LiFT: A Surprisingly Simple Lightweight Feature Transform for Dense ViT Descriptors

Saksham Suri* Matthew Walmer* Kamal Gupta Abhinav Shrivastava

University of Maryland, College Park

Abstract. We present a simple self-supervised method to enhance the performance of ViT features for dense downstream tasks. Our Lightweight Feature Transform (LiFT) is a straightforward and compact postprocessing network that can be applied to enhance the features of any pre-trained ViT backbone. LiFT is fast and easy to train with a self-supervised objective, and it boosts the density of ViT features for minimal extra inference cost. Furthermore, we demonstrate that LiFT can be applied with approaches that use additional task-specific downstream modules, as we integrate LiFT with ViTDet for COCO detection and segmentation. Despite the simplicity of LiFT, we find that it is not simply learning a more complex version of bilinear interpolation. Instead, our LiFT training protocol leads to several desirable emergent properties that benefit ViT features in dense downstream tasks. This includes greater scale invariance for features, and better object boundary maps. By simply training LiFT for a few epochs, we show improved performance on keypoint correspondence, detection, segmentation, and object discovery tasks. Overall, LiFT provides an easy way to unlock the benefits of denser feature arrays for a fraction of the computational cost. For more details, refer to our project page.

Keywords: Self-supervised Learning · ViTs · Feature Densification

1 Introduction

In recent years, Vision Transformers (ViTs) [16] have emerged as preferred architectures for many image and video recognition tasks in the Computer Vision community. They also represent a major design shift compared with the well-explored Convolutional Neural Networks (CNNs). ViTs typically convert images into a very coarse grid of image patches (or tokens) before applying transformer layers. This allows ViTs to learn increasingly powerful patch-wise representations in successive layers [46]. The expressive power of ViTs stems from their wide receptive field throughout all layers made possible by multi-headed self-attention operations [44]. The downside of this design is that despite being able to learn powerful representations, ViTs often lack spatial granularity in their features due to the low resolution of the token/patch grid. This hinders their off-the-shelf application to dense and local tasks such as object detection, segmentation, and keypoint correspondence. Increasing the feature resolution of a ViT directly using a larger image size or smaller patch size leads to an increased number of

* Equal contributors.

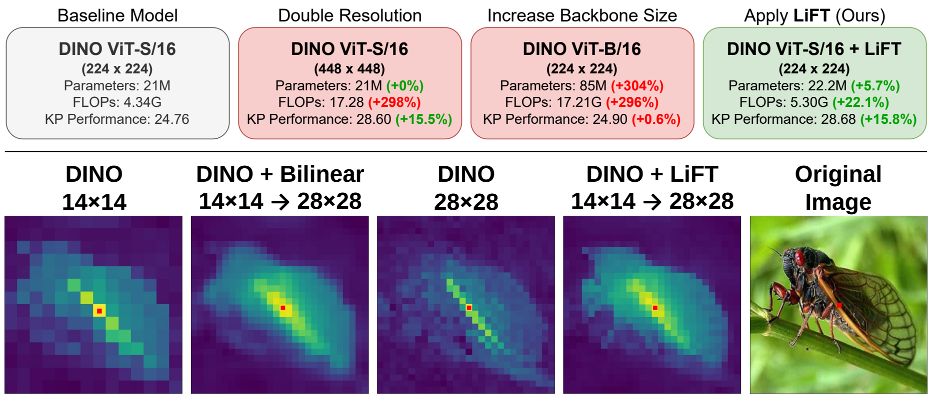


Fig. 1: (Top) Increasing the backbone size or doubling the input resolution can boost the effectiveness of self-supervised ViT features for dense tasks like keypoint (KP) correspondence. However, both of these options come at a significant cost in terms of parameter count, inference cost, or both. We present **LiFT**, a surprisingly simple **Li**ghtweight **F**eature **T**ransform that unlocks the benefits of dense self-supervised ViT representations for minimal extra cost. **(Bottom)** LiFT also has useful emergent properties, such as yielding cleaner object boundaries in feature similarity maps.

patches. Self-attention, being a quadratic operation, grows in memory consumption as $\mathcal{O}(N^2)$ where N is the number of patches in the image. Prior works have proposed alterations to the ViT architecture to make it better suited for dense tasks, but their methods either involve expensive carefully designed training, task-specific loss functions or heuristics, or high inference costs [2, 52, 55].

In this work, we propose a simple **Li**ghtweight **F**eature **T**ransform or **LiFT** to generate dense ViT features that provide significant performance gains in downstream tasks such as detection, segmentation, keypoint correspondence, and object discovery. LiFT can unlock the benefits of dense feature representations for a fraction of the computational cost compared with other approaches. As illustrated in Figure 2, our proposed method fuses the coarse high-level information of ViT features with convolution-based image features derived from the original image to generate higher-density feature maps without incurring the high computational cost of extra tokens. We show that this approach does not require any complex training recipe and, once trained on a general purpose dataset, generalizes well to multiple downstream tasks. Our approach can be trained with a simple self-supervised loss and generalizes to input image resolutions not seen during training. LiFT can be readily plugged on top of any ViT backbone to enhance its features, and it can also be integrated into pipelines that use additional task-specific downstream modules, like the Mask-RCNN head [25] used by ViTDet [28]. Additionally, we show that we can apply LiFT in a recursive manner to increase feature resolution even further.

We demonstrate the effectiveness of LiFT quantitatively on 'local' tasks, which require features computed at precise locations, as well as on 'dense' tasks, which require features computed for the entire image. Specifically, we present results for LiFT applied to SPair-71k Keypoint Correspondence [32], COCO

Detection and Segmentation [29], DAVIS Video Segmentation [33], and Unsupervised Object Discovery on Pascal VOC 2007 [18], Pascal VOC 2012 [19] and COCO20K [30]. For all of these tasks, LiFT is able to meet or exceed the performance of prior works for a fraction of the computational cost. As an example, in Figure 1 we compare three options for boosting performance in SPair-71k Keypoint Correspondence. LiFT provides a significant performance gain while increasing the total parameter count of the network by a mere 5.7%. This is compared to the 304% parameter count increase incurred by the step up from ViT-S/16 to ViT-B/16. Increasing the input resolution is a trivially easy way to boost the feature density, and it also gives improved performance. However, it increases the total inference FLOPs by almost 300% while LiFT only increases the cost by 22.1%, giving a far superior compute cost *vs*. performance trade off.

Despite the simplicity of our LiFT approach, we show that it is not just learning a more complex version of bilinear upsampling. Instead, we demonstrate that LiFT has several desirable emergent properties that enhance ViT features to make them better suited for dense tasks. We find that LiFT improves the scale invariance of ViT features, as measured using Centered Kernel Alignment (CKA) [12, 27]. We also qualitatively show that LiFT yields better object boundary maps when computing feature similarity maps. Overall, LiFT represents an orthogonal avenue of improvement compared to prior dense ViT feature extraction strategies, and it can be combined with past methods to further advance self-supervised performance on dense prediction tasks. In summary, our contributions are as follows:

– We propose **LiFT**, a **Li**ghtweight **F**eature **T**ransform that boosts the performance of existing ViT features on dense and local downstream tasks using a simple, quick training and inference strategy.
– We show that LiFT boosts the performance of self-supervised ViT features for detection, segmentation, keypoint correspondence, and object discovery tasks.
– We demonstrate the adaptability of LiFT for any ViT backbone by showing improvements with DINO [5], MoCo [8] and Supervised ViT features. Additionally, LiFT even works on image resolutions not used during training.
– We show that LiFT features have desirable emergent properties like improved scale invariance and better feature alignment with object boundaries.

2 Related Work

2.1 Vision Transformers

Vision Transformers (ViTs) [16] have gained wide popularity as general-purpose models for multiple computer vision tasks such as image classification [1, 16, 43], object detection [28, 31, 54], segmentation [36, 40], video classification [3, 4], and more. Many of these methods adapt ViTs to different tasks by making suitable changes to the output heads [28, 51]. There have also been multiple variants of ViT like Swin [31], MViT [20], and PVT [47] which incorporate hierarchy and multiscale learning. Additionally, there are some works which try to bridge the gap between transformers and CNNs by incorporating convolutions into ViT architectures [17, 50]. In this work, we focus on improving traditional or "Plain" ViT backbones, as they are the most general and widely adopted form of ViT. These models learn powerful representations, but they suffer in terms of feature resolution, an issue which we aim to address with LiFT.

