et al., 2021)	83.38
CvT-13 + SAR	**85.20** (+1.82)

Table 8: IN-100 experiments with different sampling ratios.

Sampling Ratio	**ViT-S/16**	**ViT-S/16+SAR**
0.25	21.66	29.06 (+7.4)
0.50	29.86	38.02 (+8.16)
0.75	35.62	46.12 (+10.5)
1.00	74.22	76.72 (+2.5)

we use DINO, we use as the main task loss the "self-distillation" proposed in (Caron et al., 2021), jointly with its multi-crop strategy. We used the official code of MoCo-v3 and DINO and we strictly follow the algorithms and the training protocols of the baseline methods, including all the default hyperparameters suggested by the corresponding authors. However, for computational reasons, we used a 1024 batch size for MoCo-v3 and MoCo-v3 + SAR, and a batch size of 512 for DINO and DINO + SAR. The VT backbone is ViT-S/16 for all the methods. More details in Appendix D.

We evaluate all models (with and without SAR) using the standard self-supervised evaluation protocol, consisting in freezing the network after training and then training a linear classifier on top of the frozen features (Caron et al., 2021; Chen et al., 2021). The results are reported in Tab. 9 (a), and show that, on IN-100, SAR significantly improves these state-of-the-art algorithms, including DINO (which inspired our work).

We qualitatively compare the attention maps obtained with and without SAR in Fig. 6 (MoCo-v3) and Fig. 7 (DINO). Fig. 6 shows that, in MoCo-v3 + SAR, the head-specific attention maps focus on slightly different aspects of the main object, while in MoCo-v3, the attention is much more "disordered" (spread over the whole image). On the other hand, when comparing DINO with DINO + SAR (Fig. 7), the attention map differences are more subtle. However, the higher inter-head variability in DINO + SAR is one interesting difference. For instance, while DINO's maps usually focus only on the main foreground object, in DINO + SAR, different heads cover different foreground objects (e.g., the cat and the sink in Row 11) or different background regions (e.g., the road and the sky in the "train" figure of Row 5). This difference is probably due to the difference in how DINO and DINO + SAR are optimized. In fact, in DINO, the only source of supervision is the comparison between two different views of the same image (§2), which likely encourages the network to focus on the objects most frequently in common. On the other hand, in DINO + SAR, the creation of connected regions is also encouraged inside each image view using $\mathcal{L}_{se}$.

These qualitative observations are confirmed by the quantitative results in Tab. 9 (b), where we follow the protocol described in §5.3. SAR increases the Jaccard similarity of both self-supervised algorithms.

