Efficient Self-supervised Vision Transformers for Representation Learning

Chunyuan Li[1] **Jianwei Yang**[1] **Pengchuan Zhang**[1] **Mei Gao**[2] **Bin Xiao**[2] **Xiyang Dai**[2]
Lu Yuan[2] **Jianfeng Gao**[1]
[1]Microsoft Research at Redmond, [2]Microsoft Cloud + AI
{chunyl,jianwyan,penzhan,xuga,bixi,xidai,luyuan,jfgao}@microsoft.com

Abstract

This paper investigates two techniques for developing efficient self-supervised vision transformers (EsViT) for visual representation learning. First, we show through a comprehensive empirical study that multi-stage architectures with sparse self-attentions can significantly reduce modeling complexity but with a cost of losing the ability to capture fine-grained correspondences between image regions. Second, we propose a new pre-training task of region matching which allows the model to capture fine-grained region dependencies and as a result significantly improves the quality of the learned vision representations. Our results show that combining the two techniques, EsViT achieves 81.3% top-1 on the ImageNet linear probe evaluation, outperforming prior arts with around an order magnitude of higher throughput. When transferring to downstream linear classification tasks, EsViT outperforms its supervised counterpart on 17 out of 18 datasets. The code and models will be publicly available.

1 Introduction

Self-supervised learning (SSL) with Transformers [57] has become a de facto standard of model choice in natural language processing (NLP). The dominant approaches such as GPT [48] and BERT [16] are pre-training on a large text corpus and then fine-tuning to various smaller task-specific datasets, showing superior performance. Larger Transformers pre-trained with larger-scale language datasets often lead to a stronger generalization ability, demonstrated by improved performance in downsteam tasks (with no sign of performance saturation yet), as exemplified in GPT-3 [2].

In computer vision (CV), however, self-supervised visual representation learning is still dominated by convolutional neural networks (CNNs). Sharing a similar goal/spirit with NLP, SSL in CV aims to learn general-purpose image features from raw pixels without relying on manual supervisions, and the learned networks serve as the backbone of various downstream tasks such as classification, detection and segmentation. Recently, impressive performance have been achieved by CNN-based SSL, outperforming state-of-the-art (SoTA) fully-supervised pre-training methods [26, 5] on tasks with a limited number of labels. The key to success is contrastive learning: maximizing agreement of learned representations between differently augmented views of the same example. Recent works, including SimCLR-v2 [10], BYOL [25] and SwAV [5], have scaled up the model of CNN-based contrastive learning to hundreds of millions of parameters. However, SSL has not enjoyed the same scaling success in CV as that in NLP.

Several attempts have been made to close the gap by combining SSL with Transformer and self-attention architectures. Early works include Selfie [55], which generalizes the concept of masked language modeling of BERT for images. The idea has been recently revisited in Vision Transformer (ViT) [19] via pre-training on a much larger scale dataset, *e.g.,* JFT-300M. ImageGPT (iGPT) [8] generalizes the concept of auto-regressive language modeling of GPT for images, showing encourag-

Preprint.

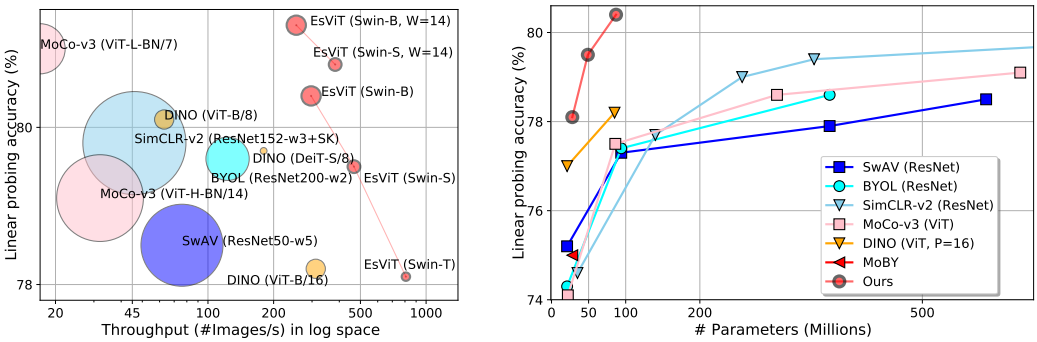


Figure 1: Efficiency vs accuracy comparison under the linear classification protocol on ImageNet. Left: Throughput of all SoTA SSL vision systems, circle sizes indicates model parameter counts; Right: performance over varied parameter counts for models with moderate (throughout/#parameters) ratio. Please refer Section 4.1 for details.

ing ImageNet recognition accuracy with a large model size. Contrastive learning with ViT has also been studied very recently in DINO [6] and MoCo-v3 [12], where new SoTA result by linear probe evaluation on ImageNet-1K is achieved, by exhaustively consuming computation resource on full self-attention operators with long sequences of split image patches.

Aiming to improve the *efficiency* of Transformer-based SSL, this paper presents **E***fficient* **s***elf-superivsed* **Vi***sion* **T***ransformers* (EsViT), by using a multi-stage architecture and a region-based pre-training task for unsupervised representation learning. Our main findings and contributions can be summarized as follows:

- We present the first comprehensive empirical study to show the pros and cons of multi-stage vision Transformer architectures for SSL. An intriguing property of self-supervised monolithic Transformers is firstly reported in our paper: automatic discovery of semantic correspondence between local regions. Though greatly reducing compute complexity, we find that the multi-stage architecture causes the loss of this property.

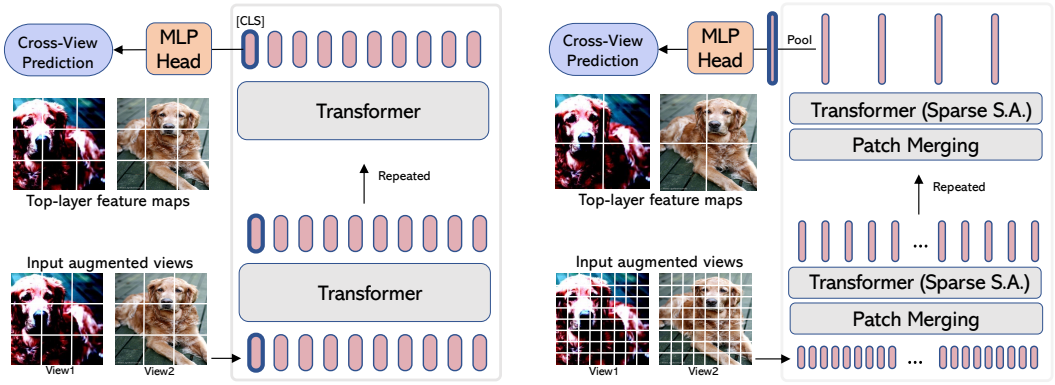
- A region matching pre-train task is proposed to alleviate this issue, and further improve the learned representations and attentions.

- We validate the new EsViT, which combines the two techniques, on a range of tasks. It significantly reduces the cost in building SoTA SSL vision systems, as summarized in Figure 1, and shows better scaling performance on accuracy vs. throughput and model size. Under the linear evaluation protocol, EsViT achieves 81.3% top-1 accuracy, showing the best performance compared with all systems, and is $3.5\times$ parameter-efficient and has at least $10\times$ higher throughput than previous SoTA (81.0%, MoCo-v3 with ViT-BN-L/7 [12]). Compared with its supervised counterpart Swin Transformers [39], EsViT shows superior performance on 17 out 18 datasets, when transferring the learned representations to downstream linear classification tasks.

2 Methods

Background. An overview of existing Transformer-based SSL methods is illustrated in Figure 2(a). They are emerged very recently to lead the state-of-the-art performance on the ImageNet linear probe task [12, 6]. It inherits the successes from (1) monolithic Transformer architectures that dominate in NLP [16, 48], and (2) instance-level contrastive learning objectives that demonstrate arguably the best SSL performance in computer vision [9]. As common practice, different crops of each image are taken under random data augmentation. Each augmented view is partitioned into non-overlapped patches with patch size $P \times P$. Concatenating a `[CLS]` token (as a global token) with these patches, they are encoded by a stack of Transformer layers with the same network configurations, such as ViT [19] or DeiT [54]. Hence, the input sequence and output representation share the same number of tokens, and it is clear that each local patch has its unique corresponding top-layer feature representation. The global representation of different views are further projected into a latent space using a shared MLP head, where the similarity among augmented views from the same image is maximized.

Though simple and effective, the existing Transformer-based SSL methods require a large amount of compute resources to reach SoTA performance. To strike for a better tradeoff between accuracy and



(a) Baseline monolithic architecture (b) Proposed multi-stage architecture

Figure 2: Architecture comparison. (a) The traditional monolithic transformer architecture. For all layers, the transformer blocks share the same network configurations and input token sequence sizes are the same. (b) The multi-stage Transformer architecture organizes an input image into a long sequence of smaller patches, sparse self-attentions (S.A.) are utilized at early stages to maintain model expressiveness while reducing computational complexity; The neighboring tokens at an intermediate layer are gradually merged, constituting a short sequence to ease the compute burden of self-attention at late stages.

efficiency, we present EsViT, by using a multi-stage architecture and a region-based pre-training task, as described next.

2.1 Network Architectures: From Monolithic to Multi-stage ViT Backbone

Inspired by great successes of the multi-stage ConvNet architecture such as VGG [49]/ResNets [27] for computer vision, the multi-stage Transformer-based networks have been explored very recently in the supervised learning setting [56, 59, 39, 70, 62]. This paper presents the first empirical study of multi-stage Transformer architectures for SSL. Though straightforward in implementation, changing from monolithic to multi-stage architecture without careful treatments may lose some desirable properties of self-supervised Transformers, as discussed later.

Multi-stage ViT. An overview of the multi-stage Transformer architecture is shown in Figure 2 (b). Each stage consists of a *patch merging/embedding* module, and a *Transformer with sparse self-attention* module. (i) The patch merging module plays a slightly different role in different stages. In the first stage, it splits an input RGB image into non-overlapping patches. Each patch is treated as a "token", constructed as a concatenation of the raw pixel RGB values, which is further projected into a C-dimension feature. In the later stage, the patch merging module concatenates the features of each group of 2×2 neighboring patches, and applies a linear layer on the $4C$-dimensional concatenated features. This reduces the number of tokens by a multiple of $2 \times 2 = 4$, and the output dimension is set to $2C$. (ii) A Transformer with sparse self-attention module are then employed to enable interactions among the merged features. The two modules above are repeated for multiple times, typically 4 times, resulting in a multi-stage ViT. As a result, a hierarchical representation is generated: the number of tokens is reduced and the feature dimension (and the number of heads in self-attentions) of each token is increased, as the network gets deeper.