2.2 Supervision Strategies for ViTs

Many works have proposed self-supervised tasks to learn meaningful representations with ViTs. These approaches do not require labels and can make use of large amounts of unlabeled data. These self-supervised models have powerful off-the-shelf features and can act as good pre-trained initializations for finetuning on downstream tasks. Among the most popular are approaches like Momentum Contrast methods (MoCo) [7, 9, 24] which enforce consistency between features for different augmentations of the same image. DINO [5] also utilizes a similar self-supervised approach, but uses different crops along with other augmentations. MAE [23] learns to predict masked-out image regions, and there are works like CLIP [34] which match text and image embeddings to learn semantically rich features without direct label-level supervision. Prior works [21, 38, 46] have shown that these differences in pre-training lead to significant differences in the properties of the learned features. We show that LiFT can improve the quality and usefulness of ViT features for a range of different pretraining methods.

2.3 Feature Densification

Works like [2] and [46] show the general benefits of dense feature maps for local tasks, and multiple works have been proposed to extract denser feature arrays from pretrained networks. [42] proposes a GAN-based approach for CNN feature densification which requires careful training and a mixture of adversarial and focal loss. In comparison, our LiFT approach is easy to train through a simple self-supervised objective, and it does not require a discriminator module or careful loss tuning. ViT-Adapter [10] applies ViTs to dense tasks through the use of finetunable side-networks for feature pyramid extraction. Like ViT-Adapter, LiFT aims to enhance "Plain" ViT features for dense tasks, however, the ViT-Adapter method is not task-agnostic, and it is trained in a fully supervised way with full detection and segmentation labels for the downstream dataset. In contrast, LiFT is a task-agnostic general enhancement for ViT features, and it is trained with a completely self-supervised objective. LiFT is also faster and easier to train than ViT-Adapter, as it does not require passing gradients through the ViT backbone, and it is also $\sim$4.8$\times$ smaller than the similar ViT-Adapter.

Other works, like [6] and [53], have applied student/teacher distillation with feature super-resolution to improve CNN classification performance on low-resolution images. While we also perform feature super-resolution, the aim of our work is not to distill a student network, but rather we aim to generate densified features while completely avoiding finetuning the ViT backbone. [2]

proposes a simpler technique to increase the density of ViT features by reducing the stride during initial image patch extraction. This does not require any training, but also becomes computationally expensive as the number of tokens increase quadratically requiring more GPU memory and FLOPs. Our LiFT approach can be thought of as a "shortcut" to achieve the benefits of denser features for a fraction of the computation cost of increased tokens.

2.4 Finetuning ViTs for Dense Tasks

The prior works most similar to LiFT are those that apply self-supervised finetuning to pretrained ViTs to improve their features for dense tasks. SelfPatch [55] and Leopart [52] both use pretrained DINO models as a starting point and improve their patch-level representations using dense self-supervised tasks. SelfPatch learns by enforcing similarity to neighboring patch features, and Leopart uses spatially dense clustering to enforce similarity within clusters to learn better part representations. These approaches finetune the full backbone with specialized training strategies and losses, which can still be expensive to train and not easily extendable to other backbones. In contrast, our approach does not require backbone finetuning at all, and instead trains a lightweight post-processing module. LiFT can be easily trained on ImageNet [14] in a self-supervised manner in very few epochs and afterwards it can generalize to multiple downstream tasks.

3 Method

3.1 ViT Background

Consider a ViT that takes an image with dimensions $H \times W \times 3$ as input and outputs feature descriptors at the resolution $\frac{H}{P} \times \frac{W}{P}$, where P is the patch size. P is usually 8 or 16 depending on the ViT variant. We typically assume that the patch-extraction stride length, S, is equal to P, though this can be altered, as proposed by [2]. For now we will assume that $S = P$. Our goal is to transform the coarse, low-resolution features of a pretrained ViT into dense feature descriptors without having to re-train or finetune the ViT. A naïve way to achieve this transformation could be to scale up the input image to $CH \times CW$ resolution to achieve a feature grid that is C times larger along both dimensions. This approach can result in a significant increase in memory consumption and can be prohibitively expensive since the memory of the ViT scales with $\mathcal{O}(H^2W^2)$. Another option could be to upscale the features directly by a factor of C using bilinear-interpolation. This approach computes sub-optimal pixel-level features as it simply bilinearly redistributes the features between the centers of patches. Such an approach fails to take advantage of the information that is readily available in the original image space. LiFT takes advantage of this information.

3.2 Lightweight Feature Transform (LiFT)

Our proposed approach, LiFT, builds on the hypothesis that, even though the ViT feature descriptors have a low spatial resolution, their high dimensionality

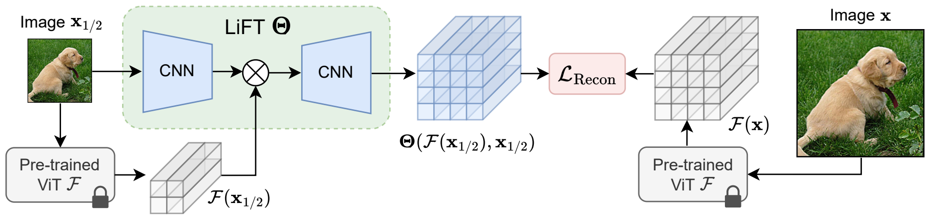


Fig. 2: Illustration of **LiFT**, our proposed **Li**ghtweight **F**eature **T**ransform for generating dense ViT descriptors. The frozen ViT backbone is used to extract features for both low- and high-resolution images. The low-resolution image and its corresponding features are passed through LiFT, which generates a dense version of the features. The LiFT Block first encodes fine-resolution image features using a small CNN. It then combines the CNN features with the ViT features at multiple phases in an upsampling CNN, which outputs dense features. The LiFT block is trained using a self-supervised reconstruction error with the corresponding high-resolution features.

allows them to store rich information about the image structure. This hypothesis is further supported by works on internal learning of images [22,39,56]. However, unlike internal learning, we propose to train a general-purpose, lightweight upsampling network using only self-supervision to double the resolution of feature descriptors obtained from a frozen, pretrained ViT. Furthermore, we can gain additional fine-level information directly from the original image at the same resolution used to generate the ViT features. We fuse these two information sources to create our final LiFT module, as illustrated in Figure 2.

Given ViT descriptors of size $\frac{H}{P} \times \frac{W}{P}$, a single LiFT expansion block scales them to $\frac{2H}{P} \times \frac{2W}{P}$ in a single forward pass. Our LiFT block is built following a U-Net-style structure [37] with skip connections, where semantically rich but coarse ViT features are combined with shallow but dense image features derived from a second input with the original image. The image input is processed through a series of convolution blocks and the resulting features are concatenated to the ViT features. Then, a single transpose convolution block is applied to generate the upscaled semantically rich features. Thanks to its fully convolutional nature, the LiFT block can be applied to any image size. One can even apply a LiFT block multiple times to further upscale the features, as shown in Section 6.4.

3.3 Training Objective

Given an image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$ and a pretrained, frozen ViT model $\mathcal{F}$ of stride P, we extract its features from the last layer such that $\mathcal{F}(\mathbf{x}) \in \mathbb{R}^{\frac{H}{P} \times \frac{W}{P} \times D}$, where D is the feature dimension. The LiFT block $\boldsymbol{\Theta}$ upscales the features from resolution $\frac{H}{P} \times \frac{W}{P}$ to $\frac{2H}{P} \times \frac{2W}{P}$. For training, we propose the following multi-scale reconstruction objective:

$$\mathcal{L}_{\text{Recon}} = \quad d\left(\mathcal{F}(\mathbf{x}), \boldsymbol{\Theta}(\mathcal{F}(\mathbf{x}_{1/2}), \mathbf{x}_{1/2}\right) + d\left(\mathcal{F}(\mathbf{x}_{1/2}), \boldsymbol{\Theta}(\mathcal{F}(\mathbf{x}_{1/4}), \mathbf{x}_{1/4}\right) \tag{1}$$

Where $\mathbf{x}_{1/2}$ and $\mathbf{x}_{1/4}$ are the images resized to 1/2 and 1/4 of the original image resolutions, and d is a distance function. For d we select the cosine distance metric, as it inherently normalizes the output and empirically achieves better performance. When processing $\mathbf{x}_{1/2}$ and $\mathbf{x}_{1/4}$, we follow the method of [5] to handle positional embeddings for images of different sizes. We optimize the parameters of the LiFT module $\boldsymbol{\Theta}$ to minimize Equation 1.