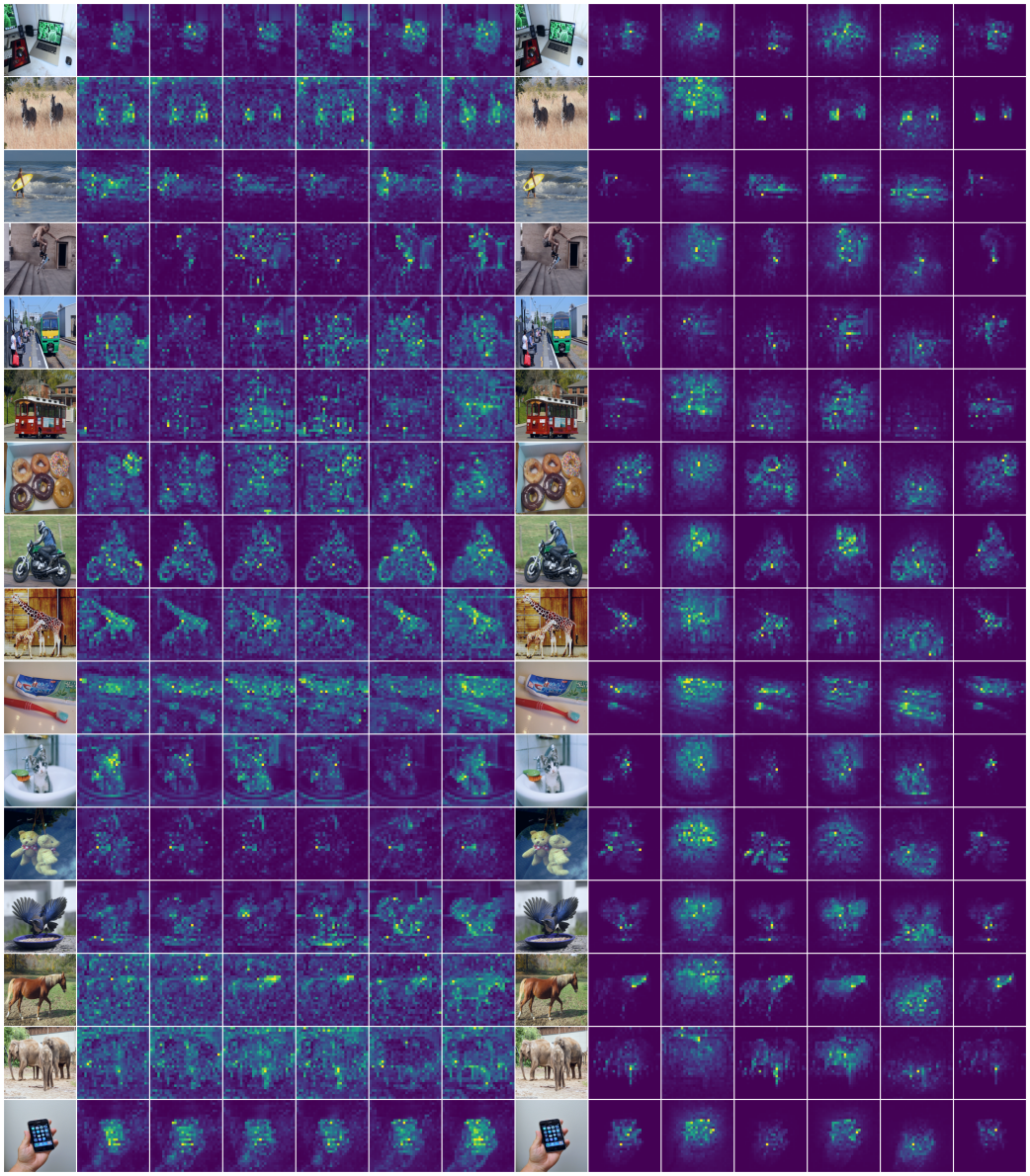

Figure 6: A qualitative comparison between the attention maps generated by MoCo-v3 and MoCo-v3 + SAR with the ViT-S/16 backbone (training on IN-100). For each image, we show all the 6 attention maps (A^h) corresponding to the 6 last-block heads, computed using only the `[CLS]` token query.

Figure 7: A qualitative comparison between the attention maps generated by DINO and DINO + SAR with the ViT-S/16 backbone (training on IN-100).

Table 9: Quantitative results of self-supervised methods trained on IN-100 for 300 epochs. (a) Accuracy on IN-100 evaluated through linear probing. (b) Segmentation properties of the corresponding attention maps evaluated on PASCAL VOC-12.

Method	**Top-1 Acc.**
MoCo-v3 (Chen et al., 2021)	77.60
MoCo-v3 + SAR	**78.88** (+1.28)
DINO (Caron et al., 2021)	77.42
DINO + SAR	**79.90** (+2.48)

(a)

Method	**Jaccard similarity**
MoCo-v3 (Chen et al., 2021)	27.40
MoCo-v3 + SAR	**33.10** (+6.17)
DINO (Caron et al., 2021)	35.06
DINO + SAR	**35.28** (+0.22)

(b)

Table 12: Segmentation downstream task on the ADE20K dataset.