Since a larger number of patches is often produced at the early stages, an efficient Transformer with sparse self-attentions is considered to reduce the computational complexity. The basic idea is to split the feature maps into non-overlapping local windows (with size $W \times W$), and self-attention is performed within each local window. This however has one drawback that features in different local windows cannot interact. Various methods have been proposed to best approximate full-attention among all features, with different trade-off between accuracy and efficiency.

We briefly describe three schemes as follows, and benchmark them in the experiments. (i) *Swin Transformer* [39]: A shifted window partitioning approach is proposed, which alternates between two partitioning configurations in consecutive Transformer blocks, so that each local feature is grouped into different windows in self-attentions. (ii) *Vision Longformer (ViL)* [70]: Features in each local

window are further allowed to attend all features in the 8-neighboring windows. (*iii*) *Convolution vision Transformer (CvT)* [62]: Features in neighboring windows are considered in the convolutional projection in self-attentions. Please refer each paper for detailed description.

Relative Position Bias. To facilitate SSL, we consider relative position bias [39] to characterize the spatial information between features for the three efficient Transformers aforementioned, and do not use absolute position embeddings. This is because augmented views of varied resolutions can be cropped from anywhere in an image in SSL, maintaining the relative positions is easy in implementation, and is largely sufficient for invariance learning among these views.

2.2 Pre-training Tasks: Delving into Views with Regions

We build our SSL systems on top of the recent method *self-distillation with no labels* (DINO) [6]. Knowledge distillation is a learning paradigm where a student network $g_{\boldsymbol{\theta}_s}$ is trained to match the output of a given teacher network $g_{\boldsymbol{\theta}_t}$, parameterized by $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_t$ respectively. The neural network g is composed of a backbone f (*e.g.,* Transformers or ConvNets), and of a projection head h: $g = h \circ f$. The features used in downstream tasks are the output of backbone f. In SSL, different augmented views $\tilde{\boldsymbol{x}}$ of an image $\boldsymbol{x}$ are fed into backbone network to obtain feature maps $\boldsymbol{z} = f(\tilde{\boldsymbol{x}})$. An MLP head combined with softmax further converts the feature vector $z \in \boldsymbol{z}$ into probability $p = h(z)$. DINO updates teacher and student network alternatively: (i) Given a fixed teacher network, the student network is updated by minimizing the cross-entropy loss: $\boldsymbol{\theta}_s \leftarrow \arg\min_{\boldsymbol{\theta}_s} \mathcal{M}(s, t; \boldsymbol{\theta}_s)$, where $\mathcal{M}(s, t) = -p_t \log p_s$. (ii) The teacher model is updated as an exponential moving average (EMA) on the student weights $\boldsymbol{\theta}_t \leftarrow \lambda\boldsymbol{\theta}_t + (1 - \lambda)\boldsymbol{\theta}_s$, with λ following a cosine schedule from 0.996 to 1 during training. Please refer to [6] for details.

More precisely, from a given image, we generate a set $\mathcal{V}$ of different views[1] following [6]. The resulting feature map at the top layer for each view is $\boldsymbol{z} = [z_1, \ldots, z_T]$, where T is the sequence length, and z_i is a region-level representation for the local patch at position i. Average pooling is applied to obtain the view-level representation $\bar{z} = \text{avg-pool}(\boldsymbol{z})$.

View-level task Given the augmented view set for student $\mathcal{V}$ and teacher $\mathcal{V}^*$, a set of pairs $\mathcal{P} = \{(s, t) | \tilde{\boldsymbol{x}}_s \in \mathcal{V}, \tilde{\boldsymbol{x}}_t \in \mathcal{V}^* \text{ and } s \neq t \}$ is constructed to perform cross-view prediction tasks. We consider the pre-training task at the view level proposed by [6]:

$$\mathcal{L}_V = \frac{1}{|\mathcal{P}|} \sum_{(s,t)\in\mathcal{P}} \mathcal{M}_V(s, t), \text{ with } \mathcal{M}_V(s, t) = -p_s \log p_t, \tag{1}$$

where $p_s = h(\bar{z}_s)$ and $p_t = h(\bar{z}_t)$ are the probability output of an MLP head h over the view-level representations $\bar{z}_s$ and $\bar{z}_t$, learned by student and teacher, respectively. In DINO, ViT/DeiT are considered, hence the view-level representation is the feature of the `[CLS]` token.

Region-level task In[6], the $\mathcal{L}_V$ encourages "local-to-global" correspondences only at a coarse level: the large crop and the small crop are matched in the view level, leaving region-to-region correspondence unspecified. As shown in our experiments later, training a multi-stage network with $\mathcal{L}_V$ results in sub-optimal representations, though efficiency is greatly improved. Inspired by the success of masked language modeling task in BERT, it is important to have region-level pre-training task for computer vision, so that the model can take into account the co-occurrences/structures between local features. Unfortunately, directly performing masked patch prediction (MPP) for the multi-stage Transformer architecture is infeasible, as the one-to-one correspondences between the input visual tokens and output features get diluted due to the merging operation. Even for monolithic architectures, MPP has not been proved effective in computer vision, as empirically shown in [19].

To address this problem, we propose an effective, dense self-supervised learning method that directly works at the level of local features by taking into account their correspondences:

$$\mathcal{L}_R = \frac{1}{|\mathcal{P}|} \sum_{(s,t)\in\mathcal{P}} \mathcal{M}_R(s, t), \text{ with } \mathcal{M}_R(s, t) = -\frac{1}{T} \sum_{i=1}^{T} p_{j^*} \log p_i, \; j^* = \arg\max_j \frac{z_i^T z_j}{\|z_i\| \|z_j\|}, \tag{2}$$

[1]This set often contains views of two different resolutions $\mathcal{V} = [\mathcal{V}_g, \mathcal{V}_l]$, where $\mathcal{V}_g = \{\tilde{\boldsymbol{x}}_{g_i} | i = 1, 2\}$ is a global-view set of higher resolution, and $\mathcal{V}_l = \{\tilde{\boldsymbol{x}}_{l_i} | i = 1, \ldots, 8\}$ is a local-view set of lower resolution. All views $\mathcal{V}$ are passed through the student while only the global views $\mathcal{V}_g$ are passed through the teacher.

where $p_i = h'(z_i)$ and $p_j = h'(z_j)$ are the probability outputs of a new MLP head h' over the local features of student $z_i \in \boldsymbol{z}_s$ and teacher $z_j \in \boldsymbol{z}_t$, respectively. j^* is the index of the feature in $\boldsymbol{z}_t$ that best matches the i-th feature in $\boldsymbol{z}_s$, in the sense of highest cosine similarity.

We can consider $\mathcal{L}_R$ as a proxy to mimick masked language modeling in BERT, where the "ground-truth" local token is a soft label provided by the teacher network, while the student network makes predictions to match that target, based on the context of regions in a different augmented view. Though the ideas of leveraging local-level pre-training tasks have been explored for ConvNets in the context of contrastive learning in a bid to improve dense visual prediction tasks [65, 60], our $\mathcal{L}_R$ is different in that no negative samples/queue is needed, we find that $\mathcal{L}_R$ is particularly effective in improving Transformers and their learned attentions, as demonstrated in our experiments.

The overall pre-training objective of EsViT is $\mathcal{L} = \mathcal{L}_R + \mathcal{L}_V$, we learn to match the feature distributions at both the view and region levels by minimizing the cross-entropy loss w.r.t. the parameters of the student network $g_{\boldsymbol{\theta}_s}$. An visual illustration is in Figure 3, and the full algorithm is in Appendix. By default, the full objective is used from the beginning. One can also load a checkpoint trained by $\mathcal{L}_V$ only, and add $\mathcal{L}_R$ for continual pre-training, which is shown effective in boosting performance in our experiments.

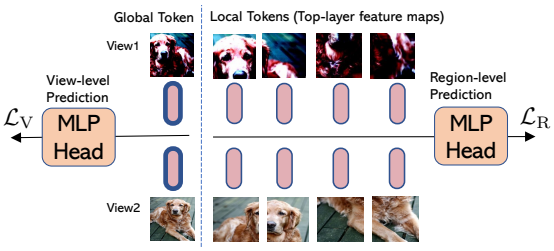


Figure 3: Pre-training objectives.

Note that applying $\mathcal{L}_R$ on the traditional monolithic Transformer architecture can be prohibitively computationally expensive, as it requires $\mathcal{O}(T^2)$ to compute $\mathcal{L}_R$. For a typical image of resolution 224×224, the feature map length of ViT/DeiT (with patch size 16) at the top layer is $T = 196$, while the multi-stage architecture yields $T = 49$, which requires 3 times less compute in computing $\mathcal{L}_R$.

3 Related Works

Self-supervised ConvNets. ConvNets-based SSL have been extensively studied in the literature. Based on the pre-training tasks, they can broadly categorized into three classes: Handcrafted pretext tasks [17, 42, 45, 22, 71, 31, 72, 46, 18], contrastive learning [20, 76, 43, 28, 1, 26, 9, 25] and prototype learning [4, 5, 34, 63, 67, 29, 69]. It is also known that data augmentations play a crucial role in SSL pipeline [11, 5, 52, 32]. The impact of pre-training dataset size/quality is explored for ConvNets in SSL [23, 68]. To date, the search of best pre-taining tasks/datasets and augmentations are based on CNNs. Among them, SimCLR-v2 [10], BYOL [25] and SwAV [5] achieve the highest ImageNet linear probe performance with large ConvNet architectures. The performance tends to saturate with an increasingly growing model size, raising a question if ConvNets reach a limit in SSL.

Self-supervised vision Transformers. The research on Transformer-based self-supervised representation learning just scratches the tip of the iceberg, and only a few attempts are made on this topic. ImageGPT [8] and MoCo-v3 [12] dedicate huge compute resource with large models to exploring the frontier. DINO [6] achieves comparable performance of large self-supervised ConvNets using small/medium-size Transformers. The proposed EsViT further pursue efficient solutions to self-supervised vision Transformers.