3.4 Training Details

Training LiFT is fast and efficient, as it is lightweight and does not require propagating gradients through the ViT backbone. The LiFT module, $\boldsymbol{\Theta}$, is a small network with $1.1M$ trainable parameters. We train LiFT for 5 epochs on the ImageNet dataset on a single GPU. We use a learning rate of 0.001 with a batch size of 256 for stride 16 training. We use DINO [5] ViT-S/16 as our base ViT, and we apply color jitter as the only augmentation. In total, training takes only $\sim$8 hours on one RTX A6000 GPU. Once our LiFT module is trained, it is a general purpose feature enhancement module that can be directly applied to a range of downstream tasks without need for any further finetuning of LiFT or the frozen ViT backbone. We demonstrate the flexibility of LiFT by applying features from the same DINO+LiFT model to several tasks in Sections 4.1, 4.2, and 4.3. We present additional analysis of different design choices for both training and inference in Appendix A.

3.5 Using LiFT with Downstream Modules

Our LiFT module can be easily applied to downstream tasks that directly operate on the output image features. However, LiFT can also be applied in circumstances where additional downstream modules follow the feature extracting backbone. To demonstrate this, we show how to apply LiFT to the ViTDet architecture [28], which combines a ViT backbone with a Mask-RCNN-style head [25], to perform COCO Object Detection and Segmentation. To achieve this, we inject the LiFT module after the backbone and before the Mask-RCNN head in a pretrained ViTDet model. We train this LiFT module on COCO training data using the same self-supervised objective described in Section 3.3. One can then optionally finetune the downstream head on the LiFT-generated features. We show that finetuning the head is preferable, but even without finetuning, adding LiFT is beneficial. We present these results in Section 4.4, and we provide additional details of our ViTDet+LiFT architecture in Appendix B.

3.6 Baseline Methods

We propose LiFT as a self-supervised, task-agnostic, general enhancement for ViT features. For baselines, we compare with the most similar prior works, Self-Patch [52] and Leopart [55], both of which finetune the entire DINO model on custom self-supervised losses to improve the quality of the spatial features for dense tasks. We demonstrate the effectiveness of LiFT compared with these methods for a range of dense and local tasks in Section 4. In addition, we compare

Table 1: Comparison between LiFT and other baselines on the keypoint correspondence task on SPair-71k. We report PCK@0.1 and 0.05 at multiple input resolutions.

	PCK@0.1				PCK@0.05			
Method/Resolution	56	112	224	448	56	112	224	448
DINO	2.04	12.67	24.76	28.6	0.51	3.61	9.54	15.33
Leopart	2.35	11.2	23.33	26.54	0.6	3.22	8.9	12.26
SelfPatch	2.13	12.18	23.03	27.34	0.44	3.61	9.32	14.44
DINO+LiFT	**5.05**	**17.72**	**28.68**	**31.38**	**1.19**	**6.29**	**14.72**	**18.90**

with the reduced token stride strategy of [2] and show the improved computational efficiency of LiFT in Section 5. Note that LiFT represents an orthogonal direction of ViT improvement that can be used in combination with methods like [2] and [55], which we show in the following sections and in Appendix C.

4 Performance Benefits of LiFT

4.1 SPair Keypoint Correspondence

This task involves matching keypoints between pairs of images in the SPair-71k dataset. We follow the evaluation protocol of Amir et al. [2] and report Percentage of Correct Keypoints (PCK) as the metric. We extract dense features using the frozen DINO+LiFT combination trained in Section 3.4. For brevity, we report results at PCK thresholds of 0.1 and 0.05 in the main paper but we can also see consistent improvements at the 0.01 threshold, as shown in Appendix D. For this task, the features of all methods are bilinearly interpolated to match the original image resolution before feature matching begins. Table 1 presents the results. Compared to both the base DINO model and finetuned approaches, LiFT performs the best across all resolutions for both PCK@0.1 (left) and PCK@0.05 (right). For lower resolutions like 56×56, LiFT more than doubles the performance on both metrics. Also note that under the PCK@0.1 metric, DINO+LiFT at 224×224 resolution beats all non-LiFT approaches run at the higher resolution of 448×448, even though both configurations produce final features of the same density.

4.2 DAVIS Video Object Segmentation

This task involves propagating a video object segmentation across multiple frames where the first frame ground truth segmentation mask is provided. This is achieved through dense feature matching between frames. Again, we extract dense features with the same pre-trained DINO+LiFT. We follow the evaluation protocol of [26] and, for brevity, we report results for the J&F mean metric in the main paper, but we can see consistent improvements across the J mean and F mean individually as shown in Appendix D. In Table 2, it can be seen that across resolutions and comparison methods, LiFT outperforms all other approaches. At the lowest resolution of 56×56, the performance gain over the base DINO is $\sim 2\times$. On average, we improve by 9.4 points over base DINO.

Table 2: Comparison on the DAVIS video object segmentation task. We report results for the J and F mean.

Method/Resolution	56	112	224	448
DINO	7.4	17.5	33.0	50.9
Leopart	6.9	16.1	30.3	45.1
SelfPatch	7.4	17.2	33.0	51.4
DINO+Bilinear	10.8	23.7	37.0	53.0
DINO+LiFT	**13.0**	**28.0**	**44.3**	**61.1**

Table 3: When added to ViTDet, LiFT boosts detection and segmentation performance on COCO. ViTDet+LiFT* shows results without fine-tuning the Mask-RCNN head.

	Detection			Segmentation		
Method	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
ViTDet	39.50	61.56	42.23	37.81	60.47	39.97
ViTDet+LiFT*	42.77	61.05	46.60	38.18	58.28	40.96
ViTDet+LiFT	**45.98**	**66.41**	**49.87**	**40.78**	**63.34**	**43.57**

Table 4: Unsupervised Object Discovery comparison on PASCAL VOC 2007, PASCAL VOC 2012, and COCO20K. We report results for the CorLoc metric.

Dataset	Method	Resolution			
		56	112	224	448
VOC07	DINO	20.74	50.07	65.60	68.27
	Leopart	18.92	32.59	51.59	48.79
	SelfPatch	18.04	41.99	62.40	63.62
	DINO+LiFT	**36.54**	**62.02**	**68.79**	**69.65**
VOC12	DINO	23.27	55.33	69.01	71.64
	Leopart	22.44	37.41	55.74	54.40
	SelfPatch	20.19	47.32	68.02	66.48
	DINO+LiFT	**40.56**	**66.21**	**70.91**	**71.71**
COCO20K	DINO	16.28	40.08	53.98	57.99
	Leopart	16.14	26.78	43.89	44.08
	SelfPatch	14.15	35.76	52.18	55.47
	DINO+LiFT	**27.72**	**50.20**	**58.03**	**60.50**

4.3 Unsupervised Object Discovery

We test the benefits of LiFT for Unsupervised Single Object Discovery on PASCAL VOC 2007 [18], PASCAL VOC 2012 [19], and COCO20K [30]. We apply TokenCut [48], which performs Graph Cut on features, to LiFT and the other baseline methods. Similar to prior works [11, 15, 35, 45, 48, 49] we report the Correct Localization (CorLoc) metric, which is computed as the fraction of the images in which at least one object box prediction has an IoU greater than a threshold (0.5) with a ground truth box. As shown in Table 4, LiFT gives a good boost in CorLoc, with gains across all resolutions and with LiFT outperforming the other approaches. It should be noted that the performance gains at lower resolutions are especially large. For example, at the 56×56 resolution, we see an improvement of 15.8, 17.29, and 11.44 on CorLoc for VOC07, VOC12, and COCO20K respectively.