Model	**mIoU**
PVT-Small (Wang et al., 2021)	39.8
PVT-Small+SAR	**41.1** (+0.3)

C.3 Supervised case: additional experiments on ImageNet-1K

C.3.1 Segmentation downstream task

In this section, we use a semantic segmentation task as the downstream task. Specifically, we adopt the well-known segmentation benchmark ADE20K (Zhou et al., 2019) and we use PVT (§5) as the VT backbone. PVT was chosen because it can be adapted to work as a Feature Pyramid Network and has already been used for segmentation tasks (Wang et al., 2021). In Tab. 12 we report the segmentation results (39.8 mIoU) obtained by Wang et al. (2021) when using ADE20K to fine-tune PVT-Small pre-trained on IN-1K (the results of this pre-training, again taken from (Wang et al., 2021), are reported in Tab. 2). In case of PVT-Small + SAR, we analogously used ADE20K to fine-tune the PVT-Small + SAR model pre-trained on IN-1K, which corresponds to the results reported in Tab. 2. Note that, as shown in Tab. 2, the improvement of SAR with respect to the PVT baseline was quite marginal when pre-trained on IN-1K, mostly because of the reduced grid size (§5.2). The results in Tab. 12 show that, using SAR for a downstream segmentation task, we can increase the PVT accuracy. Although the improvement is marginal, it is consistent with all the other experiments, in which we always get a performance boost, independently of the application scenario or the dataset, and without hyper-parameter tuning.

C.3.2 Comparison with other VT regularization methods

In this section, we compare SAR with the Dense Relative Localization (DRLoc) loss (Liu et al., 2021a), which is based on an auxiliary self-supervised task used to regularize VT training (§2). DRLoc encourages the VT to learn spatial relations within an image by predicting the relative distance between the (x, y) positions of randomly sampled output embeddings from the $k \times k$ grid of the last layer L. Tab. 13 shows that SAR outperforms DRLoc, using the same training protocol and a ViT-S/16 architecture trained on IN-1K.

C.3.3 Additional qualitative analysis of the attention maps

In this section, we extend the analysis of §5.3 providing additional qualitative results obtained using supervised training on IN-1K.

Fig. 8 shows the attention maps of ViT-S/16 and ViT-S/16 + SAR. These attention maps show that the ViT-S/16 attention scores are spread over all the image, while in ViT-S/16 + SAR they are much more spatially clustered and usually focused on the main object(s) of the input image. For example, the first row shows that the heads of ViT-S/16 focus on the upper part of the image, and only the keyboard of the laptop emerges. Vice versa, from the ViT-S/16 + SAR attention heads, it is possible to recognise

Table 13: Quantitative results using a ViT-S/16 trained with and without SAR and compared with (Liu et al., 2021a). All models are trained in ImageNet-1K for 300 epochs.

Method	**Top-1 Acc.**
ViT-S/16 (Dosovitskiy et al., 2021)	79.8
ViT-S/16 + DRLoc (Liu et al., 2021a)	80.2 (+0.4)
ViT-S/16 + SAR	**80.7** (+0.9)

the shape and size of the laptop precisely. Similarly, the second last row shows an example where the input image contains some elephants. While the different heads of ViT-S/16 seem to focus mainly on the background, the first head of ViT-S/16 + SAR is clearly focused on the elephants. Importantly, different heads of ViT-S/16 + SAR usually focus on different semantic concepts, which shows that there are no collapse phenomena using the spatial entropy loss (see Eq. (7) and the corresponding discussion in §4.3).

D Implementation details

In the following, we list the implementation details. Unless stated otherwise, we train our models on 8 V100 GPUs.

D.1 ViT based models

We train our models using the public code of Touvron et al. (2020)[1] for ViT and we modify the original code when we use SAR, as described in §4.2. The models are trained with a batch size of 1024, using the AdamW optimizer (Loshchilov & Hutter, 2019) with initial learning rate of 0.001, a cosine learning rate schedule, a weight decay of 0.05 and using the original data-augmentation protocol.