Transformers for vision. Vision Transformers (ViT) [19] shows the great potentials of generalizing Transformers for computer vision, by achieving compelling accuracy in supervised learning, especially with large-scale data and high capacity models. DeiT [54] further provides an effective ViT training strategy to ease the adaption of Transformers for practitioners. Transformers have also been applied to other vision tasks, ranging from low-level tasks such as image generation [44, 8] and enhancement [7, 66], to high-level tasks such as object detection [3, 75, 73, 14] and segmentation [58, 61], and to vision-language tasks [41, 51, 13, 50, 35, 33, 74, 38].

4 Experimental Results

We study unsupervised pre-training performed in ImageNet-1K dataset [15] without labels. The default training details are described as follows, mostly following [6]. We train with the Adamw optimizer [40], a batch size of 512, and and total epochs 300. Linear warmup of the learning rate is used during the first 10 epochs, with its base value determined with the linear scaling rule [24]: $lr = 0.0005 * \text{batchsize}/256$. After this warmup, the learning rate is decayed with a cosine schedule.

Method	#Parameters ↓	Throughput (Image/s) ↑	Linear ↑	k-NN ↑
SoTA SSL methods with Big ConvNets				
SwAV, RN50w5 [5]	586	76	78.5	67.1
BYOL, RN200w2 [25]	250	123	79.6	73.9
SimCLR-v2, RN152w3+SK [10]	794	46	79.8	73.1
Skyline methods with excessively long sequences for self-attentions				
DINO, DeiT-S/8 [6]	21	180	79.7	78.3
DINO, ViT-B/8 [6]	85	63	80.1	77.4
MoCo-v3, ViT-B-BN/7 [12]	85	∼63	79.5	-
MoCo-v3, ViT-L-BN/7 [12]	304	∼17	81.0	-
iGPT, iGPT-XL [8]	6801	-	72.0	-
EsViT, Swin-S/$W=14$	49	383	80.8	79.1
EsViT, Swin-B/$W=14$	87	254	**81.3**	**79.3**
Transformer-based SSL, with moderate sequence length for self-attentions				
Masked Patch Pred., ViT-B/16 [19]	85	312	79.9†	-
DINO, DeiT-S/16 [6]	21	1007	77.0	74.5
DINO, ViT-B/16 [6]	85	312	78.2	76.1
MoCo-v3, ViT-B/16 [12]	85	312	76.7	-
MoCo-v3, ViT-H-BN/16 [12]	632	∼32	79.1	-
MoBY, Swin-T [64]	28	808	75.1	-
EsViT, Swin-T	28	808	78.1	75.7
EsViT, Swin-S	49	467	79.5	77.7
EsViT, Swin-B	87	297	**80.4**	**78.9**

Table 1: Comparison with SoTA across different architectures on ImageNet linear probing. ViT-BN is ViT that has BatchNorm [21], and "/P" denotes a patch size of $P \times P$. "∼" indicates through-puts estimated by comparing different papers, detailed in Appendix. † The mask patch prediction in [19] is pre-trained on JFT-300M and end-to-end fine-tuned in ImageNet, which we append as a reference.

We build our systems based on Swin Transformers in our experiments. Swin-B has a model size and computation complexity similar to ViT-B/DeiT-B (patch size 16). We also considered Swin-T and Swin-S, which have the complexity that are similar to those of ResNet-50 (DeiT-S) and ResNet-101, respectively. The default window size is $W=7$.

We evaluate the proposed EsViT to answer three questions: `Q1`: How does EsViT perform on standard ImageNet benchmark compared to SoTA methods? `Q2`: how effective EsViT is when transferring to downstream tasks? `Q3`: When does the intriguing property of self-supervised Transformers exist, including learned correspondence and attentions?

4.1 Comparisons with Prior Art on ImageNet

One major common protocol to evaluate SSL is linear probe on ImageNet-1K, where features are extracted from a frozen backbone, and a supervised linear classifier is trained. For all Transformer models, we use the concatenation of view-level features $\bar{z}$ in the last 4 layers (the results are similar to the use of last 3 or 5 layers in our initial experiments). We report top-1 and k-NN classification accuracy on the ImageNet validation set. Table 1 presents comparisons with all SoTA SSL vision systems across various network architectures. Please refer to Figure 1 for visual comparisons over scaling parameter counts and throughput. Our main findings are summarized below.

Comparisons with self-supervised Transformers. Both iGPT [8] and the masked patch prediction in [19] are earlier variants of masked auto-encoding schemes, mimicking the behaviors of GPT [48] and BERT [16] in NLP, respectively. The DINO- and MoCo-based ViT has higher accuracy and smaller models than iGPT, under the same linear probing protocol and training data. At the similar level of model size and compute complexity, the proposed EsViT improve SoTA methods DINO/MoCo-v3 by a large margin: EsViT (Swin-B) outperforms DINO (ViT-B/16) by 2.2% linear probe accuracy and 2.8% k-NN accuracy in absolute values. EsViT (Swin-B) even performs slightly better than DINO (ViT-B/8) (0.3% higher linear probe accuracy and 1.5% higher k-NN accuracy), with $4\times$ higher throughput. MoBY [64] is a con-current work that investigates multi-stage ViT in SSL. With the same architecture Swin-T, our EsViT pre-training tasks significantly outperform MoBY, showing 3% higher accuracy. Overall, the proposed EsViT (Swin-B/W=14) shows the best performance (top-1 accuracy 81.3%, top-5 accuracy 95.5%, k-NN accuracy 79.3%), compared with all systems, and is $3.5\times$ parameter-efficient and has at least $10\times$ higher throughput than previous

Arch.	Objectives	Window S.	Linear	k-NN
Swin-T	$\mathcal{L}_V$	7	77.0	74.2
	$\mathcal{L}_V+\mathcal{L}_R$	7	**78.1**	**75.7**
	$\mathcal{L}_V$	14	77.9	75.5
	$\mathcal{L}_V+\mathcal{L}_R$	14	**78.7**	**77.0**
Swin-S	$\mathcal{L}_V$	7	79.2	76.8
	$\mathcal{L}_V+\mathcal{L}_R$	7	**79.5**	**77.7**
	$\mathcal{L}_V$	14	79.4	77.3
	$\mathcal{L}_V+\mathcal{L}_R$	14	**80.8**	**79.1**
Swin-B	$\mathcal{L}_V$	7	79.6	77.7
	$\mathcal{L}_V+\mathcal{L}_R$	7	**80.4**	**78.9**
	$\mathcal{L}_V$	14	80.5	78.3
	$\mathcal{L}_V+\mathcal{L}_R$	14	**81.3**	**79.3**

Table 2: Ablations of pre-train tasks and window sizes.

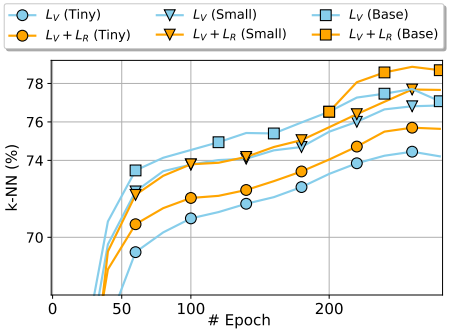


Figure 4: Learning curve of different pre-training tasks. For Base model, $\mathcal{L}_R$ is added from the 200th epoch.

SoTA MoCo-v3. Our smaller model EsViT (Swin-S/W=14) achieves 80.8% top-1 accuracy, approaching SoTA (81.0%, MoCo-v3 with ViT-BN-L/7 [12]) with $6\times$ less parameters and $22\times$ higher throughput, and outperforms the 2nd place method (80.1%, DINO with ViT-B/6 [6]), with $1.5\times$ less parameters and $6\times$ higher throughput. We leave the study of further scaling EsViT as future work.

Comparisons with big ConvNets. We compare with the SoTA big ResNets reported by SimCLR-v2 [10], BYOL [25] and SwAV [5]. Among them, the best accuracy 79.8% under the linear probing protocol is reported by SimCLR-v2 with SK-ResNet, where Selective Kernel (SK) [37] is a form of attention to enhance CNNs. It is clear in Figure 1 (b) that all ConvNets-based SSL methods show an envelope in the regime of scaling up model sizes after passing 500M. EsViT achieves better accuracy than their highest envelope, with $16\times$ less model parameters and $8\times$ higher throughput.

Longer sequences for self-attentions. It is observed that using longer sequences in self-attentions can improve recognition accuracy: In iGPT [8] sequence length is increased in the pixel domain; DINO and MoCo-v3 explore longer sequences by reducing the patch size to 8×8 or 7×7 ("/8" or"/7" in Figure 1 and Table 1), respectively. This keeps the model parameter counts unchanged, but dramatically increases compute complexity (measured by increased FLOPs and decreased throughput), *e.g.,* to around $6\times$ reported in MoCo-v3. In EsViT, longer sequences in self-attention is implemented by increasing the window size. We experiment this by considering a window size of $W = 14$, and report the results in Table 2, where we see a consistent accuracy improvement of around 1-2%. In Table 1, EsViT with Swin-S/$W = 14$ achieves highest 79.1% k-NN accuracy, and comparable linear probe accuracy with SoTA, with almost an order of magnitude higher efficiency.

Ablation: Comparison of sparse attentions. We investigate different efficient sparse Transformers in Table 3. DeiT is shown as a baseline reference. Batch size = 1024 in this experiment. To ensure fair comparison, we modify all into a 4-stage architecture with the number of Transformer layers in each stage as 2-2-6-2. ViL and CvT can be safely used as alternatives to Swin, and could potentially achieve better performance. We choose to scale up Swin in our experiments first to study the limit of EsViT, leave the scaling of others as future work.

Method	#Param.	Im./s	100 epochs Linear	100 epochs k-NN	300 epochs Linear	300 epochs k-NN
DeiT	21	1007	73.1	69.0	75.9	73.2
Swin	28	808	75.3	70.0	77.1	73.7
ViL	28	386	75.4	70.1	77.3	73.9
CvT	29	848	75.5	70.6	77.6	74.8

Table 3: Different sparse attentions in EsViT.

Ablation: The effectiveness of $\mathcal{L}_R$. We compare the pre-training objective with and without $\mathcal{L}_R$ in Table 2. Across different model scales and window sizes, the proposed region level $\mathcal{L}_R$ can consistently improve the performance. The gains can be clearly seen by k-NN accuracy (around 1-2%), where no additional tuning is needed as in linear probe. Figure 4 demonstrates that $\mathcal{L}_R$ helps model convergence, and can be used as a drop-in to improve models trained with the view level task.