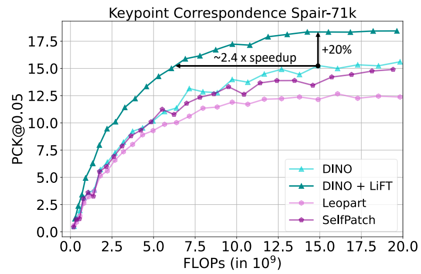
4.4 COCO Detection and Segmentation

To further demonstrate the versatility of LiFT, we show how it can be applied to COCO Detection and Segmentation using a ViTDet+LiFT model as described in Section 3.5. As LiFT is trained to emulate the same feature distribution generated for higher resolution inputs, the LiFT-generated features can be directly

Table 5: FLOPs comparison for a single forward pass across resolutions and strides for base DINO and LiFT. Along with FLOPS, we report the performance on both SPair-71k and DAVIS.

Method	Res.	Stride	FLOPs (G)	PCK@0.1	J&F mean
DINO	224	16	4.34	24.76	33.0
DINO+LiFT			5.30	28.68	44.3
DINO		8	16.07	29.92	43.9
DINO+LiFT			19.65	31.91	52.6
DINO	448	16	17.28	28.60	50.9
DINO+LiFT			21.12	31.38	61.1
DINO		8	66.60	31.92	61.9
DINO+LiFT			81.18	32.20	69.7

Fig. 3: Performance *vs.* Compute Cost trade-off curve for SPair-71k keypoint correspondence. For any given FLOP-budget, DINO+LiFT achieves far superior performance.



input into the downstream Mask-RCNN head without any additional training. However, we find that performing limited fine-tuning of the head can be beneficial. In Table 3, the combination of ViTDet+LiFT gives a ∼6.5% boost for Detection AP and a ∼3% boost for Segmentation AP when the downstream head is finetuned. Even without head finetuning, denoted as ViTDet+LiFT*, we see a small performance gain for AP and AP_{75} in both Detection and Segmentation.

5 Computational Efficiency of LiFT

We have shown that feature densification with LiFT provides significant performance benefits in several tasks. We now show why LiFT is a *Lightweight* transform, as it is vastly more computationally efficient than other alternatives for feature densification. A trivially easy way to boost the density of ViT features is to increase the resolution of the input image, which increases token density but also computational cost. Another option is the dense feature extraction strategy of [2] which increases the number of tokens in the network by reducing the stride during patch extraction. We present a comprehensive compute cost *vs.* performance benefit analysis for LiFT and alternative methods, and we show that LiFT acts as a "shortcut" to achieve higher resolution features for minimal extra compute cost. We also show that LiFT can be combined with these methods for further improved performance in exchange for higher compute.

5.1 LiFT, Resolution, and Stride

For this analysis, we take SPair Keypoint Correspondence and DAVIS Video Segmentation as the tasks, and we measure compute cost in FLOPs (G) against performance both with and without LiFT over a range of input resolutions and strides. As shown in rows 1 and 2 of Table 5, LiFT introduces a small increase in FLOPs from 4.34G to 5.30G (+22%) but also brings a significant performance improvement of 3.92, 5.18, and 11.3 points on the PCK@0.1, PCK@0.05, and J and F Mean metrics respectively. For comparison, reducing the stride from 16 to

Table 6: Comparison of parameters & FLOPs *vs*. performance at 224 × 224 resolution for Keypoint Correspondence. We report PCK@0.1 and PCK@0.05 on SPair-71k.

Method	Parameters	FLOPs (G)	PCK@0.1	PCK@0.05
DINO S/16	21M	4.34	24.76	9.54
DINO S/16+LiFT	22.2M	5.30	**28.68**	**14.72**
DINO B/16	85M	17.21	24.90	9.64

Table 7: LiFT when using different backbones for training, inference, or both. We report PCK@0.1 on SPair-71k.

Inference Model	No LiFT	Training Model: DINO	MoCo	ViT
DINO	28.6	**31.38**	16.02	20.71
MoCo	12.31	9.86	**16.34**	11.31
ViT	16.9	12.55	8.91	**18.69**

8 (row 1 *vs*. row 3) gives a similar level of improvement, but nearly quadruples the FLOPs from 4.34G to 16.07G (+270%). A similar trend of large improvements can be seen when comparing pairs of rows (3 *vs*. 4, 5 *vs*. 6) with DINO *vs*. DINO+LiFT. The best overall results are achieved by combining LiFT with increased resolution and reduced stride, but this also comes with the highest computation cost. It should also be noted that DINO+LiFT at 224×224 resolution and stride 8 (row 4) uses 3× less FLOPs than DINO at 448×448 resolution at stride 8 (row 7) while having similar performance on the PCK metrics.

5.2 Cost *vs*. Performance Trade-off Curve

Next, we present a comprehensive Compute Cost *vs*. Performance trade-off analysis on SPair Keypoint Correspondence. By incrementally increasing the input resolution, we can gradually increase both the performance and inference cost for DINO+LiFT and the baseline methods DINO, Leopart, and Selfpatch. As shown in Figure 3, we find that LiFT significantly outperforms all other methods at any given FLOP allowance, in most cases seeing a ∼20% performance gain. Alternatively, LiFT can be run at a lower input resolution to achieve equivalent performance at a fraction of the compute cost. For example, to achieve a score of 15.0 in PCK@0.05, DINO+LiFT only requires ∼6.25 Giga-FLOPs of compute power, while the other baselines require at least 15 Giga-FLOPs. At any point on the trade-off curve, DINO+LiFT far surpasses the alternatives.

5.3 Parameter Count

We acknowlege that the addition of the LiFT module slightly increases the overall model size and parameter count as shown in Table 6. However, this addition is quite small and only represents a +5.7% change in total parameters. For comparison, the jump from ViT-S to ViT-B results in a +304% increase in parameters. Furthermore, for dense tasks like SPair Keypoint Correspondence, we find that the performance benefits provided by LiFT far exceed the benefits of a larger backbone. Note that the methods Leopart and SelfPatch do not increase the parameter count of the ViT, as they finetune the DINO backbone instead of introducing new modules. However, we believe the major performance benefits of LiFT strongly justify the small extra costs in Parameter Count and Inference FLOPs for a given resolution. And as shown in the previous section, for any fixed FLOP-budget, DINO+LiFT achieves far superior performance.

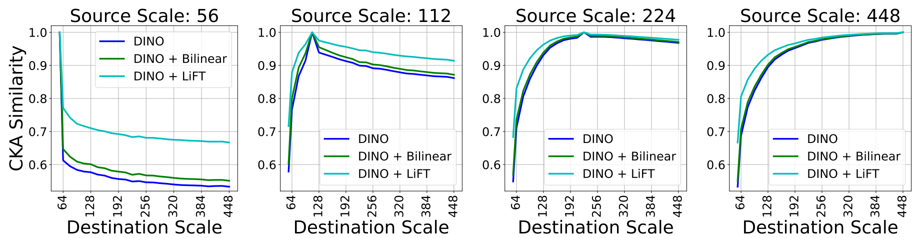


Fig. 4: CKA Similarity of ViT features extracted from SPair-71k images at different input image sizes, denoted by Source Scale and Destination Scale. LiFT produces features that are more scale-invariant, especially for smaller scale inputs and objects.

6 Properties of LiFT

6.1 LiFT and Scale Invariance of Features

In this experiment, we demonstrate that LiFT intrinsically learns to generate features that are more scale-invariant. We use the Centered Kernel Alignment (CKA) metric [13, 27, 41], which can measure the similarity of a pair of feature maps even when they are different sizes. Using images in the SPair-71k training set, we re-scale each image to a range of different sizes and then extract features with DINO or DINO+LiFT and measure the CKA similarity between all input size pairings. As a baseline, we also compare with bilinearly upsampled DINO features. In Figure 4, we take four source scales and plot the CKA similarity with the features at all other scales. We see that LiFT greatly improves the inter-scale feature similarity for small input scales, and moderately improves the similarity for medium and larger scales. This shows that LiFT produces representations that are more scale-invariant than the base DINO features. Bilinear upsampling does provide a small improvement in CKA similarity across scales, though the benefit is far smaller than that of LiFT. This scale invariance property of LiFT is learned automatically, and likely comes from its multi-scale reconstruction objective. LiFT must learn to counteract the effect of input scale in order to map features from low resolution inputs to those of high resolution inputs. This property is desirable for dense tasks where objects appear at different scales.

