ElasticDino: Dense Semantic Features using Elastic Deformation

Ulysse Mizrahi* Sean Zaretzky* Dov Danon Amit H. Bermano
Tel Aviv University
https://odusseys.github.io/ElasticDino/

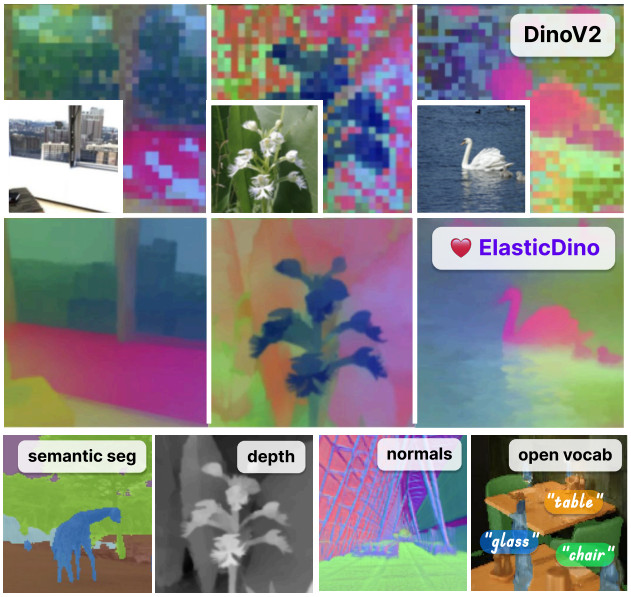


Figure 1. ElasticDino upscales ViT features and corrects their artifacts, allowing to efficiently train a variety of dense vision tasks.

Abstract

*Vision transformers (ViTs) serve as the backbone for countless vision tasks, including dense prediction tasks, such as segmentation and depth estimation, even though the quadratic complexity of self-attention severely limits feature resolution and induces spatial artifacts. In this work, we introduce **ElasticDino**, a lightweight adapter which produces high-resolution feature maps, adhering to the fine details of the input image. As a key insight, we observe the consistency of artifacts across channels, and simultaneously resolve them to match the input image structure. Using a slim architecture based on spatial deformations, **ElasticDino** produces pixel-aligned, high-resolution feature maps efficiently. Leveraging deformations allows strictly preserving sharp semantics, and thereby avoid the smoothness biases inherent to alternative approaches. Due to the absence of ground truth, we employ pixel-level linear heads to predict a set of downstream dense tasks, guiding the deformation process without leaking task specific semantics into the model. We demonstrate superior accuracy over current methods on a range of popular dense task benchmarks, with better computational performance.*

1. Introduction

Foundational Visual features have become the backbone for most vision tasks in recent literature. Under this paradigm, a large foundational model is typically contrastively trained to embed images into a latent space. This embedding is rich and semantic by nature, and serves as a great starting point for various downstream tasks with significantly less training requirements. Specifically, self-supervised features trained using a Vision Transformer (ViT) architecture (i.e., DINO [3, 19, 25]) are employed in state-of-the-Art semantic, panoptic and instance segmentation [5, 14], object detection [16], monocular depth estimation [2, 22, 34, 35], optical flow estimation [27], and more.

Despite their stellar success, these models have an inherent flaw: built on self-attention, they require quadratic memory, imposing patchification of the input image, effectively restricting the resolution of the final feature maps and introducing artifacts in their semantics. Indeed, for most hardware, a patched resolution of 32×32 or 64×64 is typically used, even when training dense tasks such as segmentation or depth estimation. Typically, a densification head is added to recover full resolution, (e.g. DPT [23]), which requires large amounts of data to recover fine details, as attested by the large and complex datasets used to train monocular depth estimation tasks [34, 35]. This approach implicitly also requires training a specific head for each downstream task due to poor transferability.

A different approach that has recently emerged in the literature [8, 11, 13, 28, 31] seeks to enhance the resolution of the produced features, indifferent the downstream task. As we demonstrate, such approaches still suffer from performance issues due to attention-based layers, and base their upscaling process on reconciling high and low resolution

*Equal contribution.

features through cropping-like mechanisms, leading to inevitable bleeding of feature semantics at boundaries.

In this paper, we introduce ***ElasticDino***, a lightweight upscaling method to enhance DINO features, yielding efficient and sharp result through a deformation-based architecture and simple training through a collection of minimalistic heads.

To achieve this we first observe that artifacts are consistent across feature-channels, and can thus be resolved for all channels simultaneously (see Figure 2, subsection 3.1). As such, matching of feature and image structure our primary goal, regardless of specific per-chanel values. Second, we observe that most feature artifacts stem from the added positional encoding or under-sampling of semantic edges, resulting in thickening of objects and region bleeding. Hence, we learn deformation fields that align all feature channels to the image structure, alternating between upsampling and deformation. We observe that deformations offer efficient structure preservation, inherently and consistently adhering to sharp semantics. This alleviates otherwise inherent smoothness biases that are typical to the mechanisms used by previous work, and hence minimizes bleeding and fuzzy transition artifacts. Critically, since no upscaled ground truth exists in our setting, instead of cropping or further under-sampling approaches, we train **ElasticDino** using a set of five lightweight downstream heads, each learning to predict a different dense map, while limited to a $1x1$ receptive field, forcing pixel-level matching of feature semantics.

Through careful evaluation, we demonstrate that **ElasticDino** produces high-quality alignment of semantics and structure, while removing artifacts such as those induced by positional encoding. Using lightweight heads, we report superior results compared to other upscaling methods on semantic segmentation, open-vocabulary segmentation, and depth estimation, with minimal computational effort.

2. Related work

2.1. Vision transformers and semantic features

Vision transformers have shown exceptional performance on vision tasks, being able to model complex long-range dependencies, and setting state-of-the-art results in a variety of vision tasks such as classification [37], segmentation [5, 14], object detection [16], monocular depth estimation, [2, 34, 35], joint text-image encoding [21], Vision Large Language Models [30], and image generation [20].

Many of the above tasks relied on pre-trained ViT encoders, much like vision models previously relied on convolutional backbones such as ResNet [12]. Indeed, ViTs have shown the ability to generate spatial semantic features, which transfer to a variety of tasks. Chief among these is the DINO family [3, 19, 25], which through unsupervised training provides exceptionally strong semantic features, and has since become the go-to backbone for various state-of-the-art models [16, 34].

2.2. Correction of ViT artifacts

In order to correct for artifacts in ViT (in particular DINO) activations, several methods have been introduced - though without considering higher feature resolutions.

Soon after the introduction of DINO, it was realized that a large part of the artifacts encountered were due to a leakage of global semantics in patch features [9]. "Register" tokens were introduced and added to the tokenized image sequence, meant to capture global semantics, while letting patch features absorb relatively more local semantics.

Self-Patch [36] improves upon DINO's self-supervision approach by enforcing local invariances among neighboring patches, improving spatial consistency and downstream performance although without upsampling features to higher resolutions. This however requires retraining of the model.

More recently, **Denoising Vision Transformers** [33] was introduced to mainly resolve the artifacts introduced by positional encodings. They achieve this through decomposing feature maps into a "ground truth" component and several shared positional noise components. They impose consistency between the features produced by image crops, thus removing positional information from the image features. It is important to note that positional information in feature maps is nonetheless important; removing it entirely severely reduces the out-of-the-box ability to model positional dependencies between objects or points of the image.

2.3. ViTs and dense vision tasks

The first ViT densification approaches were task specific, with **Dense Prediction Transformer (DPT)** [23] still being a popular architecture. It combines a ViT backbone with a convolutional decoder to generate high-resolution depth maps, leading to success in tasks such as monocular depth estimation [22, 34, 35]. In addition, **ViT-Adapter** [4] adds a trainable module to a ViT, injecting vision-specific priors. DySample [17] dynamic sampling, SAPA [18] and ReSFU [40] further refine spatial semantics. While displaying impressive performance, these approaches struggle with high-frequency details, and more importantly demand data-intensive per-task training. Thus inducing high costs and restricting applicability to specialized domains.

More recently, a series of task agnostic literature has emerged, surpassing results of the task specific approaches while offering better generalization. **FeatUp** [11] introduces two separate upscaling methods, both based on reconciling features for various augmentations of an image. The first is trained per-image and demonstrates honorable performance, but requires over a minute of training for every image, precluding most real-world applications. The

second method results in a generic upscaler, which displays gross artifacts - unavoidable with the joint-bilateral-upsampling kernels employed.

Similarly, **Lift** [26] uses a simple CNN architecture which combines image and features and then upsamples them. The training procedure consists in matching upscaled features produced at a lower image resolution to features produced at higher image resolution, and not upscaled. As the chosen "ground truth" is not free of artifacts, the incentive to fully resolve them is not complete. Further, the editing of feature values, which are mixed with image encodings, does not guarantee strict preservation of semantics.

LoftUp [13] passes images, features and positional encodings to cross-attention blocks. The training procedure is two-staged. First, they compute various segmentation masks on the image, use a form of bilateral upsampling guided by the masks, and use these as pseudo-labels. In a second stage, they perform teacher-student distillation, now using as objective the reconciliation of feature values along crops. This suffers the same issues, with direct feature value editing, and reduced incentive to correct artifacts, especially bleeding.