D.2 Hybrid architectures

In all the supervised experiments, we used the officially released code for PVT (Wang et al., 2021)[2], T2T (Yuan et al., 2021b)[3] and CvT (Wu et al., 2021)[4], strictly following the original training protocol for each architecture.

PVT and T2T are trained with batch size of 1024, using the AdamW optimizer with an initial learning rate of 0.001, momentum 0.9 and weight decay of 0.05. CvT is trained with a batch size of 2048 and an initial learning rate of 0.02, decayed with a cosine schedule. The data augumentations of the original articles are based on the DeiT protocol (Touvron et al., 2020). We refer the reader to the original papers for further details. When we use SAR, we modify the original public code, following §4.2.

D.3 Fine-tuning

We fine-tune the ViT-S/16 models pretrained on IN-1K (see Tab. 2), always keeping unchanged the VT architecture used in the pre-training stage. This is done to avoid making the adaptation task more difficult, since each of the four datasets used in Tab. 4 is composed of a relatively small number of samples. For instance, when SAR is used during pre-training but removed during fine-tuning (second row of Tab. 4), in the fine-tuning stage we use only $\mathcal{L}_{ce}$ for training but we do *not* re-introduce skip connections or LN layers in the last block (i.e., we use Eq. (10)-Eq. (11) when fine-tuning). Conversely, when the pre-training is performed without SAR, which is introduced only in the fine-tuning stage (third row of Tab. 4), when fine-tuning, we use $\mathcal{L}_{tot} = \mathcal{L}_{ce} + \lambda\mathcal{L}_{se}$, but keeping the standard skip connections and the LN layer in the last block (Eq. (1)-Eq. (2)).

[1] https://github.com/facebookresearch/deit
[2] https://github.com/whai362/PVT/tree/v1
[3] https://github.com/yitu-opensource/T2T-ViT
[4] https://github.com/microsoft/CvT

