Deep ViT Features as Dense Visual Descriptors

In this document we provide additional implementation details, and complementary results to those appearing in the paper.

1 Implementation details

In all our experiments (unless specified otherwise) we use `dino_vits8` model from the official DINO Github repository [1, 2], with `stride=4` (see Resolution Increase paragraph). We use `timm` repository [17] for ViT architecture and supervised weights, and [1] for DINO-ViT weights.

PCA (Fig. 2 Right). We used (a) `dino_vits16` and `vit_small_patch16_224` models with `stride=8` and (b) `ResNet-50` trained in a supervised manner and with DINO. All models were trained on ImageNet. We visualize ViT features from layers 2, 5, 8, 10 and ResNet activations from the end of each block. We resized the input images to size 224×224.

t-SNE (Fig. 3). We used `dino_vits8` and `vit_small_patch16_224` for DINO-ViT and Supervised ViT respectively, both with `stride=8`. We use the implementation of opentsne [13], using cosine similarity to compute affinities between descriptors. For Supervised ViT t-sne we use perplexity 500 and exaggeration 2, for DINO-ViT we use perplexity 300. To keep the visualization from being too crowded, we show every 10^{th} descriptor from each image.

Resolution Increase (§4). We increase the resolution of ViT features maps by altering the phase of patch preparation. Instead of taking non-overlapping patches we take *overlapping patches*. In practice, the separation to patches and linear embedding is done by passing the image through a single convolution layer, with stride that equals the patch size and number of out channels as the embedding dimension. We alter the *stride* of this convolution layer to achieve overlapping patches. For example, using `stride=8` for a ViT trained with patch size 16 will increase the ViT feature's resolution times two. We assume the input size $\{H_{in}, W_{in}\}$ is divided by the patch size without remainder. If that is not the case, we remove the remainder pixels from the image. The output size is given by:

$$H_{\text{out}} = \frac{H_{\text{in}} - \text{patch\_size}}{\text{stride}} + 1$$
$$W_{\text{out}} = \frac{W_{\text{in}} - \text{patch\_size}}{\text{stride}} + 1$$

1.1 Part Co-segmentation

Part Co-segmentation parameters (§5.1). We extracted the keys from the last layer (11^{th} starting from 0) and concatenated all the heads to receive a descriptor for each patch. We used the FAISS library [9, 10] for computing k-means. For the first stage (co-segmentation) our elbow coefficient is 0.975, saliency threshold is 0.065, majority percentage is 75%. We sampled every 100^{th} descriptor before applying k-means. For CelebA [12] evaluation (see Sec. 2) we choose the salient segments based if there average distance from the image center is under 0.2, and if their compactness is above 0.5.

Part Co-Segmentation of Image Pairs (Fig. 6). We present our part co-segmentation results in an extreme setting – operating on two images under significant variations of quantity, background clutter, pose, scale and appearance. Some of our sets belong to domains where training data is scarce. We compensate for the low number of images by applying flip and random-crop augmentations to the input images before applying the part co-segmentation pipeline. The random crops are of size 95% of the original images.

For images including much background clutter, we also introduce three clustering stages instead of two – one for fg/bg separation, one for removing uncommon foreground objects and one for part segmentation. This extreme setting is sensitive to hyper-parameters, but we found using the same hyperparameters as the original part co-segmentation pipeline, using 40 random-crop augmentations, and elbow coefficient of 0.94 for the second clustering stage works well for most cases.

1.2 Co-segmentation

Co-segmentation parameters (§5.2). We use the same parameters as stated in the first stage of part co-segmentation (see Sec. 1.1).

Global Outlier Filtering One of the challenges in the Internet300 [14] dataset is handling images that do not contain the common object at all. We term these *global outlier images*, and filter them automatically before applying the co-segmentation pipeline using the descriptor of the [CLS] token. We compute the average of all the [CLS] descriptors on the entire set of images, and reject images that have cosine similarity lower than 0.7 from the average descriptor.

Saliency Baselines (Tab. 3). For DINO-ViT baseline, we used keys from the 11^{th} layer and a saliency threshold of 0.04. For supervised ViT baseline, we used keys from the 9^{th} layer because they exhibited better part separation than the 11^{th} layer, giving supervised ViT a fair chance. We used `vit_small_patch16_224` with `stride=4` and saliency threshold of 0.01. DINO and supervised ResNet-50 weights are from DINO and `timm` repositories respectively. In PASCAL-Co ablations for ResNet-50 we replace the last three strides with dilation to receive high resolution feature maps, as if features were computed at `stride=4` of the input resolution. All models are trained on ImageNet data.

1.3 Semantic Correspondences

Additional Implementation Details and parameters (§5.3). Given two images, the task is to find k correspondences between the images, where k is given by the user. We find Best Buddy Pairs [5] (BPPs) between binned descriptors from both images as correspondence candidates. We use log-binning with three heirarchies - 17 bins in total (see Fig. 1). We utilize the same DINO-ViT saliency maps for co-segmentation, and eliminate BBPs where saliency is below 0.05. Then, we cluster the the pairs into k clusters using a concatenation of their descriptors. Each cluster outputs its most salient BPP as the final result. For compatibility with NBB we resize the images to size 224×224.