JAFAR [8] uses cross-attention using the encoded image as the query, features as value, and both as key, to compute pixel-to-semantics attention scores and query them within the feature values. This ensures that feature semantics are preserved. The training procedure is identical to LIFT and as such lacks the incentive to fully resolve artifacts.

AnyUp [31] proposes a backbone-agnostic upsampling module capable of transferring across models without retraining the upscaling head. To address the computational inefficiencies and the tendency of global attention to reference distant, unrelated image areas, AnyUp restricts its cross-attention computation to local spatial windows, supplemented by a feature-agnostic layer to enable transferability. However, the training procedure relies on matching features from randomly sampled image crops. Similar to FeatUp, this formulation does not provide a strong enough incentive to fully resolve the inherent spatial artifacts of the backbone.

UPLiFT [28] addresses the heavy memory bottlenecks of single-step cross-attention approaches by proposing an iterative upsampling framework. It relies on an encoder-decoder architecture equipped with a local attention pooling operator to maintain linear complexity with respect to the number of visual tokens. Similar to LIFT, the training procedure optimizes a multi-step reconstruction loss that matches iteratively upscaled features from low-resolution inputs to the standard features extracted from high-resolution inputs. By explicitly treating the raw backbone features as the ground truth target, this formulation inherits the backbone's spatial artifacts and provides no incentive to fully resolve them.

3. Method

ElasticDino is an upscaling network for DINO features, learned as iterative upsampling followed by elastic deformations of the features. Given an image and the corresponding feature maps generated by the ViT, our goal is to upscale the feature maps while matching the image structure. Figure 3 depicts an overview of the architecture: The input features are repeatedly upscaled and deformed, with the deformation field being predicted using convolutional layers. We note that the advantages of this approach are lightweight computation that allow for higher-resolution maps at negligible cost with respect to backbone evaluation, and better handling of high-frequency edges. Another major advantage of this approach is that since we sample from the original feature map, we diminish hallucination hazards, ensuring strong structural and semantic preservation.

As there are no ground truth upscaled features available, we train 1×1 convolutional heads on a collection of dense downstream tasks, ensuring no spatial context and pixel-wise feature and structure matching.

We emphasize that deformations strongly restrict the output, forbidding feature editing and preventing task leakage, or neglect of specific information correlating to the tasks used or not used during training. Indeed, when training without the depth and the segmentation tasks, performance remains strong on these held-out tasks. See supplementary materials for more detail. See Supplementary material for visualizations of deformations through upscaling process.

3.1. Cross-Channel Consistency

Due to their patch-wise decomposition of the image and combined use of positional encodings, ViTs typically introduce artifacts into the produced feature maps. We identify two main types of artifacts (Figure 2, bottom left). First, in regions of near-constant color, positional artifacts are prominent and values may vary widely, without semantic difference. It has been observed [9] that this happens in great part from global semantics being locally encoded, but positional encodings are also a significant contributor, as they are the only varying input to the ViT in a constant patch. Second is feature bleeding (Figure 2, bottom right): objects and boundaries are dilated due to discretization.

We make the crucial observation that ViTs introduce artifacts in a way which is consistent across channels (Figure 2, top). To substantiate this, we look at edge correlation between channels, and the original input image: we analyze 1,000 random ImageNet images. In each image, we isolate near-constant regions with an unsupervised local-variance filter, followed by erosion at the patch-grid scale so that measured gradients are not contaminated by neighboring edges; images without a sufficiently large flat region are discarded, leaving about 10% of images. This selection

is purely geometric and independent of the correlations being measured. On the remaining patches, we compute Sobel gradients of the backbone feature map and input image, and report the mean cross-channel Spearman correlation of gradient magnitudes, alongside two baselines: the same cross-channel correlation over the full token grid, and the channel-to-image correlation within the same masked region. In flat regions, cross-channel agreement is substantially higher than channel-to-image structure for both DINOv2 (0.29 vs. 0.11) and DINOv3 (0.30 vs. 0.12), and exceeds the full-image cross-channel baseline (0.18 and 0.22), showing that channel-aligned artifacts are concentrated where image gradients are weak rather than being a generic property of the grid. As a control, at image edges the channel-to-image correlation rises and the gap closes (0.17 vs. 0.15 for DINOv2; 0.19 vs. 0.23 for DINOv3), as expected when feature gradients follow real boundaries. Gradient directions are likewise shared across channels in flat regions (mean $\cos^2 \approx 0.73$ for both backbones). These findings indicate that one can learn a correction to all channels simultaneously, leading to very efficient resolution recovery.

3.2. Architecture

The architecture, depicted in Fig. 3 consists of alternating Feature Deformation layers, which correct artifacts and 2x bilinear upsamples. A first deformation layer is applied, then followed by an alternation of upsampling and deformations. The deformation process is guided by the input image, allowing to introduce the image structure at that resolution to guide the deformation of the features.

Each upscaling stage is composed of a series of Feature Deformation Blocks, which take as inputs the "dirty" features at the current resolution and the image at the same resolution. These are projected to a common hidden dimension and concatenated; and then fed to a depthwise-separable [6] residual convolution network, which outputs a 2D deformation field. The field is applied to the input features to recover the denoised features.

In other terms, given features f, an image I at a current identical resolution, a single deformation block consists of two projections p_f, p_I, a resnet encoder R which outputs a 2D deformation field

$$\vec{F} = R(p_f(f) \oplus p_I(I))$$

Which is applied coordinate-wise:

$$\hat{f}_{u,v} = f_{\vec{F}_{u,v}}$$

The motivation for using depthwise separable convolutions is the desire to achieve an extremely lightweight architecture, given the fact that feature dimensions are already high, which at higher resolution leads to heavy memory and compute utilization. We observed no performance degradation as well as much faster training and inference.

Pixel-wise downstream tasks for semantic guidance Unfortunately, we have no ground truth maps in the upscaled resolution, and thus need to define an alternate training objective.

We make the observation that a well-upscaled and denoised feature map should be able, by design, to predict any dense task pixel-wise.

We hence introduce a multi-task learning approach, where we simultaneously train linear heads for downstream dense vision tasks (see Section 3.3). Crucially, the 1x1 receptive field prevents the heads from combining feature values from different spatial positions, forcing the predictions to only use the semantics at that exact pixel. Should a larger receptive field be employed, the incentive to match the semantics at the current pixel would be removed, and structure would be learnt both in the upscaler and the downstream head.

It is important to note that this approach can only work because we are employing deformations and sampling from the features themselves. Indeed, if the upsampling process were to directly edit feature values, the semantics of the downstream tasks could easily just become "baked in" to the feature maps, without any incentive to jointly edit all channels simultaneously. We show the effect of such an approach in an ablation study- Fig. 5.

For each downstream task (see below), we employ a simple linear head to predict the target. Furthermore, to encourage faster learning at every resolution, the same head is applied to every intermediate feature map, and the corresponding multi-resolution predictions are all used to compute the training objective.

3.3. Losses and Tasks

All the tasks we use predict a dense map on the full resolution. For every task except segmentation (for which we use the standard cross-entropy loss), the base loss is defined as $\mathcal{L}_{\text{base}}(x, \hat{x}) = ||x - \hat{x}||^2 + \text{SSIM}(x, \hat{x})$, where SSIM denotes the structural similarity loss [29]. For all tasks except segmentation, and in order to encourage reconstruction of high frequency details, we additionally penalize discrepancies between the channel-wise spatial gradients and laplacians, following [2]: $\mathcal{L}_{\text{grad}}(x, \hat{x}) = \mathcal{L}_{\text{base}}(\nabla x, \nabla \hat{x})$, $\mathcal{L}_{\text{lap}}(x, \hat{x}) = \mathcal{L}_{\text{base}}(\Delta x, \Delta \hat{x})$

To encourage faster reconstruction at every stage, we compute the loss for all intermediary feature maps, as well as at every resolution. Therefore, denoting $\hat{x}_{i,j}$ the output of the i-th stage for the j-th prediction head, $\hat{x}_j$ the ground truth for task j, $r_k = 32 \times 2^k$ where $k \in [[0, 3]]$ the resolutions used, and $\mathcal{L}_j$ the loss for task j (see the task list below), we apply the following multi-scale loss:

$$\mathcal{L}_{\text{ms}}(x, \hat{x}) = \sum_{i,j,k} \mathcal{L}_j(\text{interpolate}(x_i, r_k), \text{interpolate}(\hat{x}y_{i,j}, r_k))$$



Figure 2. Visualizations of features values across different channels. We observe artifacts are consistent across channels (top row). In addition, low frequency regions of the image emphasize positional artifacts (bottom left), and the low resolution nature of the features typically induce thickening of object edges (bottom right).

This allows gradients to flow at each stage, rather than through the entire upscaling network, greatly stabilizing and speeding up the learning.

The downstream tasks we use represent a diverse sets of maps to be predicted from the image. We find each to help improve overall embedding quality (see Section 4), and believe more can be added. We use the following downstream tasks: RGB reconstruction, monocular depth, semantic segmentation, surface normals, intrinsic decomposition. Training is performed on two interspersed datasets: Hypersim [24] and Imagenet-22k [10]. See Supplementary material for details on losses and the training procedure.