6.2 Enhanced Self-Similarity Maps with LiFT

To further improve our understanding of the dense features generated by LiFT, we visualize the self-similarity of the features in Figure 5. Base DINO S/16 yields 14×14 features for a 224×224 image. In our comparisons, we visualize these features alongside: a bilinearly interpolated upsampling of these same features to 28×28 (DINO+Bilinear), DINO features for a 448×448 input image, and finally DINO+LiFT features generated for a 224×224 input image. The last three configurations all yield 28×28 feature grids. To visualize these features, we select the center-most token feature for each of the maps and compute its similarity with all other features and visualize the similarity scores.

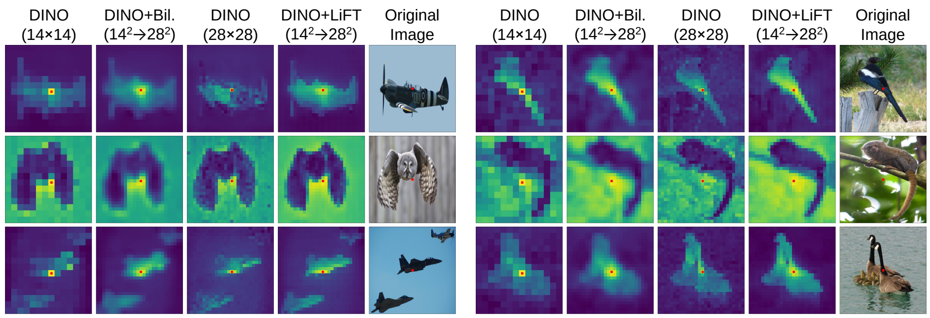


Fig. 5: Visualization of the self-similarity of features for DINO, DINO + Bilinear interpolation, DINO with higher resolution image, and DINO + LiFT. To generate this visualization, the self-similarity is computed using the feature corresponding to the center of the grid (marked in red) and all other features from each spatial location. Brighter map shows a higher similarity. Best viewed digitally in color.

We show six feature similarity maps in Figure 5, and additional sample visualizations can be found in Appendix E. We find that, qualitatively, the LiFT output gives a cleaner boundary and better highlights the content corresponding to the central patch. Having a high similarity to relevant regions and a clear boundary is beneficial for multiple localized downstream tasks. It also indicates that the DINO+LiFT features have better spatial awareness of object boundaries. Note that for row 2, where the central pixel corresponds to the background region, the similarity map highlights the background in the image, though it still appears that DINO+LiFT produces better similarity maps with sharper edges. In row 3, when there are multiple objects of the same type, DINO+LiFT better highlights the separate object instances. These results are qualitative, but they suggest that the LiFT-enhanced features have better content and boundary information than the base DINO features, which likely contributes to their improved performance in correspondence and segmentation tasks.

6.3 Variations in Backbone

One of the requirements of a feature densifying approach is that it should be easy to train on any backbone. To this end, we show that our approach consistently gives a performance gain with multiple different backbones: DINO (Table 8 row 1 *vs*. row 2), MoCo (Table 8 row 4 *vs*. row 5) and a fully-supervised ViT (Table 8 row 7 *vs*. row 8). This consistent improvement shows that LiFT can be trained in the exact same manner on differently trained ViTs without need for careful hyperparameter tuning. To verify that LiFT does not simply learn a bilinear interpolation, we apply LiFT modules trained on one backbone to the output of a different backbone. We show these results in Table 7 using the PCK@0.1 metric and 448 × 448 images. It can be seen that when LiFT is applied to a different model than what was used to train it, the performance drops and is lower than not applying LiFT. This shows that LiFT learns a model-specific feature-densifying transform and not a simple interpolation.

Table 8: Performance for LiFT with different backbones and for repeated application of LiFT. We report PCK@0.1 on SPair-71k.

Row	Backbone	Method	Resolution			
			56	112	224	448
1		-	2.04	12.67	24.76	28.6
2	DINO	LiFT	5.05	17.72	28.68	**31.38**
3		2×LiFT	**7.42**	**20.12**	**29.45**	31.35
4		-	1.27	3.43	7.37	12.31
5	MOCO	LiFT	6.48	10.51	14.13	16.34
6		2×LiFT	**8.72**	**13.21**	**16.12**	**17.08**
7		-	1.26	5.72	13.23	16.9
8	ViT	LiFT	3.76	9.21	**16.5**8	**18.69**
9		2×LiFT	**5.17**	**9.89**	16.49	18.18

6.4 Repeated Application of the LiFT Module

In our base approach, the LiFT module is applied once to the ViT features to double their resolution. In this section, we also test the potential benefits of applying the LiFT block multiple times. To check this, we take the LiFT module and super-resolute the features twice by passing the output of first super-resolution again through the same LiFT network. We denote this approach as '2×LiFT' in Table 8. We can see that in most cases the performance increases after recursively applying LiFT. This is especially true for the 56×56, 112×112, and 224×224 resolutions. For 448×448, there are a few places where it negligibly drops performance, but it still shows improvement for most cases.

7 Conclusion and Discussion

We have presented **LiFT**, a simple yet effective self-supervised **Li**ghtweight **F**eature **T**ransform to boost the density of features of pretrained ViT backbones. This approach allows us to extract higher resolution spatial features from ViTs which can then be used for multiple dense downstream tasks. LiFT is task-agnostic and gives significant boosts in SPair Keypoint Correspondence, DAVIS Video Object Segmentation, Unsupervised Object Discovery, and COCO Detection and Segmentation. This benefit comes for a fraction of the computational cost compared with other densification methods. LiFT is a lightweight module that is easily trained with a self-supervised objective, and it is far cheaper than full backbone fine-tuning, as is done by other prior works. Through extensive experiments, we have shown that LiFT can be trained easily on any backbone and consistently leads to improved performance by generating better quality, higher-density features. We also show that LiFT can be applied to its own output in a recursive manner enabling good performance with even lower image resolutions. Finally, we show that our surprisingly simple method has several desirable emergent properties including scale-invariant features, and better object boundary maps. This makes LiFT a useful multipurpose tool for many potential downstream applications.

Acknowledgements. We would like to thank our colleagues Matthew Gwilliam and Pravin Nagar for their feedback on this work.

References

1. Ali, A., Touvron, H., Caron, M., Bojanowski, P., Douze, M., Joulin, A., Laptev, I., Neverova, N., Synnaeve, G., Verbeek, J., et al.: Xcit: Cross-covariance image transformers. Advances in neural information processing systems **34**, 20014–20027 (2021)
2. Amir, S., Gandelsman, Y., Bagon, S., Dekel, T.: Deep vit features as dense visual descriptors. arXiv preprint arXiv:2112.05814 (2021)
3. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., Schmid, C.: Vivit: A video vision transformer. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6836–6846 (2021)
4. Bertasius, G., Wang, H., Torresani, L.: Is space-time attention all you need for video understanding? In: ICML. vol. 2, p. 4 (2021)
5. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9650–9660 (2021)
6. Chen, H., Pei, Y., Zhao, H., Huang, Y.: Super-resolution guided knowledge distillation for low-resolution image classification. Pattern Recognition Letters **155**, 62–68 (2022)
7. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020)
8. Chen*, X., Xie*, S., He, K.: An empirical study of training self-supervised vision transformers. arXiv preprint arXiv:2104.02057 (2021)
9. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9640–9649 (2021)
10. Chen, Z., Duan, Y., Wang, W., He, J., Lu, T., Dai, J., Qiao, Y.: Vision transformer adapter for dense predictions. In: The Eleventh International Conference on Learning Representations (2022)
11. Cho, M., Kwak, S., Schmid, C., Ponce, J.: Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1201–1210 (2015)
12. Cortes, C., Mohri, M., Rostamizadeh, A.: Algorithms for learning kernels based on centered alignment. ArXiv **abs/1203.0550** (2012), `https://api.semanticscholar.org/CorpusID:9137763`
13. Cortes, C., Mohri, M., Rostamizadeh, A.: Algorithms for learning kernels based on centered alignment. The Journal of Machine Learning Research **13**(1), 795–828 (2012)
14. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
15. Deselaers, T., Alexe, B., Ferrari, V.: Localizing objects while learning their appearance. In: Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11. pp. 452–466. Springer (2010)

16. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2021)
17. d'Ascoli, S., Touvron, H., Leavitt, M.L., Morcos, A.S., Biroli, G., Sagun, L.: Convit: Improving vision transformers with soft convolutional inductive biases. In: International Conference on Machine Learning. pp. 2286–2296. PMLR (2021)
18. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html
19. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html
20. Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., Feichtenhofer, C.: Multiscale vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6824–6835 (2021)
21. Ghiasi, A., Kazemi, H., Borgnia, E., Reich, S., Shu, M., Goldblum, M., Wilson, A.G., Goldstein, T.: What do vision transformers learn? a visual exploration. arXiv preprint arXiv:2212.06727 (2022)
22. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: 2009 IEEE 12th international conference on computer vision. pp. 349–356. IEEE (2009)
23. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16000–16009 (2022)
24. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9729–9738 (2020)
25. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
26. Jabri, A., Owens, A., Efros, A.: Space-time correspondence as a contrastive random walk. Advances in neural information processing systems **33**, 19545–19560 (2020)
27. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.: Similarity of neural network representations revisited. In: International Conference on Machine Learning. pp. 3519–3529. PMLR (2019)
28. Li, Y., Mao, H., Girshick, R., He, K.: Exploring plain vision transformer backbones for object detection. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IX. pp. 280–296. Springer (2022)
29. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. pp. 740–755. Springer (2014)
30. Lin, T.Y., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision (2014), `https://api.semanticscholar.org/CorpusID:14113767`
31. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021)

32. Min, J., Lee, J., Ponce, J., Cho, M.: Spair-71k: A large-scale benchmark for semantic correspondence. arXiv preprint arXiv:1908.10543 (2019)
33. Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., Van Gool, L.: The 2017 davis challenge on video object segmentation. arXiv preprint arXiv:1704.00675 (2017)
34. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
35. Rambhatla, S.S., Chellappa, R., Shrivastava, A.: The pursuit of knowledge: Discovering and localizing novel categories using dual memory. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9153–9163 (2021)
36. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12179–12188 (2021)
37. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. pp. 234–241. Springer (2015)
38. Shekhar, S., Bordes, F., Vincent, P., Morcos, A.: Objectives matter: Understanding the impact of self-supervised objectives on vision transformer representations. arXiv preprint arXiv:2304.13089 (2023)
39. Shocher, A., Cohen, N., Irani, M.: "zero-shot" super-resolution using deep internal learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3118–3126 (2018)
40. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: Transformer for semantic segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 7262–7272 (2021)
41. Subramanian, A.: Torch cka. In: https://github.com/AntixK/PyTorch-Model-Compare. Github (2021)
42. Tan, W., Yan, B., Bare, B.: Feature super-resolution: Make machine see more clearly. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3994–4002 (2018)
43. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International conference on machine learning. pp. 10347–10357. PMLR (2021)
44. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
45. Vo, V.H., Sizikova, E., Schmid, C., Pérez, P., Ponce, J.: Large-scale unsupervised object discovery. Advances in Neural Information Processing Systems **34**, 16764–16778 (2021)
46. Walmer, M., Suri, S., Gupta, K., Shrivastava, A.: Teaching matters: Investigating the role of supervision in vision transformers. arXiv preprint arXiv:2212.03862 (2022)
47. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 568–578 (2021)

48. Wang, Y., Shen, X., Yuan, Y., Du, Y., Li, M., Hu, S.X., Crowley, J.L., Vaufreydaz, D.: Tokencut: Segmenting objects in images and videos with self-supervised transformer and normalized cut. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023)
49. Wei, X.S., Zhang, C.L., Wu, J., Shen, C., Zhou, Z.H.: Unsupervised object discovery and co-localization by deep descriptor transformation. Pattern Recognition **88**, 113–126 (2019)
50. Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., Zhang, L.: Cvt: Introducing convolutions to vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 22–31 (2021)
51. Xu, Y., Zhang, J., Zhang, Q., Tao, D.: Vitpose: Simple vision transformer baselines for human pose estimation. Advances in Neural Information Processing Systems **35**, 38571–38584 (2022)
52. Yun, S., Lee, H., Kim, J., Shin, J.: Patch-level representation learning for self-supervised vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8354–8363 (June 2022)
53. Zhu, M., Han, K., Zhang, C., Lin, J., Wang, Y.: Low-resolution visual recognition via deep feature distillation. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3762–3766. IEEE (2019)
54. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)
55. Ziegler, A., Asano, Y.M.: Self-supervised learning of object parts for semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14502–14511 (2022)
56. Zontak, M., Irani, M.: Internal statistics of a single natural image. In: CVPR 2011. pp. 977–984. IEEE (2011)

Table 9: Ablation of different design decisions for LiFT training for three different ViT backbones. We report PCK@0.1 and PCK@0.05 on SPair-71k. For each backbone, we mark the best score for each metric and input resolution in **bold**.

		PCK@0.1				PCK@0.05			
Row	Method/Resolution	56	112	224	448	56	112	224	448
1	DINO	2.04	12.67	24.76	28.6	0.51	3.61	9.54	15.33
2	DINO + Random LiFT	1.45	2.37	4.21	6.16	0.35	0.7	1.41	2.35
3	DINO + LiFT No Img.	4.38	15.74	28.49	**31.42**	1.14	5.03	13.28	18.33
4	DINO + LiFT L1	4.48	16.64	27.77	31.03	1.01	5.93	13.88	18.09
5	DINO + LiFT L2	4.82	**17.72**	28.17	31.13	**1.29**	6.18	14.12	18.37
6	DINO + LiFT	**5.05**	**17.72**	**28.68**	31.38	1.19	**6.29**	**14.72**	**18.90**
7	MOCO	1.27	3.43	7.37	12.31	0.21	0.84	2.35	5.49
8	MOCO + Random LiFT	2.59	3.08	4.05	5.79	0.67	0.77	1.31	2.1
9	MOCO + LiFT No Img.	4.58	8.78	13.01	15.48	1.18	2.69	4.95	7.27
10	MOCO + LiFT L1	6.12	9.80	13.73	14.98	1.59	3.22	5.86	7.53
11	MOCO + LiFT L2	6.37	10.08	13.91	**16.41**	1.53	3.12	5.99	**8.34**
12	MOCO + LiFT	**6.48**	**10.51**	**14.13**	16.34	**1.74**	**3.36**	**6.42**	8.05
13	ViT	1.26	5.72	13.23	16.9	0.27	1.62	4.89	7.34
14	ViT + Random LiFT	2.36	3.29	7.15	8.21	0.58	1.09	2.33	3.13
15	ViT + LiFT No Img.	2.94	7.76	15.69	18.74	0.79	2.22	5.68	8.23
16	ViT + LiFT L1	3.27	8.32	16.04	18.29	0.79	2.74	6.77	8.45
17	ViT + LiFT L2	3.57	8.78	16.29	**18.80**	0.97	2.64	**6.82**	**8.87**
18	ViT + LiFT	**3.76**	**9.21**	**16.5**8	18.69	**1.02**	**2.71**	6.63	8.81

A Ablation Study of LiFT Design Choices

We present a careful analysis of LiFT design configurations by varying different factors in both training and inference. To show the general applicability and benefits of LiFT, we include three different backbones in this study: DINO [5], MoCo v3 (MoCo for short) [24], and a Fully Supervised ViT (ViT for short). We standardize the architecture to a ViT S/16 backbone for this analysis. We use the SPair-71k dataset and the keypoint correspondence task as the main representative metric for this analysis. The results are summarized in Table 9.