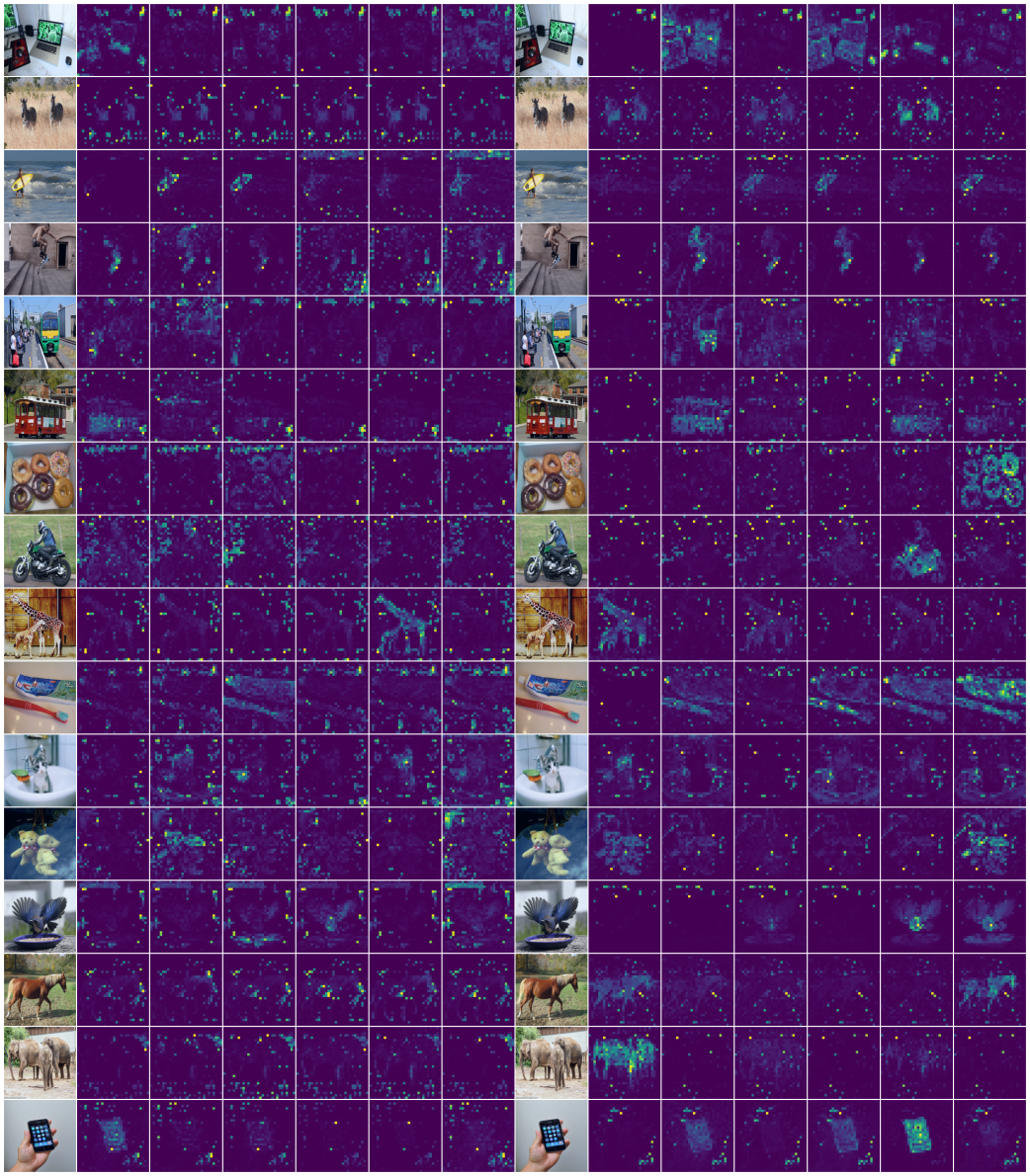

Figure 8: A qualitative comparison between the attention maps generated by ViT-S/16 and ViT-S/16 + SAR (supervised case, training on IN-1K).

The models are fine-tuned for 100 epochs with a batch size of 512, and an initial learning rate of 0.0005 decayed with a cosine schedule.

D.4 SELF-SUPERVISED EXPERIMENTS

In our self-supervised experiments, we adopt the original code for MoCo-v3 (Chen et al., 2021)[5] and DINO (Caron et al., 2021)[6] with a ViT-S/16 backbone. For computational reasons, we restrict our experiments training the models on IN-100 for 300 epochs. Moreover, to fit the available resources, we reduce the batch size to 1024 for MoCo-v3, while DINO is trained with the default multi-crop strategy $(2 \times 224^2 + 10 \times 96^2)$, but with a batch size of 512. We thoroughly follow the authors' specifications for the other hyperparameters. The results in Tab. 9 are obtained using a standard linear evaluation protocol in which the pretrained backbone is frozen, and a linear classifier is trained on top of it, using SGD for 100 epochs on IN-100.

E DATASET LICENSING DETAILS

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

PASCAL VOC 2012 images abide by the Flickr Terms of Use[10].

Cars images have a non-commercial research and educational license[11]

ClipArt, Painting and Sketches are part of the DomainNet dataset which is released under a fair use license[12].

The ImageNet-A[13] and ImageNet-C[14] images are released with unknown licence, so we refer to the original authors to use these datasets.

[5] https://github.com/facebookresearch/moco-v3
[6] https://github.com/facebookresearch/dino
[7] https://www.cs.toronto.edu/~kriz/cifar.html
[8] https://www.robots.ox.ac.uk/~vgg/data/flowers/102/
[9] https://image-net.org/download
[10] http://host.robots.ox.ac.uk/pascal/VOC/voc2012/
[11] http://ai.stanford.edu/~jkrause/cars/car_dataset.html
[12] http://ai.bu.edu/DomainNet/#dataset
[13] https://github.com/hendrycks/natural-adv-examples
[14] https://github.com/hendrycks/robustness