4. Experiments

We perform quantitative and qualitative evaluation on a prototype designed to upscale DINOv2 features, and report the results below. We report semantic segmentation and depth estimation comparisons, and open vocabulary segmentation in the supplementary material. Following recent publications [31], we also perform some of the same experiments for DINOv3 [25] features. For implementation details, including results from our training head tasks, see supplementary material.

Qualitative Results are presented in Figure 4. Along side the input image, we visualize the top-3 PCA components of DINOv2 features, and resulting upscaling maps. As can be seen, artifacts are corrected and crisp semantic edges are formed, corresponding to the original image structure. **ElasticDino** alleviates artifacts in low variance regions of the image, and demonstrates semantic results that are decoupled from RGB variations in the image. For example, The similarly colored faucet is represented with distinctly different features compared to the background, while the flag is represented similarly throughout even though it consists of differently colored texture. For a more comprehensive demonstration of qualitative results, please refer to the supplementary material. We also note that while positional encoding related artifacts are alleviated, the positional information itself persists, and is actually extracted from the features with more ease compared to competing upscaling methods. See supplementary material for details on this experiment.

4.1. Quantitative Comparisons

4.1.1. Task Specific (Semantic segmentation)

We train simple convolution heads for semantic segmentation on top of **ElasticDino** and compare performance with two leading dedicated segmentation methods, Mask2Former [5] and OneFormer [14]. The data sets examined are MS-COCO [15], ADE20k [38, 39] and Cityscapes [7] (Cscapes), all standard data sets in semantic segmentation benchmarks.

ElasticDino displays higher performance as measured by Mean-IoU (Intersection over Union) on all three validation datasets, with a very large margin on ADE20k and Cityscapes. Qualitatively (In Supplementary material) we observe not only better classification ability, but much finer segmentation boundaries. This implies obvious advantages to downstream applications where sensitivity is key, such as autonomous driving.

4.1.2. Task Agnostic (Segmentation and Depth)

As our primary comparison, we further compare to task-agnostic methods that employ DINO feature upsampling (Table 3 for DINOv2, and Table 4 for available methods with DINOv3). These include training free Nearest neighbor and bilinear upsampling, and recent state-of-the-art methods [8, 11, 13, 28, 31]. We follow established evaluation protocols, as described in by JAFAR [8], measuring mean Intersection over Union (mIoU) and pixel-level accuracy (Acc) for the semantic segmentation task over four datasets, and Root Mean Square Error (RMSE) and δ_1 (measuring the percentage of pixels where that are less than 25% of the ground-truth) for the depth estimation task over one dataset. In the supplementary material, we also compare performance for zero-shot open vocabulary segmenta-

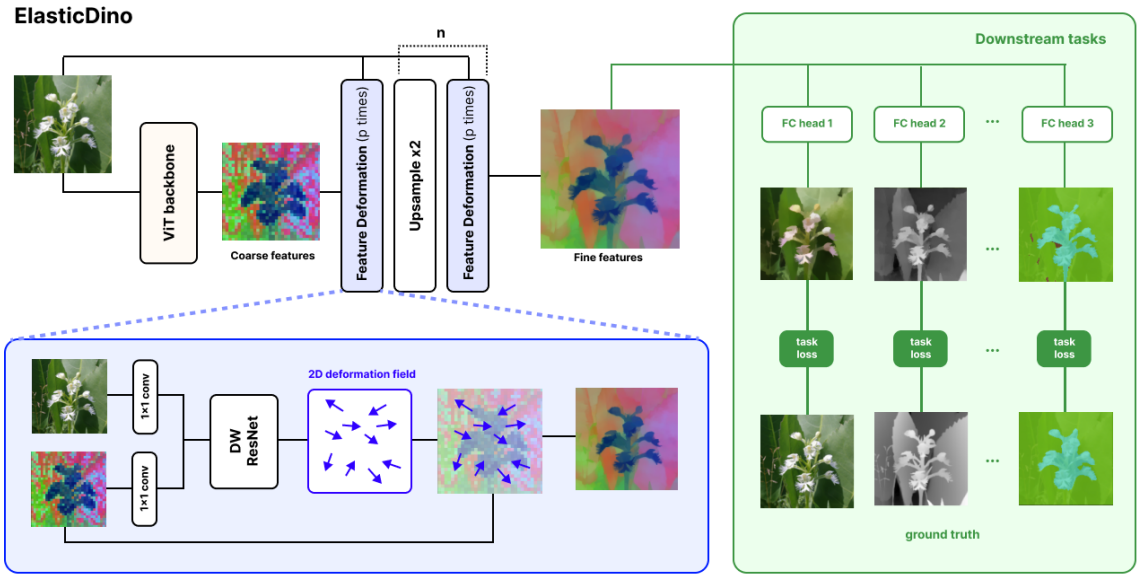


Figure 3. Method overview. The input image is passed to the (DINO) backbone for the low resolution, artifact-full features. These are successively passed to a sequence feature-fixing layers and then bilinearly upsampled by a factor 2. **Feature Deformation Layers** project and concatenate both the image and the semantic features, which are fed to a series of depthwise residual blocks, that end up generating a deformation field that warps the feature map. **Task heads** induce the sole losses of the system. The are simple MLPs that operate on the individual pixel level. Each head is compared to the ground truth using a task-specific loss.

Table 1. Forward pass performance measures of **ElasticDino** and competing feature upscaling works over DINOv2. Note that even though **ElasticDino** trains more parameters, it is faster and more memory efficient, due to the linear (as opposed to attention-based) nature of our operations

	ED (ours)	AnyUp	LoftUp	JAFAR	UPLift
Avg Time (ms)	108.41	545.42	418.89	177.79	147.27
Peak mem (GB)	0.71	1.94	8.31	7.33	9.22
#Params (K)	27,095	878	4,303	628	789

Table 2. Validation Mean IoU for semantic segmentation benchmarks. Note that results differ from the comparison benchmarks, as we use a convolutional head (rather than linear) and train for longer than specified in the JAFAR benchmark.

Model	**Cscapes**	**COCO**	**ADE20k**
Oneformer-sw-L	<u>0.6412</u>	<u>0.6264</u>	<u>0.5109</u>
Mask2Former-sw-L	0.5986	0.6239	0.4928
ElasticDino-L	**0.6875**	**0.6270**	**0.5299**

tion, and the CLIP [21] backbone. As can be seen, **ElasticDino** demonstrates leading performance consistently for all datasets except Cityscapes. We interpret this surprising discrepancy to the lower ability for our method to capture small objects, possibly due to the low-frequency bias of the convolutional architecture. To support this class size bias, we provide in the supplementary material a comparison of model performance by object instance size, demonstrating that while ElasticDino vastly outperforms other methods on larger classes, it struggles with smaller ones, indicating both better semantic preservation but also imperfect reconstruction of small objects. Notably, these improved results are produced with less computational efforts and memory footprint, as reported in Table 1.

4.2. Ablation studies

We perform an ablation study on the architecture and design choices. In order to ensure a fair benchmark, we train ElasticDino and its ablations on the Hypersim dataset and omit the depth task, reserving it for evaluation. The training tasks are RGB, illumination, reflectance, residuals and normals [24].

First, we ablate the deformations, employing convolutions only. Second, we ablate tasks in a leave-one-out fashion. Finally, we ablate by progressively omitting tasks.

For each trained upscaler, we train a UNet head to predict the unseen depth, reserving a fraction as a test set (which is also held out from upscaler training), with SiLogLoss as the depth metric. Quantitative results are summarized in table 5, and some qualitative results are shown in figure 5.

Table 3. Comparison of various upsampling methods. All benchmarks are run using the method and code used for the JAFAR paper. Bold = best, underline = second best.

	Semantic Segmentation								Depth Est.	
	COCO		VOC		ADE20K		Cityscapes		COCO	
Method	mIoU↑	Acc↑	mIoU↑	Acc↑	mIoU↑	Acc↑	mIoU↑	Acc↑	δ_1 ↑	RMSE↓
Nearest	56.17	76.97	76.41	93.80	37.27	71.91	54.05	90.36	58.08	0.70
Bilinear	59.03	79.07	80.70	95.17	39.23	73.69	59.37	92.47	59.92	0.66
FeatUp [11]	60.10	79.95	81.08	95.32	38.82	73.74	56.06	91.86	<u>61.69</u>	<u>0.64</u>
LoftUp [13]	<u>61.62</u>	<u>81.11</u>	84.18	96.22	<u>41.51</u>	<u>75.41</u>	<u>62.13</u>	<u>93.53</u>	58.01	0.68
JAFAR [8]	60.32	80.30	84.19	96.18	40.16	74.39	61.58	93.45	60.05	0.65
AnyUp [31]	60.68	80.59	83.81	96.13	40.39	74.76	60.99	93.37	60.65	<u>0.64</u>
UPLift [28]	61.10	80.85	<u>84.82</u>	<u>96.41</u>	41.29	75.05	**65.02**	**94.34**	60.55	<u>0.64</u>
ED (ours)	**62.72**	**81.74**	**86.00**	**96.78**	**42.10**	**76.00**	61.33	93.44	**64.14**	**0.57**

Table 4. Comparison of models over DINOv3-small backbone. Benchmarks are run using the method and code used for the JAFAR [8] paper. Bold = best, underline = second best.