2 Additional Results

DAVIS Label Propagation. We empirically show the usefulness of test-time resolution increase by applying it to one of the applications shown in [1] - using pre-trained DINO features for DAVIS label propagation. We used a `dino_vits8` model with `stride=4`. Table 1 exhibits a significant improvement in results when using our alteration, and exhibit results even better than `dino_vitb8`. These results further demonstrate the effectiveness of high-granularity features as local descriptors.

Part Co-segmentation (§5.1) In Tab. 2 we measure landmark regression on a subset of the CelebA [12] as mentioned in [7], using $k = 4$ and $k = 8$ parts. Our results are superior to the other methods, both supervised and unsupervised. Surprisingly, our lightweight method with *supervised* ViT features performs impressively as well. These results strengthen our observations that DINO-ViT features are shared across semantically common parts.

Architecture	$(\mathcal{J}\&\mathcal{F})_m$	$\mathcal{J}_m$	$\mathcal{F}_m$
`dino_vits16` (stride=16)	61.8	60.2	63.4
`dino_vitb16` (stride=16)	62.3	60.7	63.9
`dino_vits8` (stride=8)	69.9	66.6	73.1
`dino_vitb8` (stride=8)	71.4	67.9	74.9
Ours	**72.2**	**67.9**	**76.5**

Table 1: **DAVIS 2017 Video Object Segmentation.** Using our inference-time resolution increase surpasses previous methods, and demonstrates the usability of high-granularity features. The first four rows present results using the DINO tokens from different backbones (as done in [1]). Our result uses the `dino_vits8` backbone with stride=4.

Method	$k = 4$	$k = 8$
ULD [16, 18]	-	40.82
DFF [4]	-	31.30
IMM [8]	19.42	**8.74**
SCOPS [7] (w/o sal.)	46.62	22.11
SCOPS [7] (with sal.)	21.76	15.01
Liu [11] et al.	15.39	12.26
Ours (DINO-ViT)	**11.36**	<u>10.74</u>
Ours (Sup. ViT)	<u>12.83</u>	12.74

Table 2: **Landmark regression on Celeba dataset.** Our results with DINO-ViT backbone are superior to all methods. First three methods [16, 18, 4] are designed specifically for landmark discovery, IMM [8] specializes on faces.

2.1 Co-segmentation

We ablate our co-segmetation method on all datasets using different ViT backbones trained with DINO (see Tab. 3). For a fair comparison, for each backbone we take all the attention heads when creating our saliency map within the co-segmentation pipeline. In addition, we use the same parameters (e.g. stride, saliency threshold) for all runs. Using the same stride means all backbones have *the same spatial granularity*. It seems our method performs quite similarly for the different backbones, while favoring those originally trained with a smaller patch size (e.g. `dino_vits8` and `dino_vitb8`). This empirically validates our observations *across DINO-ViT backbones*, and shows they are all suitable for extracting deep local features when used with proper stride.

We also ablate our co-segmentation method using different strides (see Tab. 3). It seems that using our resolution increasing method by setting `stride=4` causes an improvement in performance. Our method also benefits from using a subset of the attention heads (marked by Ours stride=4) than all the attention heads (marked by `dino_vits8`).

2.2 Semantic Correspondences

In Tab. 4 we ablate our keypoint matching method with different strides and different backbones. We use a similar setting to the co-segmentation ablation, in which all different backbones are used with the same `stride=4`, hence have *the same spatial granularity*. Evidently, using a lower stride causes an increase in performance. When using the same stride for different backbones, the performance is similar. Interestingly, our *lightweight zero-shot* method surpasses the *supervised* baseline CATs [3] in roughly a fourth of the categories. These results further validate our observations hold *across different DINO-ViT backbones* when used with proper stride.

Method	MSRC [15]		Internet300 [14]		PASCAL -VOC [6]		PASCAL -CO	
	$\mathcal{J}_m$	$\mathcal{P}_m$	$\mathcal{J}_m$	$\mathcal{P}_m$	$\mathcal{J}_m$	$\mathcal{P}_m$	$\mathcal{J}_m$	$\mathcal{P}_m$
`dino_vits8` (stride=4)	81.6	94.3	75.1	93.0	55.7	85.8	74.2	92.8
`dino_vits16` (stride=4)	72.3	88.9	49.3	76.1	46.3	79.7	70.1	90.8
`dino_vitb8` (stride=4)	70.8	86.9	69.5	89.9	56.9	87.9	71.2	91.0
`dino_vitb16` (stride=4)	69.6	86.5	65.1	89.2	48.1	81.9	71.6	91.6
Ours (stride=8)	83.5	95.0	73.8	92.6	52.7	85.3	71.5	91.7
Ours (stride=4)	**86.7**	**96.5**	**79.5**	**94.6**	**60.7**	**88.2**	**79.5**	**94.7**