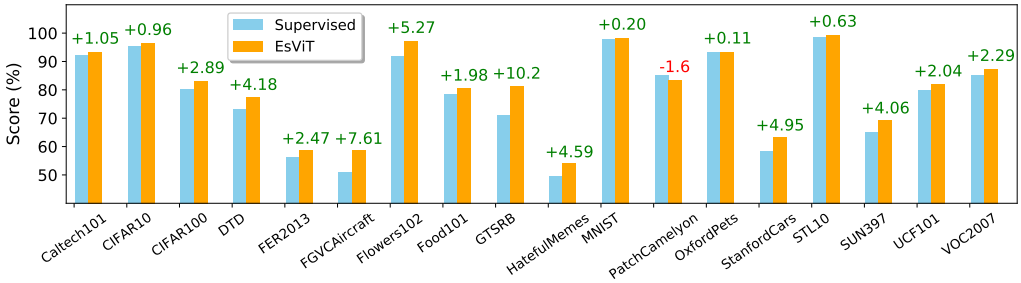


Figure 5: Transfer learning to a wide variety of datasets. Fitting a linear classifier on EsViT's features outperforms using its supervised counterpart on 17 out of 18 datasets.

	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
Supervised	46.0	68.1	50.3
EsViT	46.2	68.0	50.6
	AP^{mb}	AP^{mb}_{50}	AP^{mb}_{75}
Supervised	41.6	65.1	44.9
EsViT	41.6	64.9	44.8

Table 4: COCO Detection & Segmentation.

Pre-train Data	ImageNet-1K		18 Datasets	
	Linear	k-NN	Scores	# Wins
Supervised	-	-	77.29	-
ImageNet-1K	77.1	73.7	80.66	16
WebVision-v1	75.4	69.4	80.00	14
OpenImages-v4	69.6	60.3	77.97	10
ImageNet-22K	73.5	66.1	**81.03**	**17**

Table 5: Impact of the pre-train datasets.

4.2 Transfer Learning

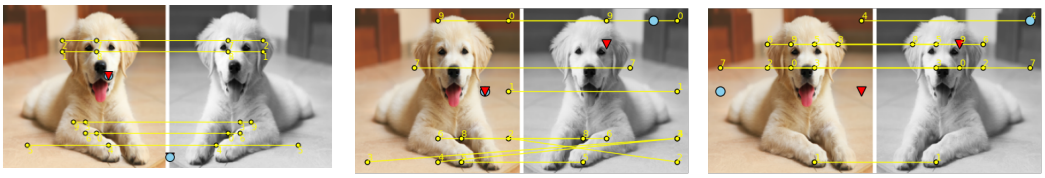
We also conduct transfer learning in downstream tasks to evaluate the quality of learned representations. Two sets of tasks are considered:

- *Classification on a suite of 18 small datasets*. As exemplified in [47], it is a common and clean approach to evaluate a learned representation by fitting a linear classifier on the representation and measuring its performance across multiple datasets. We study 18 datasets used in [47]. Automatic hyper-parameter tuning is considered to ensure fairness of comparison. Besides averaged `scores`, we report `# wins` as the number of datasets on which the model outperforms its supervised counterpart. Detailed dataset description and settings are in Appendix.
- *Detection and segmentation on COCO*. Different from previous monolithic self-supervised ViT, the multi-stage architecture in EsViT can be readily used for dense visual tasks that require hierarchical feature representations.

Comparison with supervised learning counterparts. We compare with the supervised-learning Swin, whose checkpoints are downloaded from the official codebase[2]. Figure 5 shows the classification results of Swin-S, EsViT consistently outperforms its supervised variant, often by a large margin. Similar conclusions are drawn for other model sizes. On COCO detection and segmentation task, however, comparable results with the supervised counterpart are obtained, as shown in Table 4 for Swin-T.

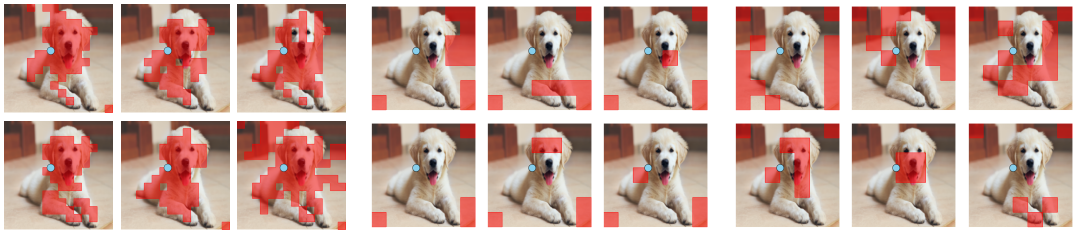
Effects of larger, less-curated pre-train datasets. The performance of Transformer-based SSL research has thus far been limited to highly curated pre-train data such as ImageNet-1K. To push the frontier in leveraging large amounts of unlabeled data, we explore the effects of pre-training from larger, less-curated image datasets: WebVision-v1 [36], OpenImages-v4 [30] and ImageNet-22K [15], described in Appendix. The pre-train epochs on different datasets are adjusted so that all models see a similar number of augmented views. We summarize the results in Table 5 and would like to emphasize the following findings. First, all EsViT pre-trained checkpoints outperform supervised checkpoint in downstream classification tasks, but performance varies a lot, with ImageNet-22K checkpoint showing the best transfer ability. Second, ImageNet-1K pre-trained model shows the best ImageNet-1K linear probe performance. We hypothesize that it is not only the size of pre-train dataset matters, but also the distribution of image classes matters: more diverse and well-balanced distribution results in a stronger generalization ability. Similar observations are reported in a concurrent work [68] for ConvNets.

[2] `https://github.com/microsoft/Swin-Transformer`



(a) DINO: DeiT-S (b) EsViT: $\mathcal{L}_V$ (c) EsViT: $\mathcal{L}_V + \mathcal{L}_R$

Figure 6: The learned correspondences. **Yellow** lines are the top-10 correspondences between two views, where the numbers indicates the rankings of similarity scores, yellow dots with the same number are paired. The **blue** dot and **red** triangle indicates the most similar local regions that correspond to the global feature of the view itself and the other view, respectively.



(a) DINO: DeiT-S (b) EsViT: $\mathcal{L}_V$ (c) EsViT: $\mathcal{L}_V + \mathcal{L}_R$

Figure 7: Visualization of the the learned attention map for different heads in the last layer. The query is the **blue** dot in the center of the images. We visualize masks (as **red**) obtained by thresholding the self-attention maps to keep 60% of the probability mass. Note that all 6 heads are visualized for DINO with DeiT-S, and 6 out of 24 heads in EsViT are chosen to visualize (ranked by entropy values). Please see enlarged pictures with all heads in Appendix.

4.3 Qualitative Studies

Visualization of correspondences. Given two views of the same image, we use the pre-trained backbone to extract the top-layer features $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. For each feature vector in $\boldsymbol{z}_1$, we find the feature vector in $\boldsymbol{z}_2$ that best matches it in terms of highest cosine similarity, as defined in Equation (2). In Figure 6, we show the top-10 correspondences between two views for three methods. In Figure 6 (b), EsViT with $\mathcal{L}_V$ tends to identify pairs in the background as the most matched ones (and in a wrong way in this example). This could be a valid solution to $\mathcal{L}_V$, as the invariance in the level of aggregated global features does not necessarily induce invariances in the local region level. This is significantly alleviated with $\mathcal{L}_R$ (shown in Figure 6 (c)), a task that implicitly requires local matching.

Surprisingly, DINO is able to learn good correspondences even without the region-level matching task. To the best of our knowledge, this is a previously unreported intriguing property of self-supervised Transformers with monolithic architectures: good semantic correspondences are automatically learned. We hypothesize that features at lower layers (image patch itself in the extreme case) can directly pass to higher layers, and the former regularizes the latter to remain discriminative. Nevertheless, the proposed $\mathcal{L}_R$ can dramatically reduce the issue, and is good remedy to rescue the loss of semantic correspondence for the multi-stage architecture. In Appendix, we quantitatively measures the correspondence learning ability of these SSL methods on ImageNet validation dataset, the observations are consistent.

Visualization of attention maps. We look at the self-attention in the different heads of the last layer in Figure 7. A local region on the edge of the main object is employed as query, and the attended regions are highlighted in red for those the query's top 60% mass are assigned. In Appendix, we visualize more examples with different query positions. DINO tends to automatically learns class-specific attention maps leading to foreground object segmentations, regardless of its query located in foreground or background. This is probably because main objects remain as the major invariance factor in different augmented views. This property is lost when a multi-stage architecture is employed, as shown in EsViT with $\mathcal{L}_V$. These patterns are consistent for different heads. After introducing $\mathcal{L}_R$ for EsViT, we note that the attention maps become more diverse in different heads, *i.e.,* entropy values of attentions get more skewed, and attended regions are more different. This is

perhaps because $\mathcal{L}_R$ requires each region to consider many matching tasks to regions in different augmented views, each head automatically learns to distribute the tasks and complete a few of them.

5 Discussions

In this paper, we present efficient self-supervised vision Transformers (EsViT) with two major insights: a multi-stage Transformer architecture with sparse self-attentions, and a region-matching pre-training task. The synergy of both helps EsViT reach the best performance of SoTA SSL vision systems with significantly less compute and smaller model size.

A potential future work is to perform self-supervised pre-training for bigger multi-stage ViT models. Our study also reveals that exploration of effective solutions to learn from larger and less curated pre-training data is a key but less studied factor in paving the way toward the scaling success of SSL vision systems. Another interesting research direction to investigate the correspondence learning property of self-supervised Transformers for image-text paris, *i.e.,* , automatic grounding between regions and words.

References

[1] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *NeurIPS*, 2019.

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

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

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

[5] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

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

[7] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. *arXiv preprint arXiv:2012.00364*, 2020.

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

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

[10] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

[11] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

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

[13] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*, 2019.

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

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

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

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

[18] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *NeurIPS*, 2019.

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

[20] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *T-PAMI*, 2015.

[21] Jonathan Frankle, David J Schwab, and Ari S Morcos. Training Batchnorm and only Batchnorm: On the expressive power of random features in CNNs. *arXiv preprint arXiv:2003.00152*, 2020.