DinoV3	Semantic Segmentation								Depth Est.	
	COCO		VOC		ADE20K		Cityscapes		COCO	
	mIoU(↑)	Acc(↑)	mIoU(↑)	Acc(↑)	mIoU(↑)	Acc(↑)	mIoU(↑)	Acc(↑)	δ_1(↑)	RMSE(↓)
JAFAR [8]	62.2	81.38	82.9	95.96	42.31	75.7	57.25	92.3	58.2	0.68
AnyUp [31]	62.70	81.69	83.88	96.21	42.69	76.02	58.32	92.81	<u>59.43</u>	<u>0.66</u>
UPLift [28]	<u>63.52</u>	<u>82.14</u>	<u>84.86</u>	<u>96.58</u>	<u>44.12</u>	<u>76.62</u>	<u>62.31</u>	<u>93.82</u>	59.29	0.66
ED (ours)	**64.04**	**82.43**	**86.26**	**96.89**	**44.57**	**77.03**	**62.91**	**94.03**	**61.10**	**0.62**

The ablation of deformations results in almost imperceptible differences in feature values: because of the lack of incentive for the network to jointly modify features, task semantics are leaked and baked into the feature values. Deformation-based upsampling preserves feature semantics, allowing only local feature resampling and alleviating catastrophic forgetting. When deformations are ablated, the upsampler can directly alter feature values and may encode only the supervised task signal, e.g. depth, rather than preserving semantic structure. However, because some tasks exhibit correlation with depth prediction (e.g. surface normals), and because residual connections let the features flow through, some ability to transfer is conserved. It can be seen that the model gets to good performance with low amount of tasks but still improves with additional tasks. When ablating each task individually, we see that every task leads to some amount of improvement in performance, showing that adding more tasks improves the performance.

5. Discussion

5.1. Limitations



While demonstrating promising performance, **ElasticDino** struggles with distinguishing small holes in the source image that are smaller than patch size (see inset figure). Our intuition is that the convolutional layers still display low frequency biases especially at large receptive fields, even though leakage is mitigated due to the deformation. Further experimenting with alternative architectures with lower frequency bias may be rewarding.

5.2. Future directions

Beyond solving the above-mentioned issue, a clear future direction is the extension of ElasticDino to other modalities, such as video or 3D. Video could be handled through video-specific feature generators such as VJEPA-2 [1]. 3D could

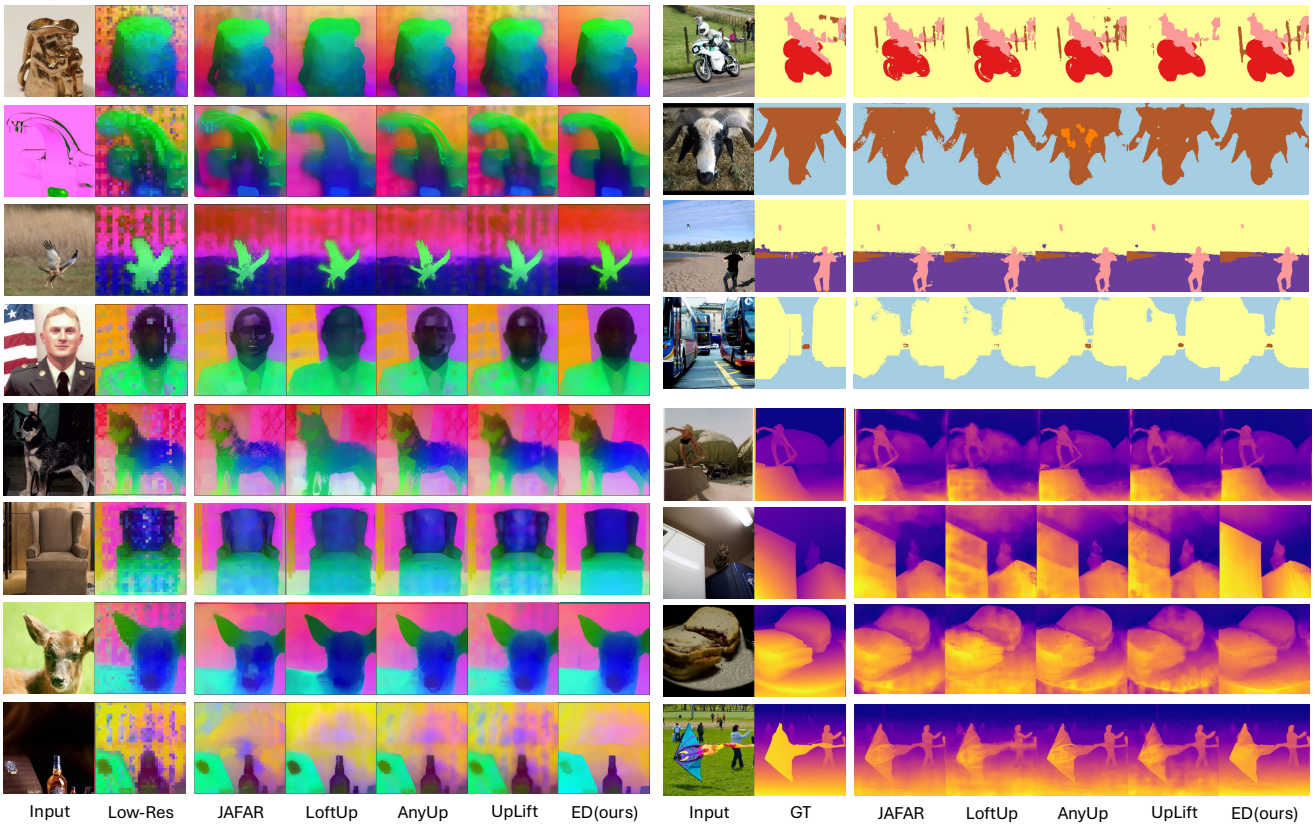


Figure 4. Qualitative evaluation. Left: The top three PCA components of the feature maps of input images are depicted, as generated originally by DINOv2 (Low-Res), previous approaches, and us. Top Right: Semantic segmentation maps of input images, as generated by humans (GT), three previous methods, and us are also depicted. Bottom right: depth maps for the same approaches and input setting are depicted as well. Throughout all results, **ElasticDino** depicts sharper edges that respect the input image resolution and mitigates semantic bleeding between regions, even when RGB colors are similar.

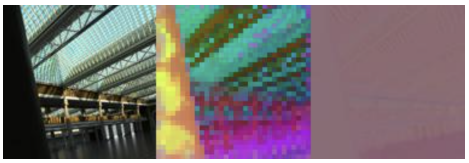


Figure 5. Ablating deformations weakens the preservation of original features by allowing the encoding of the task specific values in the output, instead of only blending them. From left to right: input image, low-resolution DINOv2 features, the result without deformations.

Table 5. Depth-estimation ablation test loss

Ablation	$\delta_1(\%)\uparrow$	$\Delta\delta_1$	$\delta_3(\%)\uparrow$	AbsRel↓	SqRel↓	RMSE↓	ΔRMSE	$\log_{10}\downarrow$	SILog↓
w/o DEFormations	26.80	−31.55	70.11	1.024	1.19	1.32	+0.718	0.238	66.30
w/o ILLumination	57.21	−1.14	92.83	0.428	0.278	0.620	+0.018	0.117	33.06
w/o REFlectance	**58.43**	+0.08	93.10	0.419	0.268	0.603	+0.001	<u>0.115</u>	**32.45**
w/o RESiduals	57.74	−0.61	93.02	0.423	0.273	0.611	+0.009	0.116	32.73
w/o Normals	57.81	−0.54	92.95	0.424	0.274	0.611	+0.009	0.116	32.89
w/o RGB	58.22	−0.13	<u>93.13</u>	<u>0.418</u>	<u>0.266</u>	<u>0.603</u>	+0.001	<u>0.115</u>	32.47
w/o RGB,ILL	57.20	−1.15	92.81	0.425	0.273	0.617	+0.015	0.117	33.24
w/o RGB,ILL,REF	57.17	−1.18	92.73	0.432	0.285	0.622	+0.020	0.118	33.54
w/o RGB,ILL,REF,RES	56.29	−2.06	92.50	0.437	0.292	0.634	+0.032	0.120	33.99
full model	<u>58.35</u>	–	**93.17**	**0.417**	**0.265**	**0.602**	–	**0.115**	<u>32.46</u>

Bold denotes the best result and Underline the second-best result. Rankings are computed from the unrounded values; consequently, identically displayed rounded values may have different ranking markers.

be handled through multi-view projection of features from renderings, as done in TRELLIS [32].

5.3. Conclusion

We have introduced **ElasticDino**, a solution for producing high resolution semantic features on top of vision transformers. We demonstrated **ElasticDino** achieves superior performance compared to existing methods, with lower memory usage and faster runtimes.

ElasticDino, like the vision transformers it enhances, does not produce content that can be directly misused. As a foundational model, any broader impact, positive or negative, is likely to stem from improved performance in downstream tasks or the emergence of new applications.