Table 3: **Co-segmentation ablation:** We report mean Jaccard index $\mathcal{J}_m$ and precision $\mathcal{P}_m$ over all sets in each dataset. Different ViT backbones perform comparably, with a slight favor to those trained with a small patch size. Using a smaller stride causes an increase in performance.

category	**NBB**	**CATs**	**Stride 4**	**Stride 8**	`dino_vits8`	`dino_vits16`	`dino_vitb8`	`dino_vitb16`
aeroplane	0.44	0.57	**0.69**	0.64	**0.69**	0.66	0.60	0.68
bicycle	0.28	0.48	**0.50**	0.49	**0.50**	0.42	0.49	0.41
bird	0.67	**0.89**	0.82	0.78	0.82	0.82	0.78	0.79
boat	0.12	0.39	0.47	0.43	0.46	0.41	0.40	**0.54**
bottle	0.17	**0.44**	0.37	0.33	0.37	0.33	0.36	0.34
bus	0.20	**0.63**	0.42	0.36	0.42	0.40	0.40	0.43
car	0.28	**0.60**	0.53	0.52	0.53	0.50	0.51	0.53
cat	0.30	0.65	0.66	0.62	0.66	0.66	0.66	**0.69**
chair	0.20	0.34	**0.45**	0.39	**0.45**	0.41	0.44	**0.45**
cow	0.29	0.73	**0.75**	0.63	**0.75**	0.61	0.66	0.68
dog	0.37	**0.65**	**0.65**	0.63	**0.65**	0.55	0.55	0.61
horse	0.13	**0.60**	0.46	0.38	0.46	0.47	0.44	0.46
motorbike	0.51	**0.80**	0.69	0.68	0.69	0.58	0.58	0.61
person	0.14	**0.66**	0.48	0.38	0.47	0.45	0.50	0.59
pottedplant	0.15	0.48	0.44	0.44	0.43	0.44	0.44	**0.49**
sheep	0.11	**0.70**	0.65	0.62	0.64	**0.70**	0.65	0.57
train	0.23	**0.83**	0.54	0.45	0.54	0.55	0.61	0.68
tvmonitor	0.26	0.62	0.59	0.55	0.59	0.45	0.54	**0.68**
all	0.27	**0.61**	0.56	0.52	0.56	0.53	0.54	0.56

Table 4: **Spair71k keypoint matching.** We report PCK ($\alpha = 0.1$) for every category. Our method benefits from working with a lower stride.

References

[1] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. ICCV (2021) 1, 2, 3

[2] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers: Github repository. https://github.com/facebookresearch/dino (2021) 1

[3] Cho, S., Hong, S., Jeon, S., Lee, Y., Sohn, K., Kim, S.: Semantic correspondence with transformers. NeurIPS (2021) 3

[4] Collins, E., Achanta, R., Susstrunk, S.: Deep feature factorization for concept discovery. In: The European Conference on Computer Vision (ECCV) (2018) 3

[5] Dekel, T., Oron, S., Rubinstein, M., Avidan, S., Freeman, W.T.: Best-buddies similarity for robust template matching. CVPR (2015) 2

[6] Everingham, M., Eslami, S.M.A., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. IJCV (2015) 4

[7] Hung, W.C., Jampani, V., Liu, S., Molchanov, P., Yang, M.H., Kautz, J.: Scops: Self-supervised co-part segmentation. CVPR (2019) 2, 3

[8] Jakab, T., Gupta, A., Bilen, H., Vedaldi, A.: Unsupervised learning of object landmarks through conditional image generation. NeurIPS (2018) 3

[9] Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. arXiv preprint arXiv:1702.08734 (2017) 1

[10] Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus: Github repository. https://github.com/facebookresearch/faiss (2017) 1

[11] Liu, S., Zhang, L., Yang, X., Su, H., Zhu, J.: Unsupervised part segmentation through disentangling appearance and shape. CVPR (2021) 3

[12] Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. ICCV (2015) 1, 2

[13] Poličar, P.G., Stražar, M., Zupan, B.: opentsne: a modular python library for t-sne dimensionality reduction and embedding. bioRxiv (2019) 1

[14] Rubinstein, M., Joulin, A., Kopf, J., Liu, C.: Unsupervised joint object discovery and segmentation in internet images. CVPR (2013) 2, 4

[15] Shotton, J., Winn, J., Rother, C., Criminisi, A.: Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. ECCV (2006) 4

[16] Thewlis, J., Bilen, H., Vedaldi, A.: Unsupervised learning of object landmarks by factorized spatial embeddings. ICCV (2017) 3

[17] Wightman, R.: Pytorch image models. GitHub repository (2019) 1

[18] Zhang, Y., Guo, Y., Jin, Y., Luo, Y., He, Z., Lee, H.: Unsupervised discovery of object landmarks as structural representations. CVPR (2018) 3