[22] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[23] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021.

[24] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[25] 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, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

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

[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, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

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

[30] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, 2020.

[31] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, 2016.

[32] Chunyuan Li, Xiujun Li, Lei Zhang, Baolin Peng, Mingyuan Zhou, and Jianfeng Gao. Self-supervised pre-training with hard examples improves visual representations. *arXiv preprint arXiv:2012.13493*, 2020.

[33] Gen Li, Nan Duan, Yuejian Fang, Daxin Jiang, and Ming Zhou. Unicoder-VL: A universal encoder for vision and language by cross-modal pre-training. *arXiv preprint arXiv:1908.06066*, 2019.

[34] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020.

[35] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[36] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017.

[37] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *CVPR*, 2019.

[38] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *ECCV*, 2020.

[39] 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 preprint arXiv:2103.14030*, 2021.

[40] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in Adam. *arXiv preprint arXiv:1711.05101*, 2018.

[41] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. VilBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *NeurIPS*, 2019.

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

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

[44] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, 2018.

[45] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[46] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. *NIPS*, 2016.

[47] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[48] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[49] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[50] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. VL-BERT: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

[51] Hao Tan and Mohit Bansal. LXMERT: Learning cross-modality encoder representations from transformers. *EMNLP*, 2019.

[52] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. *arXiv preprint arXiv:2005.10243*, 2020.

[53] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. MLP-mixer: An all-MLP architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

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

[55] Trieu H Trinh, Minh-Thang Luong, and Quoc V Le. Selfie: Self-supervised pretraining for image embedding. *arXiv preprint arXiv:1906.02940*, 2019.

[56] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. *CVPR*, 2021.

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

[58] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. *arXiv preprint arXiv:2012.00759*, 2020.

[59] 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. *arXiv preprint arXiv:2102.12122*, 2021.

[60] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. *arXiv preprint arXiv:2011.09157*, 2020.

[61] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. *arXiv preprint arXiv:2011.14503*, 2020.

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

[63] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

[64] Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. Self-supervised learning with swin transformers. *arXiv preprint arXiv:2105.04553*, 2021.

[65] Yuwen Xiong, Mengye Ren, and Raquel Urtasun. Loco: Local contrastive representation learning. *arXiv preprint arXiv:2008.01342*, 2020.

[66] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *CVPR*, 2020.

[67] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, 2016.

[68] Tian Yonglong, Olivier J. Henaff, and Aaron van den Oord. Divide and contrast: Self-supervised learning from uncurated data. *arXiv preprint arXiv:2105.08054*, 2021.

[69] Xiaohang Zhan, Jiahao Xie, Ziwei Liu, Yew-Soon Ong, and Chen Change Loy. Online deep clustering for unsupervised representation learning. In *CVPR*, pages 6688–6697, 2020.

[70] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021.

[71] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.

[72] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017.

[73] Minghang Zheng, Peng Gao, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. *arXiv preprint arXiv:2011.09315*, 2020.

[74] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and VQA. *AAAI*, 2020.

[75] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

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

Supplementary Material: Efficient Self-supervised Vision Transformers for Representation Learning

A Methods

A.1 Algorithms

We summarize the training algorithm procedure of EsViT with $\mathcal{L}_V + \mathcal{L}_R$ in Algorithm 1. To clearly outline the main idea of the algorithm, we show the algorithm for two augmented views. For the full algorithm to deal with multi-crop, please refer to our codebase. In Algorithm 1, for a mini-batch of size n, the teacher/student network consists of three output variables: (1) $\texttt{p} \in \mathbb{R}^{n \times K}$ is the probability vector for the view-level representation, output by an MLP head. (2) $\texttt{z} \in \mathbb{R}^{n \times T \times P}$ is the feature map, containing T region-level features of dimension P. (3) $\texttt{pz} \in \mathbb{R}^{n \times T \times K}$ are probability vectors of $\texttt{z}$, output by a different MLP head.

A.2 Network architecture configurations and implementation details

The window size is set to $W = 7$ by default. The query dimension of each head in self-attentions is $d = 32$, and the hidden layer width of each MLP is $4\times$ of its input's width, for all experiments. The architecture configurations of model variants employed in the experiments are summarized in Table 6. Some notable implementation detailed are described as follows:

- The three configurations Swin-T, Swin-S and Swin-B are almost identical to the original implementation [39], except that we add special treatments to deal with input augmented views of different resolutions, when the resolution (feature map size more specifically) is not divisible by the window size (*i.e.,* resolution 96 and window size=7 or 14).

- Swin-T and Swin-S with window size $W = 14$ are customized by us to allow full self-attention in stage 3 (where the majority of model capacity is allocated to) and stage 4, to study the impact of longer sequences in EsViT.

- In the original ViL [70] and CvT [62] papers, different positional embedding strategies and multi-stage network configurations were employed. We modify them by only utilizing relative position bias and their proposed sparse self-attention mechanisms, and create a similar 4-stage architecture with Swin-T for fair comparison.

B Experiments

B.1 Throughput estimate and conversion

Since different papers report throughput on different hardwares, it is not ready to compare the numbers directly. Noting that all papers report the throughput for ViT-B/DeiT-B, we use this number to align and convert the throughput. In Table 7, we describe our process and results of standardizing the throughput.

B.2 Region matching tasks for more sparse self-attentions

In Table 8, we report the performance of EsViT with and without the region matching task $\mathcal{L}_R$ for three efficient Transformer variants: Swin, ViL and CvT. In this experiments, batch size is set to 1024. The model is pre-trained for 300 epochs. At the 200th epoch, $\mathcal{L}_R$ is added. We see that $\mathcal{L}_R$ can boost the accuracy by a large margin on all three Transformer variants.

B.3 Linear probe on a suite of small datasets

Datasets. Table 9 shows details and source of all datasets used for linear probe, including the number of classes, the size of training set and testing set, metrics used in evaluation, as well as a

Algorithm 1: EsViT with $\mathcal{L}_V + \mathcal{L}_R$, pseudocode with 2-crop.

```
# gs, gt:  student and teacher networks
# Cv, Cr:  view and region center (K)
# tmp_s, tmp_t:  student and teacher temperatures
# a, b:  network and center momentum rates.
# n:  batch size, K: MLP-head-projected probability vector length,
  T: last layer feature map length, P: last layer feature vector length
gt.params = gs.params
# The main training loop
for  x in loader:
    x1, x2 = augment(x), augment(x)  # two random views

    # student output, p:n×K, pz:n×T×K, z:n×T×P
    p_s1, pz_s1, z_s1 = gs(x1)
    p_s2, pz_s2, z_s2 = gs(x2)
    # teacher output, p:n×K, pz:n×T×K, z:n×T×P
    p_t1, pz_t1, z_t1 = gt(x1)
    p_t2, pz_t2, z_t2 = gt(x2)

    # view-level loss
    loss_v = Hv(p_s1, p_t2)/2 + Hv(p_s2, p_t1)/2
    # region-level loss
    loss_r = Hr(pz_s1, pz_t2, z_s1, z_t2)/2+Hr(pz_s2, pz_t1, z_s2, z_t1)/2

    loss = loss_v/2 + loss_r/2
    loss.backward()  # back-propagate

    # update student, teacher and centers
    update(gs)                                          # AdamW for student
    gt.params = a * gt.params + (1 − a) * gs.params     # EMA for teacher
    Cv = b * Cv + (1 − b) * cat([p_t1,p_t2].mean(0))    # EMA for view center
    Cr = b * Cr + (1 − b) * cat([pz_t1,pz_t2].mean(0))  # EMA for region center

# The view-level loss function
def Hv(s, t):
    t = t.detach()  # stop gradient
    s = softmax(s / tmp_s, dim=-1)
    t = softmax((t - Cv) / tmp_t, dim=-1)
    return  - (t * log(s)).sum(dim=-1).mean()
# The region-level loss function
def Hr(ps, pt, zs, zt):
    pt = pt.detach()                                    # stop gradient
    ps = softmax(ps / tmp_s, dim=-1)                    # n×T×K
    pt = softmax((pt - Cr) / tmp_t, dim=-1)             # n×T×K
    sim_matrix = torch.matmul(zs , zt.permute(0, 2, 1))  # n×T×T
    sim_idx = sim_matrix.max(dim=-1)[1].unsqueeze(2)     # n×T×1
    pt_idxed = torch.gather(pt, 1, sim_idx.expand(-1, -1, pt.size(2)))
    return  - (pt_idxed * log(ps)).sum(dim=-1).mean()
```

	Stage 1	Stage 2	Stage 3	Stage 4
Merging Rate Feature Map	4× 56 × 56	8× 28 × 28	16× 14 × 14	32× 7 × 7
Swin-T, $W=7$	concat 4×4 , 96-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{96-d (3 heads)}\end{bmatrix}\times2$	concat 2×2, 192-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{192-d (6 heads)}\end{bmatrix}\times2$	concat 2×2, 384-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{384-d (12 heads)}\end{bmatrix}\times6$	concat 2×2, 768-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{768-d (24 heads)}\end{bmatrix}\times2$
Swin-S, $W=7$	concat 4×4 , 96-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{96-d (3 heads)}\end{bmatrix}\times2$	concat 2×2, 192-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{192-d (6 heads)}\end{bmatrix}\times2$	concat 2×2, 384-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{384-d (12 heads)}\end{bmatrix}\times18$	concat 2×2, 768-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{768-d (24 heads)}\end{bmatrix}\times2$
Swin-B, $W=7$	concat 4×4 , 128-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{128-d (4 heads)}\end{bmatrix}\times2$	concat 2×2, 256-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{256-d (8 heads)}\end{bmatrix}\times2$	concat 2×2, 512-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{512-d (16 heads)}\end{bmatrix}\times18$	concat 2×2, 1024-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{1024-d (32 heads)}\end{bmatrix}\times2$
Swin-T, $W=14$	concat 4×4 , 96-d, LN $\begin{bmatrix}\text{window size}: 14\times14\\ \text{96-d (3 heads)}\end{bmatrix}\times2$	concat 2×2, 192-d, LN $\begin{bmatrix}\text{window size}: 14\times14\\ \text{192-d (6 heads)}\end{bmatrix}\times2$	concat 2×2, 384-d, LN $\begin{bmatrix}\text{window size}: 14\times14\\ \text{384-d (12 heads)}\end{bmatrix}\times6$	concat 2×2, 768-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{768-d (24 heads)}\end{bmatrix}\times2$
Swin-S, $W=14$	concat 4×4 , 96-d, LN $\begin{bmatrix}\text{window size}: 14\times14\\ \text{96-d (3 heads)}\end{bmatrix}\times2$	concat 2×2, 192-d, LN $\begin{bmatrix}\text{window size}: 14\times14\\ \text{192-d (6 heads)}\end{bmatrix}\times2$	concat 2×2, 384-d, LN $\begin{bmatrix}\text{window size}: 14\times14\\ \text{384-d (12 heads)}\end{bmatrix}\times18$	concat 2×2, 768-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{768-d (24 heads)}\end{bmatrix}\times2$
ViL-T, $W=7$	concat 4×4 , 96-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{96-d (3 heads)}\end{bmatrix}\times2$	concat 2×2, 192-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{192-d (3 heads)}\end{bmatrix}\times2$	concat 2×2, 384-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{384-d (6 heads)}\end{bmatrix}\times6$	concat 2×2, 768-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{768-d (12 heads)}\end{bmatrix}\times2$
CvT-T, $W=7$	concat 4×4 , 64-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{64-d (1 head)}\end{bmatrix}\times2$	concat 2×2, 192-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{192-d (3 heads)}\end{bmatrix}\times2$	concat 2×2, 384-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{384-d (6 heads)}\end{bmatrix}\times6$	concat 2×2, 768-d, LN $\begin{bmatrix}\text{window size}: 7\times7\\ \text{768-d (12 heads)}\end{bmatrix}\times2$