We refer to the supplementary material for additional implementation details and qualitative results.

References

[1] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025. 7

[2] Aleksei Bochkovskii, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R. Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second, 2025. 1, 2, 4

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In Proceedings of the IEEE/CVF international conference on computer vision, pages 9650–9660, 2021. 1, 2

[4] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions, 2023. 2

[5] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. arXiv, 2021. 1, 2, 5

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017. 4

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016. 5

[8] Paul Couairon, Loick Chambon, Louis Serrano, Jean-Emmanuel Haugeard, Matthieu Cord, and Nicolas Thome. Jafar: Jack up any feature at any resolution, 2025. 1, 3, 5, 7, 12, 13

[9] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers, 2024. 2, 3

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009. 5

[11] Stephanie Fu, Mark Hamilton, Laura E. Brandt, Axel Feldmann, Zhoutong Zhang, and William T. Freeman. Featup: A model-agnostic framework for features at any resolution. In The Twelfth International Conference on Learning Representations, 2024. 1, 2, 5, 7

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2

[13] Haiwen Huang, Anpei Chen, Volodymyr Havrylov, Andreas Geiger, and Dan Zhang. Loftup: Learning a coordinate-based feature upsampler for vision foundation models. arXiv preprint arXiv:2504.14032, 2025. 1, 3, 5, 7

[14] Jitesh Jain, Jiachen Li, MangTik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. Oneformer: One transformer to rule universal image segmentation, 2022. 1, 2, 5

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 5

[16] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection, 2024. 1, 2

[17] Wenze Liu, Hao Lu, Hongtao Fu, and Zhiguo Cao. Learning to upsample by learning to sample. In Proceedings of the IEEE/CVF international conference on computer vision, pages 6027–6037, 2023. 2

[18] Hao Lu, Wenze Liu, Zixuan Ye, Hongtao Fu, Yuliang Liu, and Zhiguo Cao. Sapa: Similarity-aware point affiliation for feature upsampling. Advances in Neural Information Processing Systems, 35:20889–20901, 2022. 2

[19] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193, 2023. 1, 2

[20] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. 2

[21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 2, 6

[22] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, 2020. 1, 2

[23] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction, 2021. 1, 2

[24] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding, 2021. 5, 6

[25] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025. 1, 2, 5

[26] Saksham Suri, Matthew Walmer, Kamal Gupta, and Abhinav Shrivastava. Lift: A surprisingly simple lightweight feature transform for dense vit descriptors, 2024. 3

[27] Narek Tumanyan, Assaf Singer, Shai Bagon, and Tali Dekel. Dino-tracker: Taming dino for self-supervised point tracking in a single video, 2024. 1

[28] Matthew Walmer, Saksham Suri, Anirud Aggarwal, and Abhinav Shrivastava. Uplift: Efficient pixel-dense feature upsampling with local attenders. arXiv preprint arXiv:2601.17950, 2026. 1, 3, 5, 7

[29] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4):600–612, 2004. 4

[30] Zirui Wang, Jiahui Yu, Adams Wei Yu, Zihang Dai, Yulia Tsvetkov, and Yuan Cao. Simvlm: Simple visual language model pretraining with weak supervision, 2022. 2

[31] Thomas Wimmer, Prune Truong, Marie-Julie Rakotosaona, Michael Oechsle, Federico Tombari, Bernt Schiele, and Jan Eric Lenssen. Anyup: Universal feature upsampling. arXiv preprint arXiv:2510.12764, 2025. 1, 3, 5, 7, 12

[32] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation, 2025. 8

[33] Jiawei Yang, Katie Z Luo, Jiefeng Li, Congyue Deng, Leonidas Guibas, Dilip Krishnan, Kilian Q Weinberger, Yonglong Tian, and Yue Wang. Denoising vision transformers, 2024. 2

[34] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data, 2024. 1, 2

[35] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2, 2024. 1, 2

[36] Sukmin Yun, Hankook Lee, Jaehyung Kim, and Jinwoo Shin. Patch-level representation learning for self-supervised vision transformers, 2022. 2

[37] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2022. 2

[38] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 5

[39] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through ade20k dataset. International Journal of Computer Vision (IJCV), 2018. 5

[40] Minghao Zhou, Hong Wang, Yefeng Zheng, and Deyu Meng. A refreshed similarity-based upsampler for direct high-ratio feature upsampling. IEEE Transactions on Neural Networks and Learning Systems, 2025. 2

ElasticDino: Dense Semantic Features using Elastic Deformation - Supplementary Material

A. Implementation Details

We train our model using batches drawn alternately from two datasets: Hypersim [24], which provides ground truth for intrinsic decomposition and normal estimation, and ImageNet-22k [10] for increased image diversity.

To avoid introducing additional datasets for semantic segmentation and depth prediction, we utilize pseudo-labels generated by pre-trained models, specifically, Mask2Former (COCO,ViT-Large) [5] and DepthAnything [34] respectively. These labels are computed for images across both of the training datasets.

To minimize parameter count and overhead on top of the base ViT prediction, we use depth-wise separable convolutions [6]. We observe no performance hit compared to regular convolutions, with significantly faster training and inference, and vastly reduced memory usage.

We train in mixed-precision using an 8-bit version of the AdamW optimizer [51], with parameters $\epsilon = 10^{-5}$ for stability, an effective batch size of 16 and an exponential learning rate decay starting at 10^{-3}. The final model was trained on a single A100 GPU for 24 hours, amounting to 3.42kg of CO2-equivalent emissions, fully offset by the cloud provider (Google Cloud Services), and computed using the MachineLearning Impact calculator presented in [49].

B. Further Quantitative Results

B.1. Open Vocabulary Evaluation

We evaluate our model on zero-Shot open-vocabulary segmentation following the method presented in MaskCLIP [58]. This training-free approach computes pixel-wise predictions by taking the argmax over similarity scores between visual features and text embeddings generated from dataset labels in textual templates. Following JAFAR [8], we evaluate on CLIP-ViT-B/16 and observe competitive results, demonstrating that **ElasticDino** preserves the original backbone features through its upsampling. Results against competitors that are compatible with CLIP-ViT-B/16 (JAFAR [8], AnyUp [31], LoftUp [13]) are reported in Table 1.

Table 1. Zero-Shot Open-Vocabulary segmentation on VOC and Ade20k datasets following the methodology used in JAFAR [8] and introduced in MaskCLIP [58] paper. Competitive results indicating good alignment with original internal CLIP features.

Method	VOC		ADE20K	
	mIoU(↑)	Acc(↑)	mIoU(↑)	Acc(↑)
Bilinear	51.25	68.18	9.05	25.15
JAFAR	**63.99**	**77.91**	<u>10.69</u>	28.06
AnyUp	57.59	73.40	10.71	**28.95**
LoftUp	38.73	55.29	7.64	32.16
ElasticDino (ours)	<u>58.43</u>	<u>73.89</u>	**10.75**	<u>28.28</u>

B.2. Positional Information Preservation

While positional encoding related artifacts are alleviated, positional information still persists and can be extracted more easily from the features compared to competing upscaling methods. We train a simple pixel-wise head to predict normalized (x, y) coordinates from upsampled features. Bilinear upsampling performs best, as expected for a linear target, but **ElasticDino** indeed preserves positional information. See accuracy in [10^{-3} MSE] results below.

	JAFAR	FeatUp	AnyUp	LoftUp	Bilinear	Ours
$10^{-3}MSE$	5.3	11.5	4.2	64.0	3.2	4.5

B.3. Semantic Segmentation and Depth Estimation over CLIP

Following the comparison against other task agnostic upsampling methods (JAFAR [8], AnyUp [31], LoftUp [13], UPLift [28]) over DINOv2 [19] and DINOv3 [25], we also compare recent works over CLIP [58] and get competitive results (Table 2). We follow established evaluation protocols, as described by JAFAR [8], measuring mean Intersection over Union (mIoU) and pixel-level accuracy (Acc) for the semantic segmentation task over four datasets, and Root Mean Square Error (RMSE) and δ_1 (measuring the percentage of pixels with a relative error under 25% of the ground-truth) for the depth estimation task over one dataset.

Table 2. Comparison of models over CLIP backbone. Benchmarks are run using the method and code used for the JAFAR [8] paper. Bold = best, underline = second best.

CLIP	Semantic Segmentation								Depth Est.	
	COCO		VOC		ADE20K		Cityscapes		COCO	
	mIoU(↑)	Acc(↑)	mIoU(↑)	Acc(↑)	mIoU(↑)	Acc(↑)	mIoU(↑)	Acc(↑)	δ_1(↑)	RMSE(↓)
JAFAR [8]	**58.16**	**78.52**	<u>79.07</u>	**94.69**	**36.65**	**71.39**	**50.98**	**91.23**	<u>45.75</u>	<u>0.92</u>
AnyUp [31]	53.05	74.54	69.85	91.46	31.16	67.35	39.63	86.62	44.81	0.96
ED (ours)	<u>57.86</u>	<u>78.23</u>	**79.26**	<u>94.67</u>	<u>35.5</u>	<u>71.22</u>	<u>46.21</u>	<u>90.67</u>	**48.20**	**0.86**

B.4. Semantic Segmentation and Depth Estimation for no-seg and no-depth ElasticDino Variants

We evaluated **ElasticDino** variants trained without the depth and the segmentation tasks in the same framework used to evaluate the full variant against task agnostic competitors over DINOv2 [19]. This is to strengthen the claim that **ElasticDino** only uses these tasks for structural guidance and does not rely on task specific information. Performance remains strong on these held-out tasks (Table 3).