A.1 Random LiFT

One might question if LiFT actually benefits from training, or if the simple act of increasing the feature resolution with any arbitrary function is sufficient to improve performance. To test this question, we take a random initialization of the LiFT model and measure its performance. We denote this model as 'Random LiFT' in Table 9. It can clearly be seen in Table 9 rows 2, 8, and 14 that a randomly initialized LiFT model does not do anything meaningful as it performs poorly on all metrics. These results validate the importance of LiFT's self-supervised training method.

A.2 Ablation of Image Input to LiFT

In our approach, we increase the feature resolution through LiFT by also using the image as a source of finer spatial information. It should be noted that we use the image at the same resolution as was used to generate the initial features, which means LiFT does not have or require any additional information beyond the original ViT's input. To show the importance of this image information, we

Table 10: Ablation of LiFT training epochs on ImageNet, including longer training. Results are shown for DINO+LiFT on Keypoint Correspondence using PCK@0.1.

Res/Epochs	5	10	30	50	100
112×112	17.47	17.53	17.75	17.97	18.14
224×224	28.45	28.50	28.65	29.00	29.11

present a version of LiFT with the image input ablated, denoted as 'LiFT No Img.' in Table 9. We can see from rows 3 *vs*. 6, 9 *vs*. 12, and 15 *vs*. 18, that providing the image input helps LiFT produce better quality features which give improved performance on the keypoint correspondence task. It appears that ablating the image input is less harmful for higher-resolution inputs like 448, which makes intuitive sense as the feature map resolution is higher and thus more detail about the object boundaries can be represented. For DINO and the supervised ViT (rows 3 and 15), the no-image LiFT actually does very slightly better at 448 input resolution for PCK@0.1, but for all other cases normal LiFT is better. For PCK@0.05, the standard LiFT with image input is consistently much better. We believe this happens because LiFT can take direct cues regarding scene and object boundaries from the image input and generate higher resolution features which better respect these contours.

A.3 Effect of Distance Function

In our final approach, we use cosine distance to compute the loss between the ViT-generated higher resolution features and the upscaled features from LiFT. In Table 9, we compare with two alternative options for this distance function, specifically the L1 and L2 distance metrics. We denote these as 'LiFT L1' and 'LiFT L2' respectively. Cosine distance gives the best performance in most cases, such as in rows 4 & 5 *vs*. row 6, rows 10 & 11 *vs*. row 12, and rows 16 & 17 *vs*. row 18. For higher-resolution inputs, L2 distance is sometimes slightly better than cosine distance, but in most cases cosine is preferable. We believe this occurs because of the inherent normalization that cosine distance provides before computing the final loss.

A.4 Ablation of Training Epochs

As an additional experiment, we train the LiFT module on ImageNet for an extended period up to 100 epochs on 4 GPUs in Table 10. We find that there are small performance gains from training to very long epochs, however performance mostly saturates by epoch 5. At resolution 224, DINO+LiFT at 5 epochs gives a $\sim$3.7 point gain over the base DINO model, while training 95 epochs further only gives an additional 0.66 point gain. We believe this early saturation is thanks to the LiFT network's small size.

Table 11: Application of LiFT to various backbones for the Keypoint Correspondence task on SPair-71k for all metrics. LiFT gives consistent performance improvements.

	PCK@0.1				PCK@0.05				PCK@0.01			
Method/Resolution	56	112	224	448	56	112	224	448	56	112	224	448
DINO S/16	2.04	12.67	24.76	28.60	0.51	3.61	9.54	15.33	0.01	0.20	0.54	1.40
DINO S/16 + LiFT	5.05	17.72	28.68	31.38	1.19	6.29	14.72	18.90	0.06	0.29	0.91	2.52
DINO B/16	1.98	12.20	24.90	28.22	0.46	3.61	9.64	15.04	0.01	0.17	0.52	1.15
DINO B/16 + LiFT	5.43	17.74	29.35	31.27	1.29	6.56	14.80	18.10	0.04	0.37	0.92	2.43
DINO S/8	9.39	21.30	31.05	32.15	2.35	8.44	16.74	18.96	0.15	0.39	1.19	2.32
DINO S/8 + LiFT	12.90	26.73	34.54	34.58	4.35	11.99	20.61	21.01	0.18	0.75	2.21	3.77
DINO B/8	8.88	20.40	30.08	30.89	2.83	7.70	15.81	17.84	0.12	0.39	1.09	1.95
DINO B/8 + LiFT	12.21	25.17	33.23	33.17	4.22	11.73	19.39	20.18	0.13	0.69	2.27	3.32
MOCO S/16	1.27	3.43	7.37	12.31	0.21	0.84	2.35	5.49	0.00	0.03	0.10	0.31
MOCO S/16 + LiFT	6.48	10.51	14.13	16.34	1.74	3.36	6.42	8.05	0.04	0.16	0.42	0.73
ViT S/16	1.26	5.72	13.23	16.90	0.27	1.62	4.89	7.34	0.02	0.06	0.30	0.50
ViT S/16 + LiFT	3.76	9.21	16.58	18.69	1.02	2.71	6.63	8.81	0.02	0.13	0.45	0.72
Leopart S/16	2.35	11.20	23.33	26.54	0.60	3.22	8.90	12.26	0.05	0.10	0.47	0.79
Leopart S/16 + LiFT	4.24	15.61	27.77	30.06	1.22	5.16	12.81	15.66	0.02	0.25	0.74	1.39

B Additional Details for ViTDet+LiFT

For our experiments combining LiFT with ViTDet [28], we increase the size of our LiFT module to address the additional complexity of the task and backbone. To be consistent with ViTDet, we use an MAE-trained ViT-Base backbone instead of the ViT-Small used in our other primary experiments. Note that a standard ViT-Small model outputs feature maps with 384 channels, while ViT-Base outputs 768 channels. To handle the increased number of channels, we commensurately increase the number of channels in the layers of our LiFT module. We also add an additional convolutional block to the encoder segment. This larger LiFT module has a total of $7M$ parameters, as compared with the $1.2M$ parameter version used for smaller architectures. The ViTDet model used has $111M$ parameters, so our combined ViTDet+LiFT architecture has $118M$ parameters total. This is a 6.3% increase in total parameters, which is similar to the relative percentage increase of the smaller LiFT version used with DINO S/16. Also, here we train LiFT on the COCO dataset in place of ImageNet. Because the COCO dataset is much smaller than ImageNet, we train on it for 100 epochs.

C Additional Backbones with LiFT

C.1 Performance Improvements with LiFT for Other Backbones

We further demonstrate the general utility of LiFT by applying it to several additional backbones, including Leopart [55] and several other DINO [5] ViTs, namely ViT-S/8, ViT-B/16, and ViT-B/8. The results are summarized in Table 11. LiFT shows consistent improvement for the various architectures and models across patch sizes (8 and 16), trainings (Leopart and DINO) and backbone sizes (Base and Small).

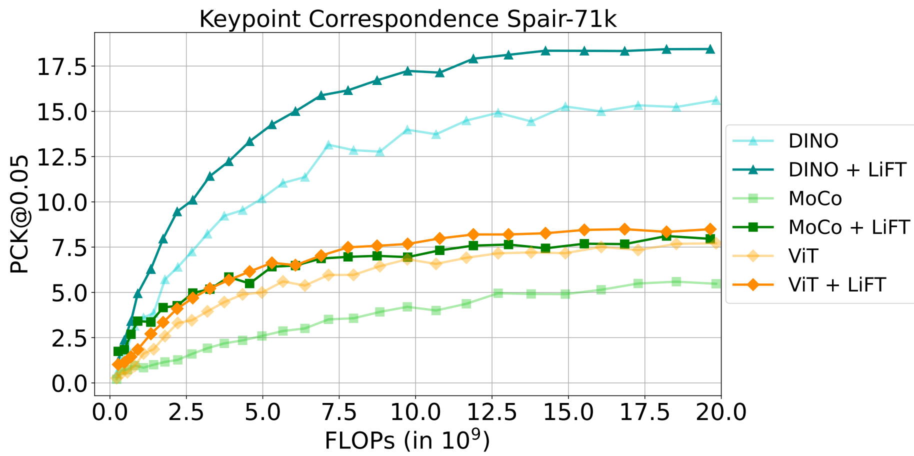


Fig. 6: Performance *vs*. Compute Cost trade-off curve for LiFT when combined with different ViT backbones. Results are presented for SPair-71k Keypoint Correspondence. LiFT provides a performance boost for all three backbones at any FLOP budget.