Table 6: Model configurations considered in our experiments.

Data source	Table 2 in DINO [6]	Table 3 in MLP-Mixer [53]	Table 1 in Swin [39]	Our runs
DeiT-S / $P=16$	1007		940.4	
DeiT-B / $P=16$	312		292.3	
DeiT-S / $P=8$	180			
DeiT-B / $P=8$	63			
ViT-B / $P=16$	312	861		
ViT-S / $P=16$	102	280		
ViT-H / $P=14$	32	87		
ViT-L / $P=7$	17	47†		
Swin-T / $W=7$	808		755.2	726.13
Swin-S / $W=7$	467		436.9	
Swin-B / $W=7$	297		278.1	
Swin-T / $W=14$	660			593.24
Swin-S / $W=14$	383			344.20
Swin-B / $W=14$	254			228.36
ViL-T / $W=7$	386			346.72
CvT-T / $W=7$	848			761.89

Table 7: Throughput estimate and standardization. All numbers in orange are estimated/converted, while numbers in blue are collected from the papers, and numbers in green are runs on our machines. All papers report the throughput of ViT-B or DeiT-B, which are essentially the same model. We use this fact to align the throughput reported in different papers. † This number is estimated via the statement in [12] that "reducing the patch size to 7×7 keeps the model size unchanged, but increases FLOPs to $\sim 6\times$". All numbers are standardized into throughput reported by [6].

public source of the dataset. Note that original UCF101 dataset is a video dataset. Here the middle frame of each video is extracted to form a classification dataset. There are 3 train/val splits in Tensorflow, we use the first one.

Automatic hyper-parameter tuning. We rigorously follow [47] to conduct training and evaluation for linear probe on the downstream datasets. We train a logistic regression classifier using scikit-learn's L-BFGS implementation, with maximum $1,000$ iterations, and report the corresponding metric for each dataset. We determine the L_2 regularization strength λ using a hyperparameter sweep on the validation sets over the range between 10^{-6} and 10^{6} , with 96 logarithmically spaced steps. To save compute required for the sweeps, we perform a parametric binary search that starts with $\lambda = [10^{-6}, 10^{-4}, 10^{-2}, 1, 10^{2}, 10^{4}, 10^{6}]$ and iteratively halves the interval around the peak until it reaches a resolution of 8 steps per decade. The hyperparameter sweeps are performed on a validation split of each dataset. For the datasets that contain a validation split in addition to a test split, we use

Method	#Param.	Im./s	Pre-train tasks	Linear	k-NN
DeiT	21	1007	$\mathcal{L}_V$	75.9	73.2
ResNet-50	23	1237	$\mathcal{L}_V$	75.3†	67.5†
			$\mathcal{L}_V$	75.0	69.3
			$\mathcal{L}_V+\mathcal{L}_R$	**75.7**	**71.2**
Swin	28	808	$\mathcal{L}_V$	77.1	73.7
			$\mathcal{L}_V+\mathcal{L}_R$	**77.6**	**75.4**
ViL	28	386	$\mathcal{L}_V$	77.3	73.9
			$\mathcal{L}_V+\mathcal{L}_R$	**77.5**	**74.5**
CvT	29	848	$\mathcal{L}_V$	77.6	74.8
			$\mathcal{L}_V+\mathcal{L}_R$	**78.5**	**76.7**

Table 8: Different sparse attentions in EsViT with and without $\mathcal{L}_R$. DeiT and ResNet-50 are shown as references. † indicates numbers reported in [6].

Dataset	Classes	Train size	Test size	Evaluation metric	Source link
Food-101	102	75,750	25,250	Accuracy	Tensorflow
CIFAR-10	10	50,000	10,000	Accuracy	TensorFlow
CIFAR-100	100	50,000	10,000	Accuracy	TensorFlow
SUN397	397	19,850	19,850	Accuracy	Tensorflow
Stanford Cars	196	8,144	8,041	Accuracy	Stanfold Cars
FGVC Aircraft (variants)	100	6,667	3,333	Mean-per-class	FGVC website
VOC2007 classification	20	5,011	4,952	11-point mAP	voc2007
Describable Textures	47	3,760	1,880	Accuracy	TensorFlow
Oxford-IIIT Pets	37	3,680	3,669	Mean-per-class	Oxford-IIIT Pet
Caltech-101	102	3,060	6084	Mean-per-class	TensorFlow
Oxford Flowers 102	102	2,040	6,149	Mean-per-class	TensorFlow
MNIST	10	60,000	10,000	Accuracy	TensorFlow
Facial Emotion Recog. 2013 *	8	32,298	3,589	Accuracy	Kaggle fer2013
STL10	10	5,000	8,000	Accuracy	TensorFlow
GTSRB *	43	26,728	12,630	Accuracy	GTSRB website
PatchCamelyon	2	294,912	32,768	Accuracy	TensorFlow
UCF101 *	101	9,537	3783	Accuracy	TensorFlow
Hateful Memes	2	8,500	500	ROC-AUC	FaceBook

Table 9: A suite of 18 datasets used in linear probe.* indicates dataset whose train/test size we obtained is slightly different from Table 9 in [47].

the provided validation set to perform the hyperparameter search, and for the datasets that do not provide a validation split or have not published labels for the test data, we split the training dataset to perform the hyperparameter search. For the final result, we combine the validation split back with the training split and report the performance on the unused split.

Detailed results. Only the last layer feature is considered for all models for simplicity, though adding features from more layers may potentially improve the results. Table 10 shows the results for architectures at a similar scale of ResNet-50 or Swin-T. The first two columns are numbers from [47]. CLIP with ResNet-50 is pre-trained on 400 million image-text pairs. Supervised ResNet-50 and Swin-T are pre-trained on ImageNet-1K, on which EsViT with Swin-T is pre-trained as well (Batch Size=512). EsViT outperforms its supervised counterpart, and is on par with the performance of CLIP in a similar image encoder architecture scale.

B.4 Pre-training datasets

We describe the statistics and training schedule on larger and less curated datasets in Table 11. The pre-training epochs are chosen so that the model is trained with a similar number of augmented views.

Methods	CLIP [47] ResNet-50	Supervised [47] ResNet-50	Supervised‡ ResNet-50	Supervised Swin-T	EsViT Swin-T
Food-101	86.4	71.3	71.3	77.4	80.0
CIFAR-10	88.7	91.8	91.8	94.0	95.3
CIFAR-100	70.3	74.5	74.5	77.5	82.2
SUN397	73.3	60.5	60.3	64.3	67.6
Stanford Cars	78.3	49.9	50.1	55.3	66.4
FGVC Aircraft (variants)	49.1	48.5	48.4	51.5	61.1
VOC2007 classification	87.1	83.8	83.6	84.2	85.5
Describable Textures	76.4	72.3	72.6	73.1	78.1
Oxford-IIIT Pets	88.2	92.4	92.1	93.3	92.8
Caltech-101	89.6	90.8	90.4	90.8	93.0
Oxford Flowers 102	96.1	90.8	91.1	91.5	97.4
MNIST	98.3	98.3	98.3	98.3	98.3
Facial Emotion Recog. 2013	64.2	54.9	55.9	55.1	59.3
STL10	97.2	96.4	97.0	97.9	98.9
GTSRB	82.4	70.6	75.7	72.9	84.3
PatchCamelyon	82.7	82.5	82.6	84.0	84.6
UCF101	81.6	71.2	72.1	79.0	81.1
Hateful Memes	65.7	56.5	49.9	51.2	52.0
Average	80.86	75.39	75.43	77.29	80.99

Table 10: The linear probe results on 18 datasets at the scale of ResNet-50/Swin-T. ‡ indicates the results reproduced by us, which verifies that our implementation pipeline is consistent with [47].

Name	Description	Size (#Images)	Epochs	Warmup
ImageNet-1K [15]	Images evenly distributed in 1K object concepts	1.2 million	300	10
WebVision-v1 [36]	Web images with 1K concept queries from ImageNet-1K	2.4 million	150	5
OpenImages-v4 [30]	Diverse/complex scenes with several objects for detection	7.5 million	50	2
ImageNet-22K [15]	Images distributed in 22K object concepts in a hierarchy	14.2 million	30	1

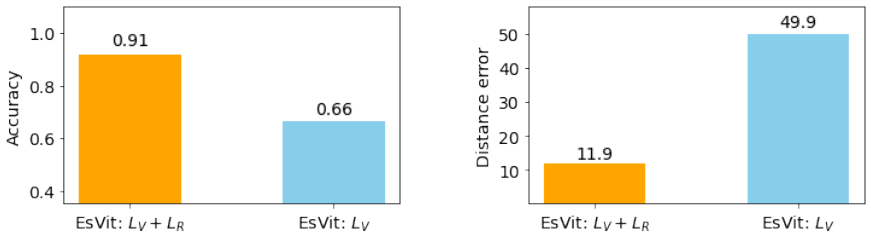
Table 11: Pre-train dataset statistics and training schedule.