C. Further Qualitative Results

In this section we offer additional qualitative results.

Dedicated Segmentation models. We evaluate **ElasticDino** against two leading dedicated segmentation specific approaches Mask2Former [5] and OneFormer [14] over the Cityscapes[7] dataset. We showcase that even though our approach is task agnostic we surpass leading dedicated segmentation models using only our lightweight operations over the DINOv2[19] backbone, see Figure 2.

Training heads. **ElasticDino** employs different tasks to train the upscaler, by co-training a set of local and minimalistic 1×1 convolution heads, each performing a different dense task, considering pixels individually. In Figure 8, we demonstrate the results of these seven heads, both in their final step, where the features are of 256×256 resolution, and in an intermediate point in the network, where the resolution is upscaled from 32×32 to only 64×64. We further demonstrate qualitative results for intrinsic decomposition using the probe heads over the validation set of HyperSim [24] in Figure 1.

PCA feature visualization. In the feature maps comparison figures (Figures 6 and 7), we visualize the upscaled feature maps of **ElasticDino** using the first three components of a PCA decomposition and those of JAFAR [8], AnyUp [31], LoftUp [13], UPLift [28]. As can be seen, our method achieves crisp features and cleaner maps from different artifacts as opposed to the competitors. It can be noticed that **ElasticDino** is better at removing artifacts originating from spatial-encodings and separates far and near objects (the bars of the forklift) as well as appearance from geometry (the shield).

Visualization of deformations along upscaling process. Figures 4, 5 visualize the cumulative deformations through the process of feature upscaling for different images. Deformations are visualized using color wheel optical flow [41] and features are visualized through the first three PCA components. It can be seen that deformations help to make the edges more crisp and reduce artifacts in the feature maps.

Linear segmentation and depth probe heads. Finally, we also compare **ElasticDino** against JAFAR [8], AnyUp [31], LoftUp [13] in the framework suggested by JAFAR where linear probe heads are trained over the upsamplers for segmentation and depth tasks displaying the information encoded inside the upsampled features produced by the model allows achieving top results for dense tasks such as semantic segmentation (Figures 10, 12, 9, 11) and depth estimation (Figure 13). It can be seen that **ElasticDino** achieves superior results with cleaner edges and often detects details other models are not able to.

D. Experiments to understand degraded performance over Cityscapes

We investigated our change in relative performance on Cityscapes [7], in contrast to all other datasets. We hypothesize it is because Cityscapes consists mainly of smaller objects and segments, connecting to our known limitation of filling "holes" in feature maps. To further investigate this we evaluated AP (Average Precision) metric for the different models over segments of different size categories Small (area $< 32^2$), Medium ($32^2 <$ area $< 96^2$), Large (area $> 96^2$) as taken from COCO [15] instance segmentation evaluation methodology (see Figure 3). We also mea-

Table 3. Comparison of no-seg, no-depth **ElasticDino** variants against fully trained model. Benchmarks are run using the method and code used for the JAFAR[8] paper. Bold = best, underline = second best.

DINOv2	**Semantic Segmentation**								**Depth Est.**	
	COCO		VOC		ADE20K		Cityscapes		COCO	
Method	mIoU↑	Acc↑	mIoU↑	Acc↑	mIoU↑	Acc↑	mIoU↑	Acc↑	δ_1 ↑	RMSE↓
ED (no-depth variant)	61.7	81.1	84.1	96.3	41.3	75.5	<u>60.1</u>	93.1	60.5	0.64
ED (no-seg variant)	<u>62.5</u>	<u>81.6</u>	<u>85.9</u>	**96.8**	<u>41.5</u>	<u>75.8</u>	59.8	<u>93.3</u>	**64.8**	**0.56**
ED (full variant)	**62.72**	**81.74**	**86.00**	<u>96.78</u>	**42.10**	**76.00**	61.33	93.44	<u>64.14</u>	<u>0.57</u>

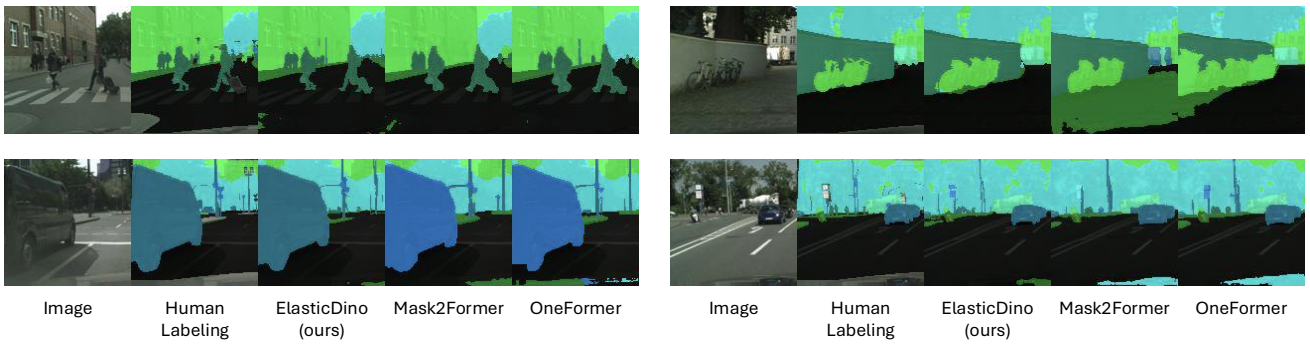


Figure 1. Qualitative comparison of **ElasticDino** to dedicated segmentation approaches (Mask2Former [5],OneFormer [14]) over the cityScapes dataset.

sured distribution of different segment size categories over the COCO[15], Cityscapes[7], Ade20k datasets. From this can see that the lower performance of **ElasticDino** can over Cityscapes can be explained by its lower performance on small thin objects present in this dataset in particular.

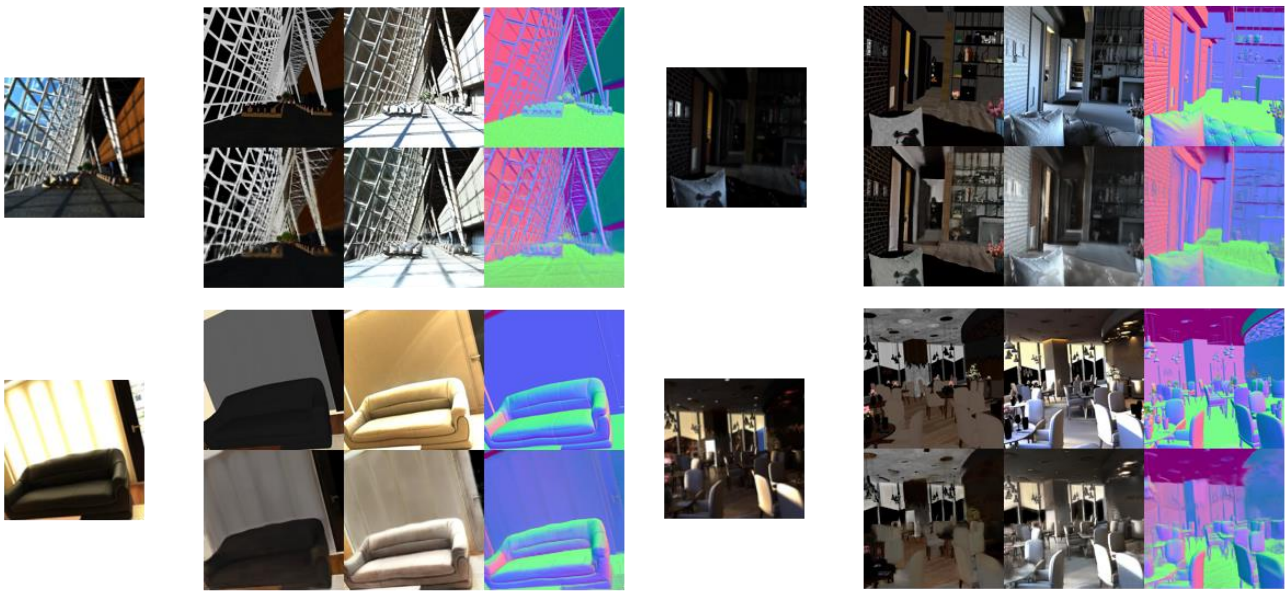


Figure 2. Intrinsic decomposition estimation trained on Hypersim [24] with a convolutional head at resolution 128x128. For each image, the decomposition is shown on the right: top row is the ground truth and bottom row is the prediction from ElasticDino. Left to right: albedo, diffuse shading, surface normals. Even in poor light conditions, we obtain strong qualitative results.