C.2 Cost *vs*. Performance Trade-off for Other Backbones

We extend the Performance *vs*. Compute Cost analysis from Section 5.2 to include both the MOCO and fully-supervised ViT backbones, as shown in Figure 6. We find that LiFT consistently boosts the performance of all three backbones at all FLOP allowances.

D Additional Metrics

We present results for additional metrics on the SPair-71k [32] and DAVIS [33] datasets. For SPair-71k, we additionally report the values for PCK@0.01 in Table 12. For DAVIS, we also report the J Mean and F Mean in Table 13. We present these results alongside the previously reported metrics for completeness. For SPair-71k, the PCK@0.01 metric demands the most precision, and the additional feature resolution provided by LiFT provides consistent improvements. For DAVIS, both the J Mean and F Mean are also consistently improved by adding LiFT.

E Additional Similarity Map Samples

In Section 6.2, we found that the feature self-similarity maps for DINO+LiFT more clearly and sharply outline the central object in an image. To further highlight this, we provide a zoomed-in comparison of the difference between DINO+LiFT and DINO+Bilinear upscaling in Figure 7. We can see that DINO with Bilinear upsampling highlights the main object, but the outline is hazier and less precise due to the smoothing of the features. Meanwhile, the upscaled

Table 12: Comparison between LiFT and other baselines on the Keypoint Correspondence task on SPair-71k for all metrics.

	PCK@0.1				PCK@0.05				PCK@0.01			
Method/Resolution	56	112	224	448	56	112	224	448	56	112	224	448
DINO	2.04	12.67	24.76	28.6	0.51	3.61	9.54	15.33	0.01	0.2	0.54	1.4
Leopart	2.35	11.2	23.33	26.54	0.6	3.22	8.9	12.26	0.05	0.1	0.47	0.79
SelfPatch	2.13	12.18	23.03	27.34	0.44	3.61	9.32	14.44	0.02	0.17	0.42	1.12
DINO+LiFT	**5.05**	**17.72**	**28.68**	**31.38**	**1.19**	**6.29**	**14.72**	**18.90**	**0.06**	**0.29**	**0.91**	**2.52**

Table 13: Comparison between LiFT and other baselines on the DAVIS Video Object Segmentation task with additional metrics J Mean and F Mean.

	J Mean				F Mean				J & F Mean			
Method/Resolution	56	112	224	448	56	112	224	448	56	112	224	448
DINO	0.10	0.22	0.38	0.52	0.05	0.13	0.28	0.50	0.07	0.18	0.33	0.51
DINO+Bilinear	0.14	0.29	0.43	0.54	0.08	0.18	0.31	0.52	0.11	0.24	0.37	0.53
Leopart	0.09	0.20	0.35	0.47	0.05	0.12	0.26	0.43	0.07	0.16	0.30	0.45
SelfPatch	0.10	0.22	0.38	0.53	0.05	0.13	0.28	0.50	0.07	0.17	0.33	0.51
DINO+LiFT	**0.16**	**0.33**	**0.48**	**0.59**	**0.10**	**0.23**	**0.41**	**0.63**	**0.13**	**0.28**	**0.44**	**0.61**

feature map produced by LiFT better respects object contours and produces a much sharper feature self-similarity map.

Finally, we provide additional samples further showing the benefits of LiFT for self-similarity maps, as shown in Figure 8. In rows 1 to 8 (left), we show samples with single central objects of differing shapes and sizes. We see that the feature self-similarity maps for DINO+LiFT more uniformly fill the foreground object region, and have less noisy correlations with background regions. In rows 1 to 3 (right), we show samples where the central feature vector, shown by the red marker, lies on a background region. In these cases, we still see sharp contours around the foreground objects, or around the body of water in row 1 (right). In cases like rows 4 to 8 (right), when there are multiple overlapping instances of the same object class, we see a uniform highlighting of the multiple object instances. We also see that DINO+LiFT better highlights thin structures in objects, like the teapot handle and tripod legs in rows 7 and 8 (left). For comparison, when DINO (without LiFT) is given a doubled input size, these details are sometimes lost to noisy background regions.

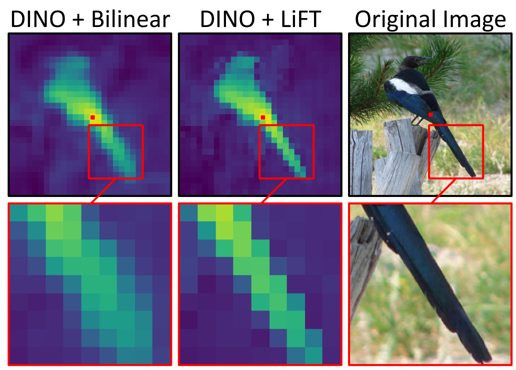


Fig. 7: Compared with DINO+Bilinear, DINO+LiFT gives feature self-similarity maps with much sharper object boundaries, especially when zoomed in.

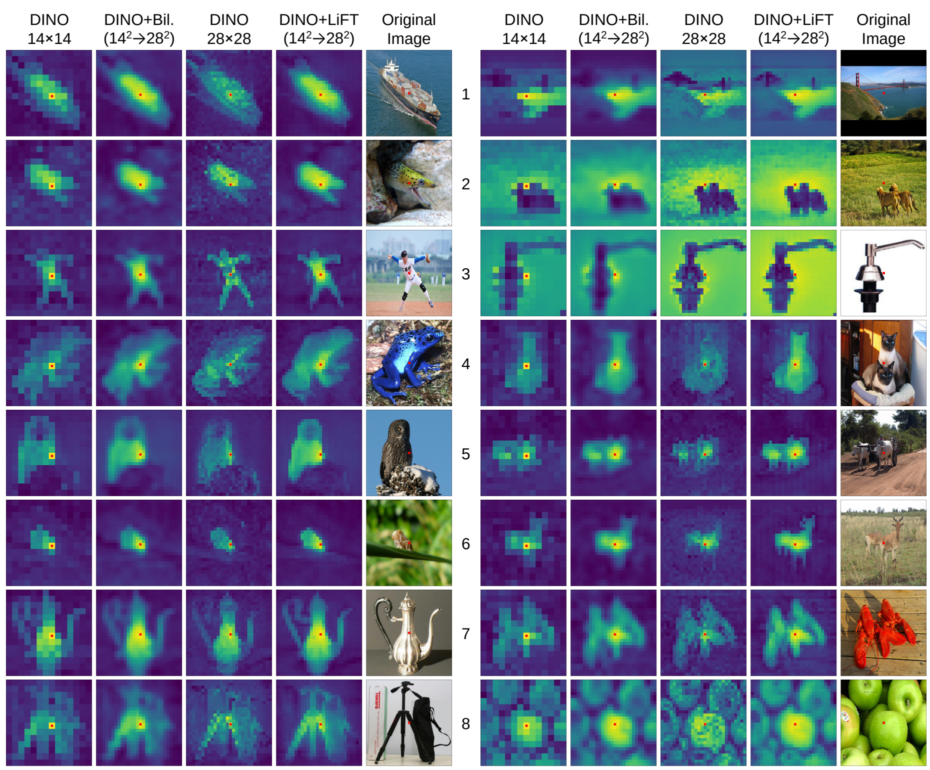


Fig. 8: Additional visualizations of the self-similarity of features extracted from DINO, DINO+Bilinear interpolation, DINO with higher resolution image, and DINO+LiFT. The input image is shown for comparison. The self-similarity is computed using the feature corresponding to the center of the grid (marked in red) and all other features from each spatial location. Brighter pixels show a higher similarity.