B.5 Results on correspondence learning

We first quantitatively evaluate the correspondence learning results with 50K images in the ImageNet validation dataset. We create a simple evaluation dataset with mild augmentations. For a center-crop image, we apply `HorizontalFlip`, then `ColorJitter` and `RandomGrayscale` to create a new augmented view. In this way, ground-truth correspondences are created. Please see the 1st row of Figure 9 for one such example. The top-10 correspondences are used for evaluation. Two metrics are considered: (1) Accuracy measures the percentage of correctly matched region pairs, (2) distance error indicates the averaged ℓ_2 distance between the predicted matched region and ground-truth region (the value is 0 for perfect matching). The results are reported in Figure 8. DINO with monolithic Transformers shows surprisingly good performance on correspondence learning. The use of multi-stage Transformer architecture reduces this ability, shows a lack of good region correspondence. With $\mathcal{L}_R$, the region matching ability is significantly recovered.

In Figure 9, we visualize the correspondences for more images. Overall, DINO with monolithic Transformers is able to discover most salient correspondences of semantic meaning in the mild augmentation conditions, even without an implicit region matching loss in training. We believe this previously underestimated property is whole-noting, and has potentials to enable more applications. However, this desired property gets dilated when changing from monolithic to multi-stage Transformer architecture (from column 1 to column 2), then the proposed region level task can alleviate this issue (from column 2 to column 3).

To more specifically analyze the correspondences, we note the following results. The first row shows a simple case, where only images of left-to-right flipped views are presented. The ground-truth correspondences should be horizontal lines that link the two flipped regions. It reveals that the view-level pre-train task alone is insufficient to learn good correspondences for the multi-stage



(a) Accuracy (b) Distance error

Figure 8: Quantitative evaluation on correspondence learning on ImageNet validation set. $\mathcal{L}_R$ can significantly improve correspondence learning quality for multi-stage architectures. As a reference, DINO ($\mathcal{L}_V$ with monolithic Transformer architecture) achieves 0.95 accuracy and 2.49 distance error, which we believe is a strong evidence to identify the intriguing property of automatic correspondence learning.

Transformer architecture, while region matching task can alleviate this issue significantly. Similar observations are shown in row 3 and row 4.

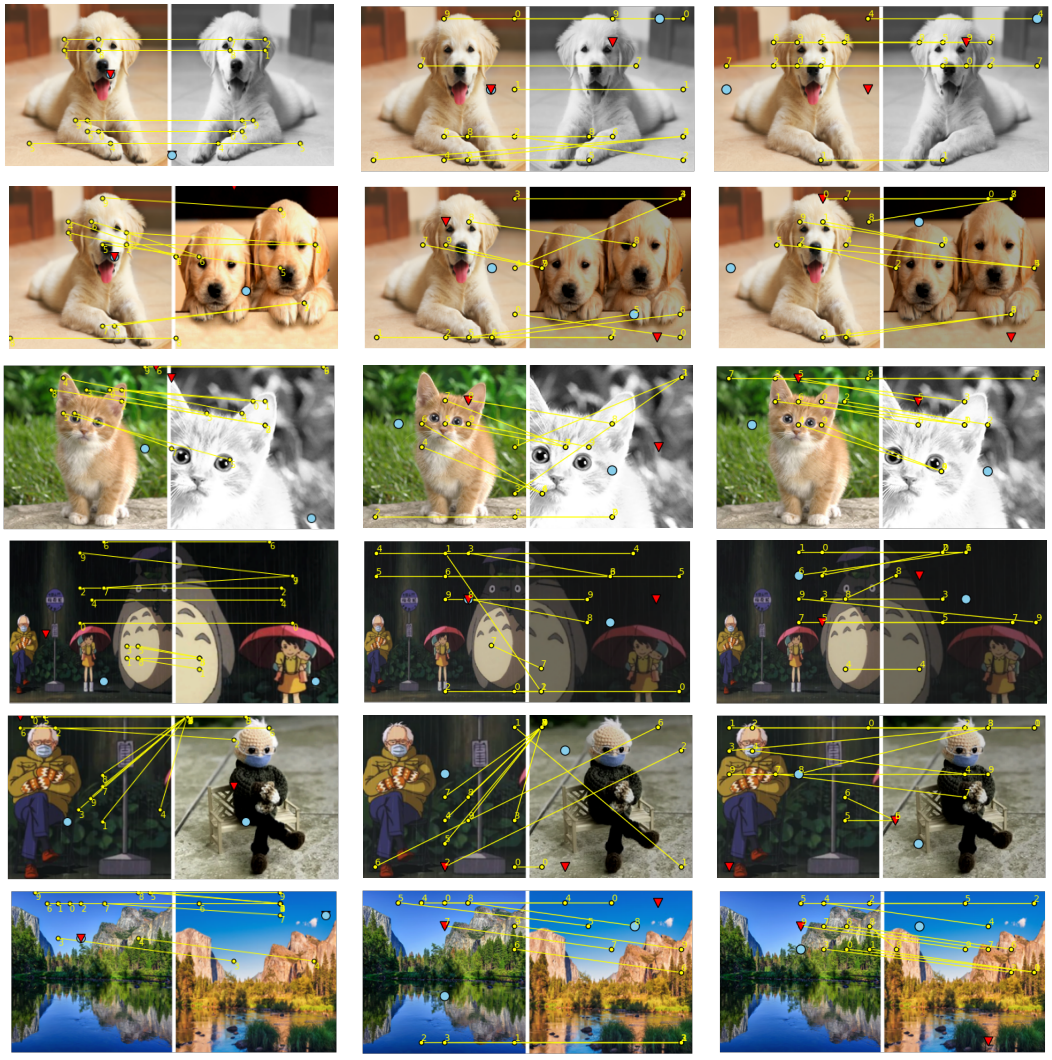
We further study more cases that requires real-world correspondences in row 2, row 5 and row 6. These views are not generated with data augmentation (as in model pre-training), but are often presented in more practical scenarios: one-to-many mappings, cartoon-to-toy, seasonal changing of the scene, respectively. The proposed region matching task can work particularly well in those cases.

B.6 More visualization results of attention maps

We visualize attention maps at the top layer in Figure 10, 11, 12. With a monolithic Transformer architecture, DINO can automatically identify the main foreground objects. Unfortunately, changing from monolithic to the multi-stage Transformer architecture (From left column to middle column), this property gets lost. There are more heads in the multi-stage architecture than monolithic architecture (24 heads vs 6 heads in this case) in the last year. A fair number of heads in EsViT shows redundant patterns, this issue can be reduced when the region-level matching task is added (From middle column to right column).

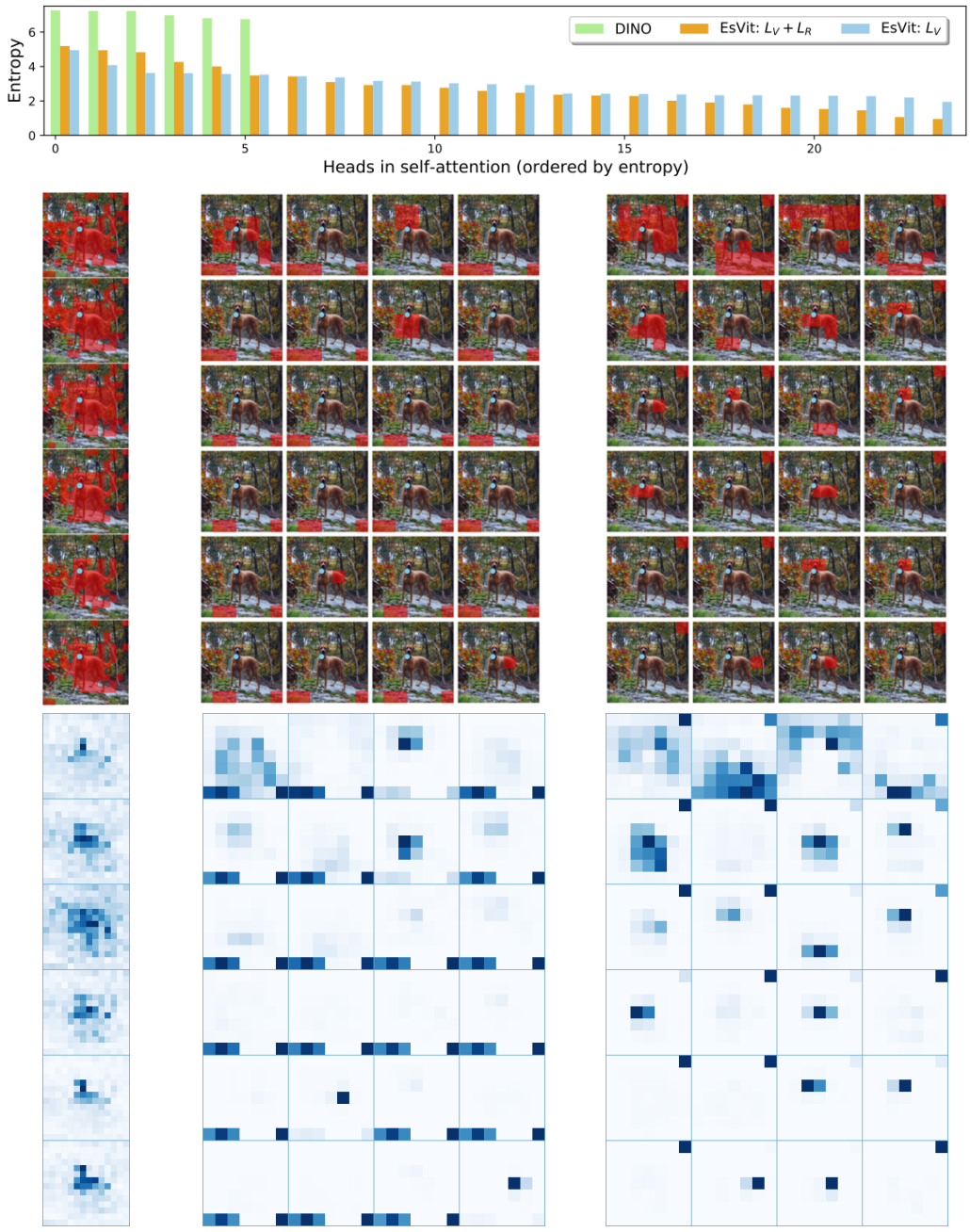
We observed that DINO with monolithic Transformer architecture only learns to attend the foreground objects, even when the query is a background region (see Figure 12). This is perhaps because DINO models are trained to learn view-level invariance, the main objects in the pre-train dataset ImageNet tend to be the principle factor that remains invariant across different augmented views. Hence, all backgrounds are ignored, regardless of the query positions. This is improved in EsViT with the region-level pre-train task, as the model is trained to match individual regions.