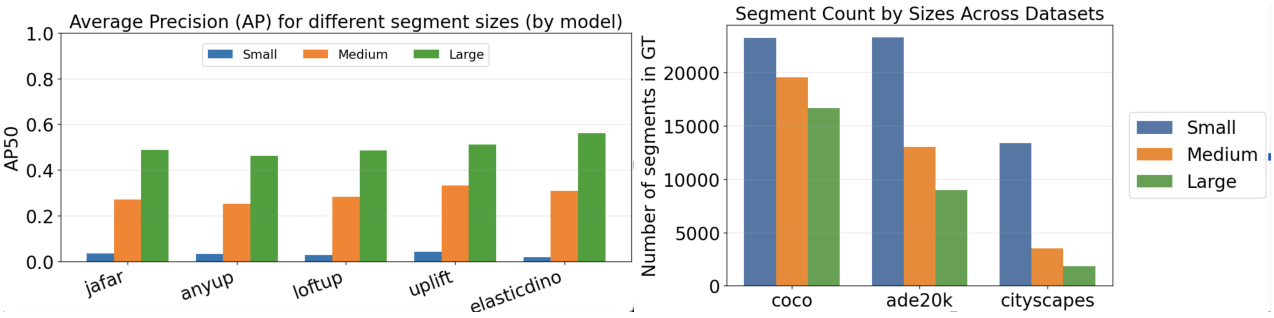
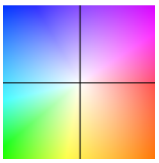


Figure 3. Left: Accuracy vs size. We perform better for larger segments. Right: sizes across datasets. Cityscapes consists of smaller objects.



Flow field color coding

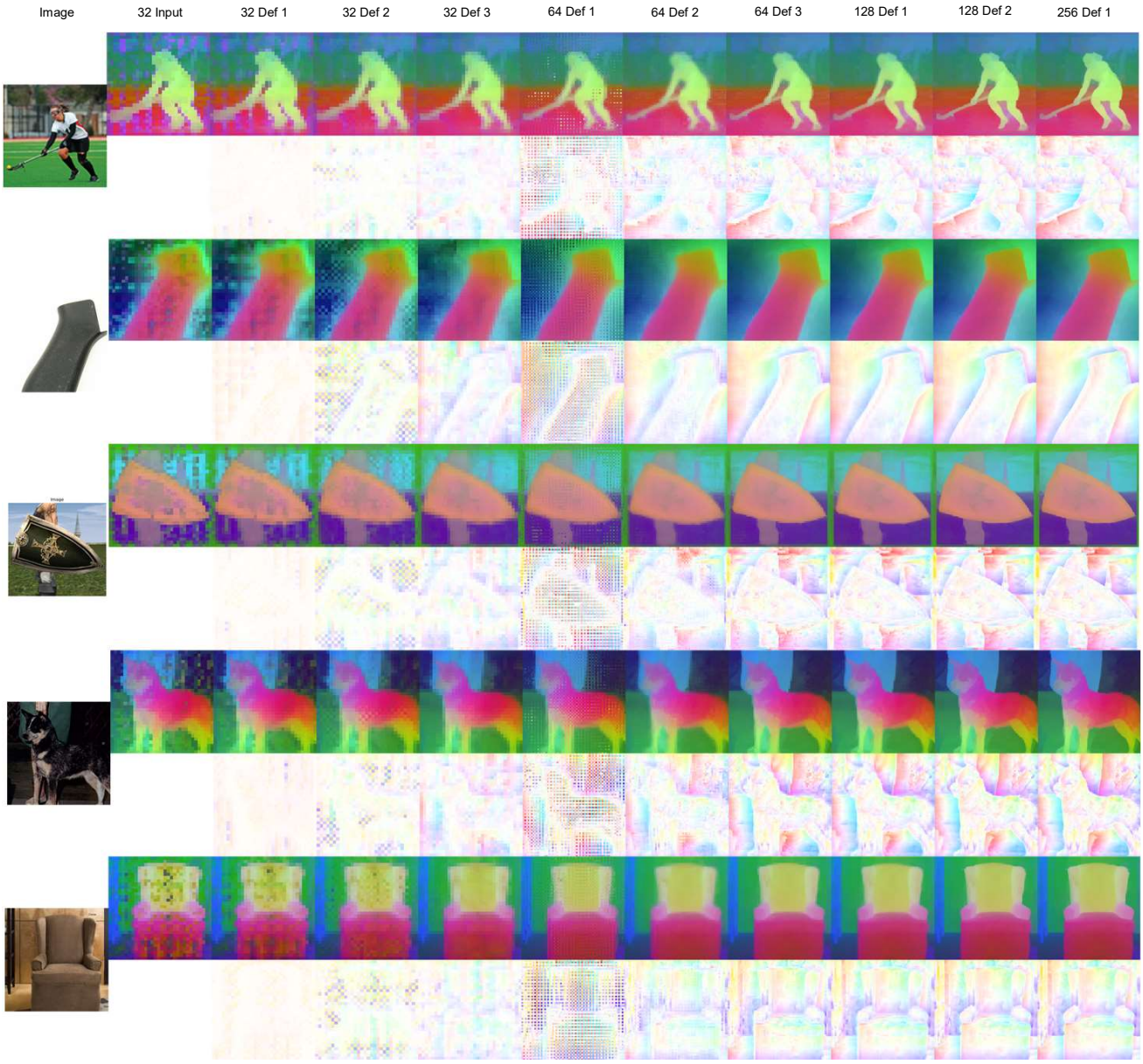
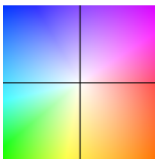


Figure 4. Visualization of deformations along the different resolution stages of the upscaling process and their effect on the features while being upscaled. Deformations are visualized using color wheel optical flow [41], showing where each pixel was sampled from. Features are visualized through the first three PCA components. For each image, top row shows the feature maps and bottom row shows the cumulative deformations so far. Zoomed-in viewing recommended.



Flow field color coding

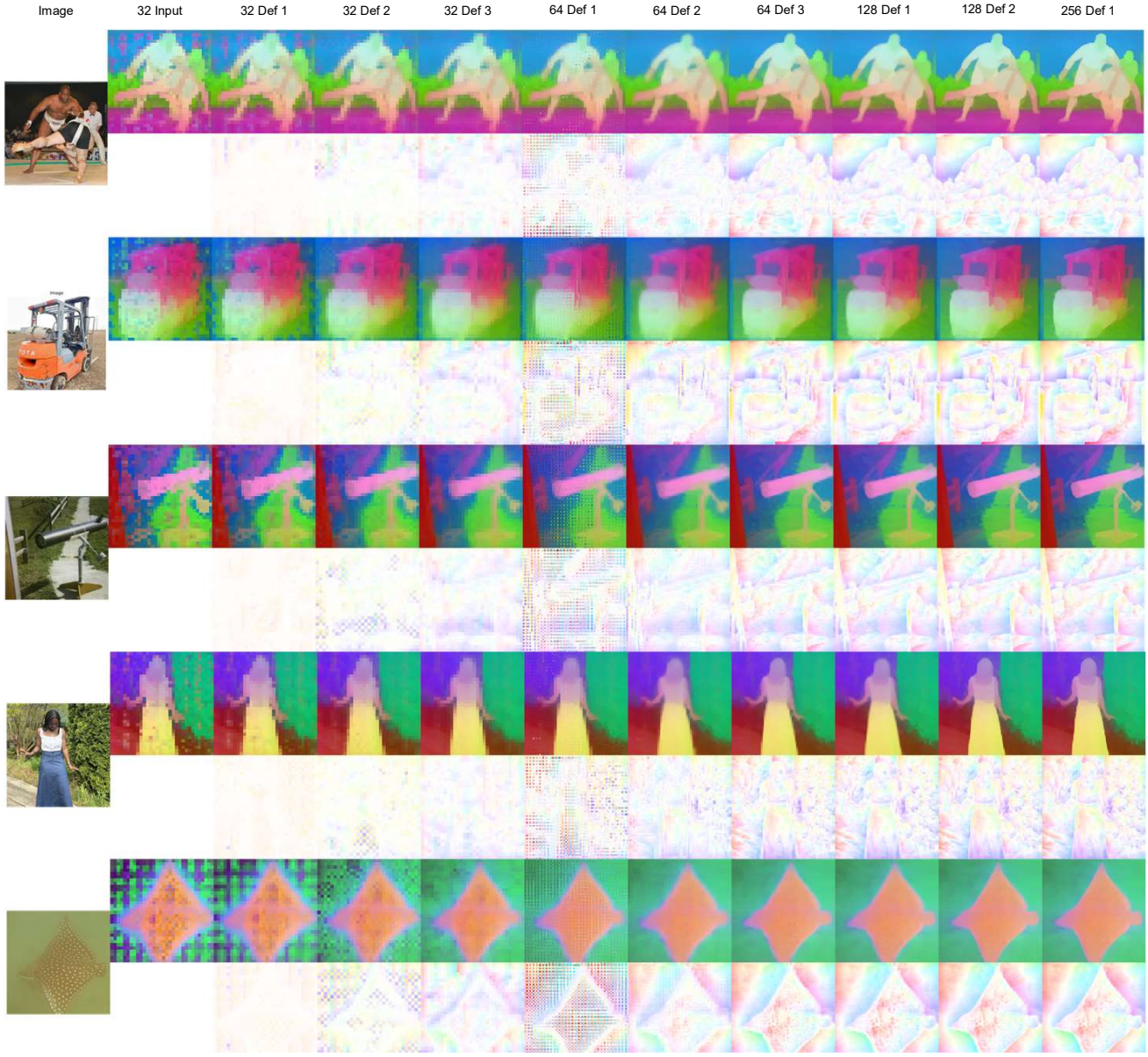


Figure 5. Visualization of deformations along the different resolution stages of the upscaling process and their effect on the features while being upscaled. Deformations are visualized using color wheel optical flow [41], showing where each pixel was sampled from. Features are visualized through the first three PCA components. For each image, top row shows the feature maps and bottom row shows the cumulative deformations so far. Zoomed-in viewing recommended.