DINO shows high entropy values in all of 6 heads (perhaps a required condition to cover all regions of the main object). In EsViT, $\mathcal{L}_R$ plays an interesting role in modulating the entropy distributions among heads: it increases those with larger entropy values, while decreasing those with lower entropy values. In another word, it makes the attention patterns in different heads more diverse.



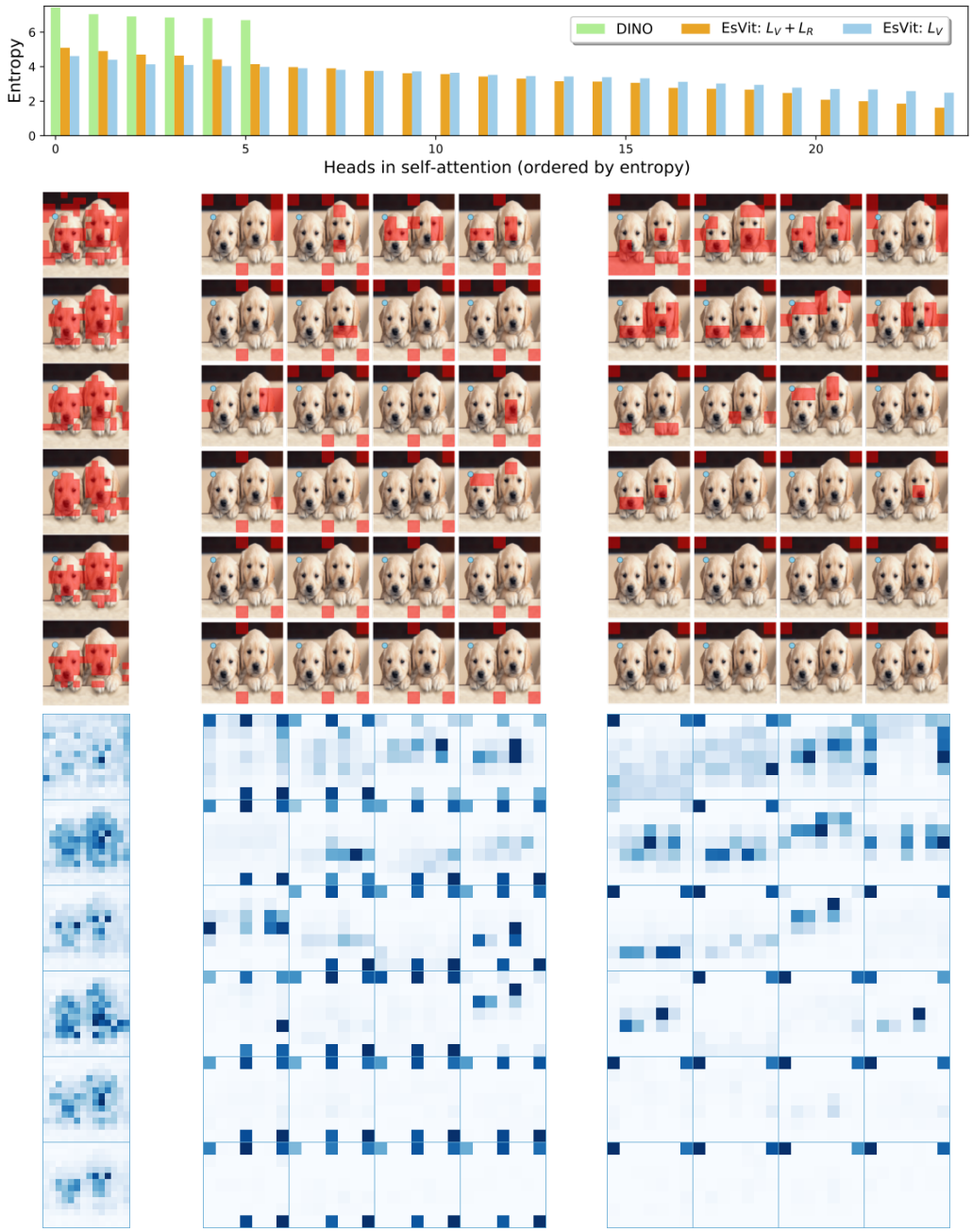
(a) DINO: DeiT-S (b) EsViT: $\mathcal{L}_V$ (c) EsViT: $\mathcal{L}_V + \mathcal{L}_R$

Figure 9: The learned correspondences. **Yellow** lines are the top-10 correspondences between two views, where the numbers indicates the rankings of similarity scores, yellow dots with the same number are paired. The **blue** dot and **red** triangle indicates the most similar local regions that correspond to the global feature of the view itself and the other view, respectively. Please zoom in for detailed correspondence mappings.



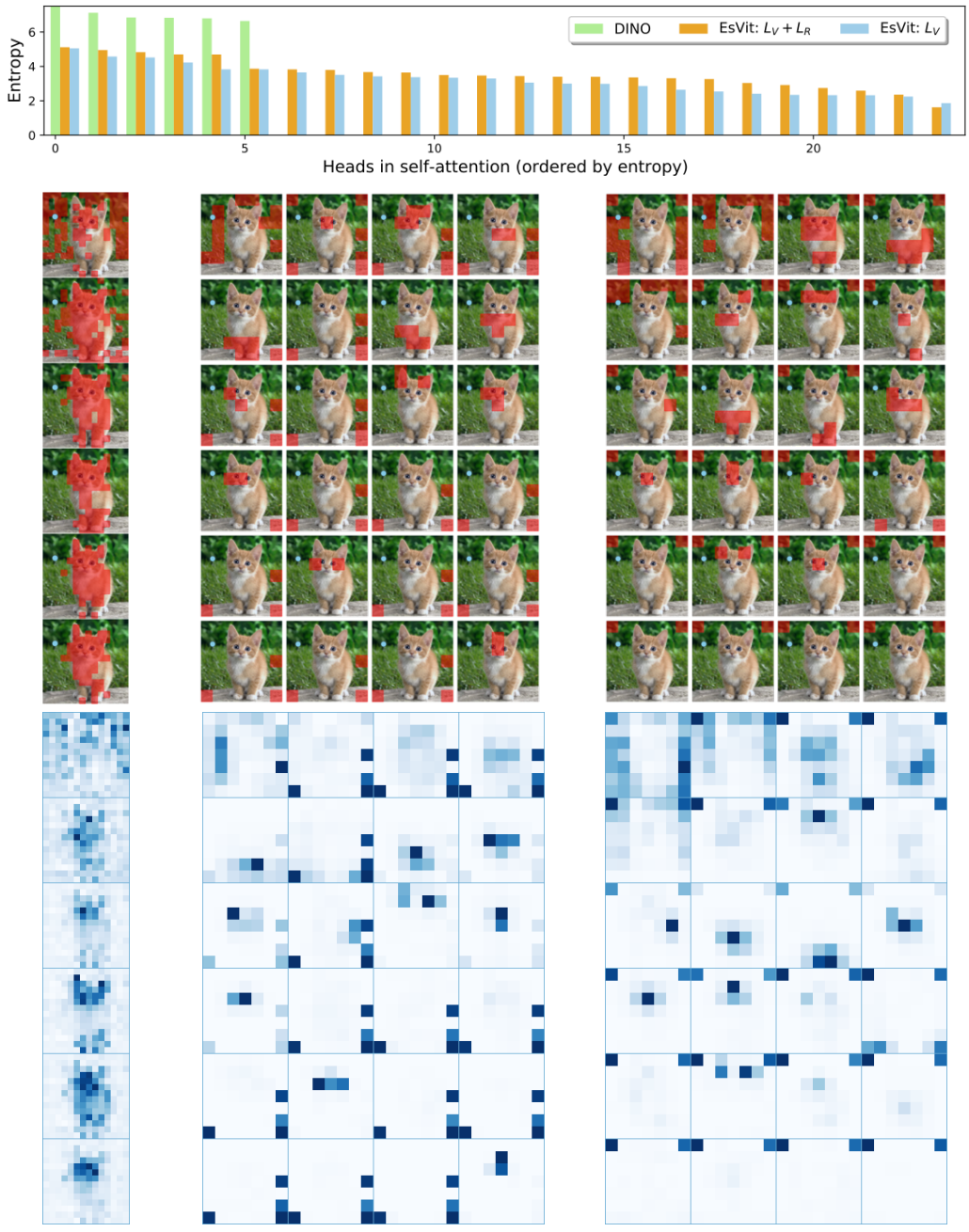
(a) DINO: DeiT-S (b) EsViT: $\mathcal{L}_V$ (c) EsViT: $\mathcal{L}_V + \mathcal{L}_R$

Figure 10: The learned attention maps for all heads at the top layer, ranked by the entropy of softmax probability. Query is the blue dot in the top-left of the image. Top: Entropy of each heads. Middle: top 60% probability mass. Bottom: full attention maps. $\mathcal{L}_R$ shows more attention patterns than $\mathcal{L}_V$ only.



(a) DINO: DeiT-S (b) EsViT: $\mathcal{L}_V$ (c) EsViT: $\mathcal{L}_V + \mathcal{L}_R$

Figure 11: The learned attention maps for all heads at the top layer, ranked by the entropy of softmax probability. Query is the blue dot in the center of the image. Top: Entropy of each heads. Middle: top 60% probability mass. Bottom: full attention maps. $\mathcal{L}_R$ shows more attention patterns than $\mathcal{L}_V$ only.



(a) DINO: DeiT-S (b) EsViT: $\mathcal{L}_V$ (c) EsViT: $\mathcal{L}_V + \mathcal{L}_R$

Figure 12: The learned attention maps for all heads at the top layer, ranked by the entropy of softmax probability. Query is the blue dot in the top-left of the image. Top: Entropy of each heads. Middle: top 60% probability mass. Bottom: full attention maps. DINO mainly attends the main object even when the query is a background region.