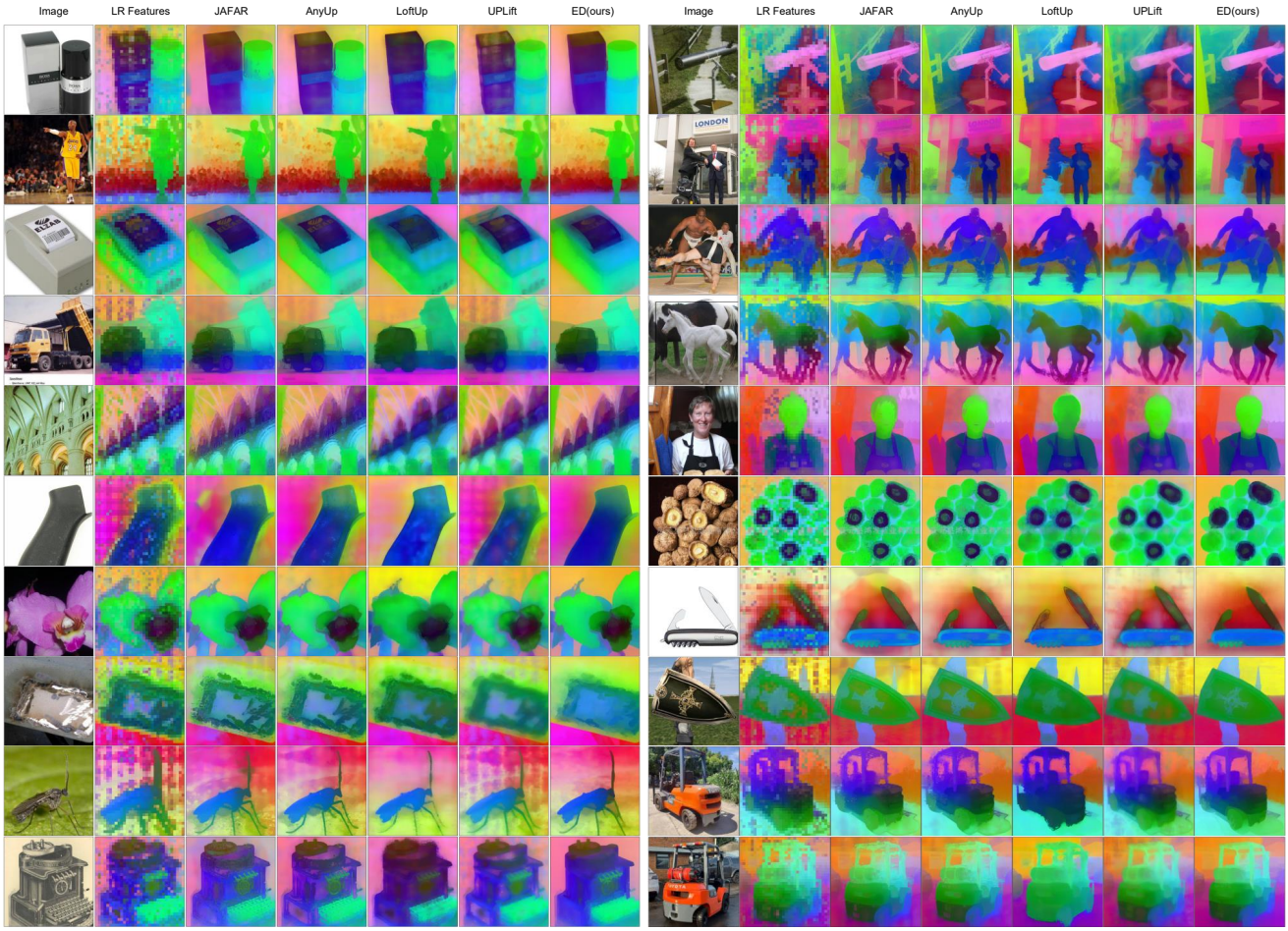


Figure 6. Additional qualitative comparisons through visualizations of the first three PCA components of the resulting features. From left to right: Input image, Original DINOv2 low-res features, JAFAR [8], Anyup [31], LoftUp [13], UPLift [28], ours. Zoomed-in viewing recommended.

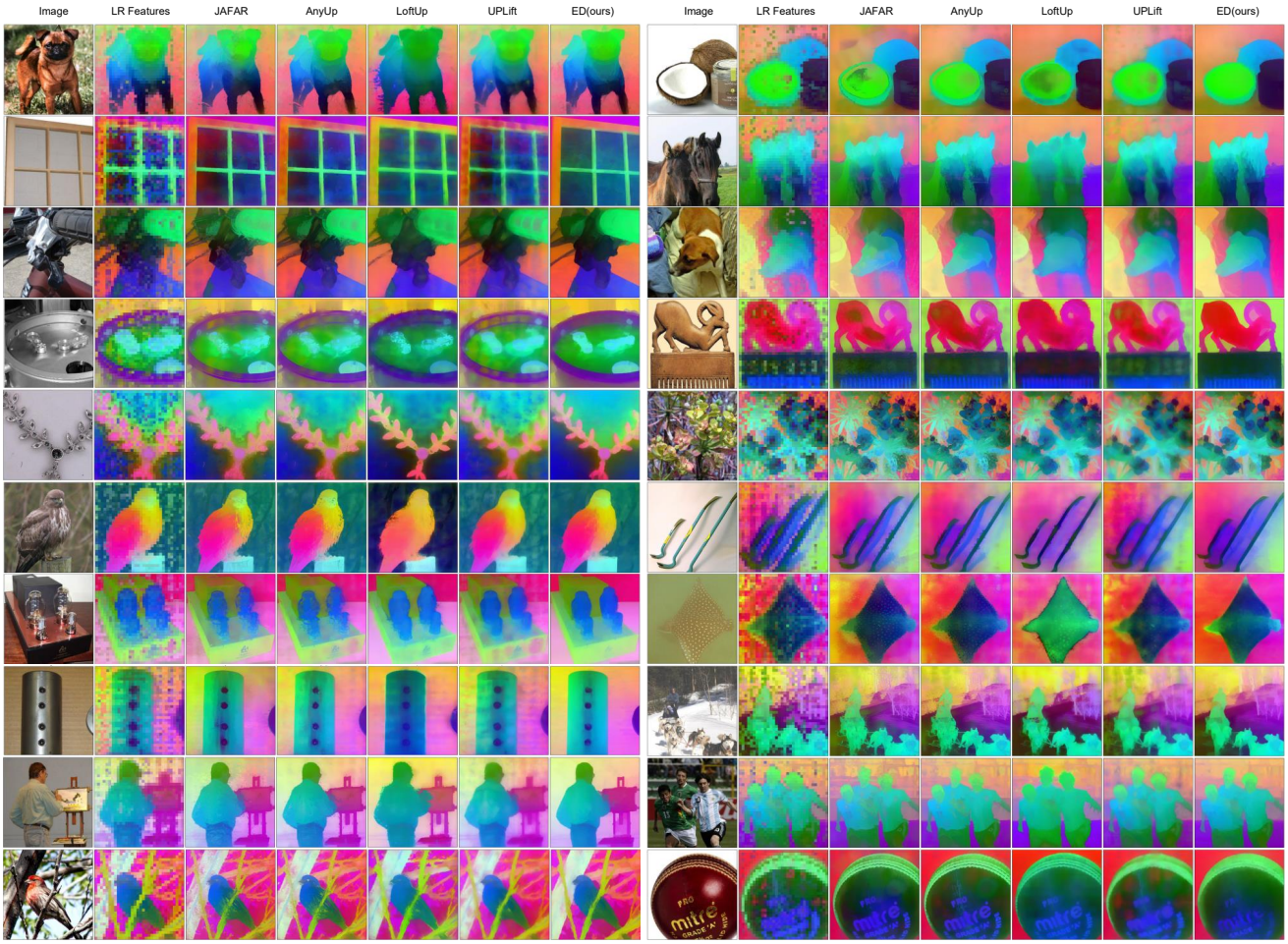


Figure 7. Additional qualitative comparisons through visualizations of the first three PCA components of the resulting features. From left to right: Input image, Original DINOv2 low-res features, JAFAR [8], Anyup [31], LoftUp [13], UPLift [28], ours. Zoomed-in viewing recommended.

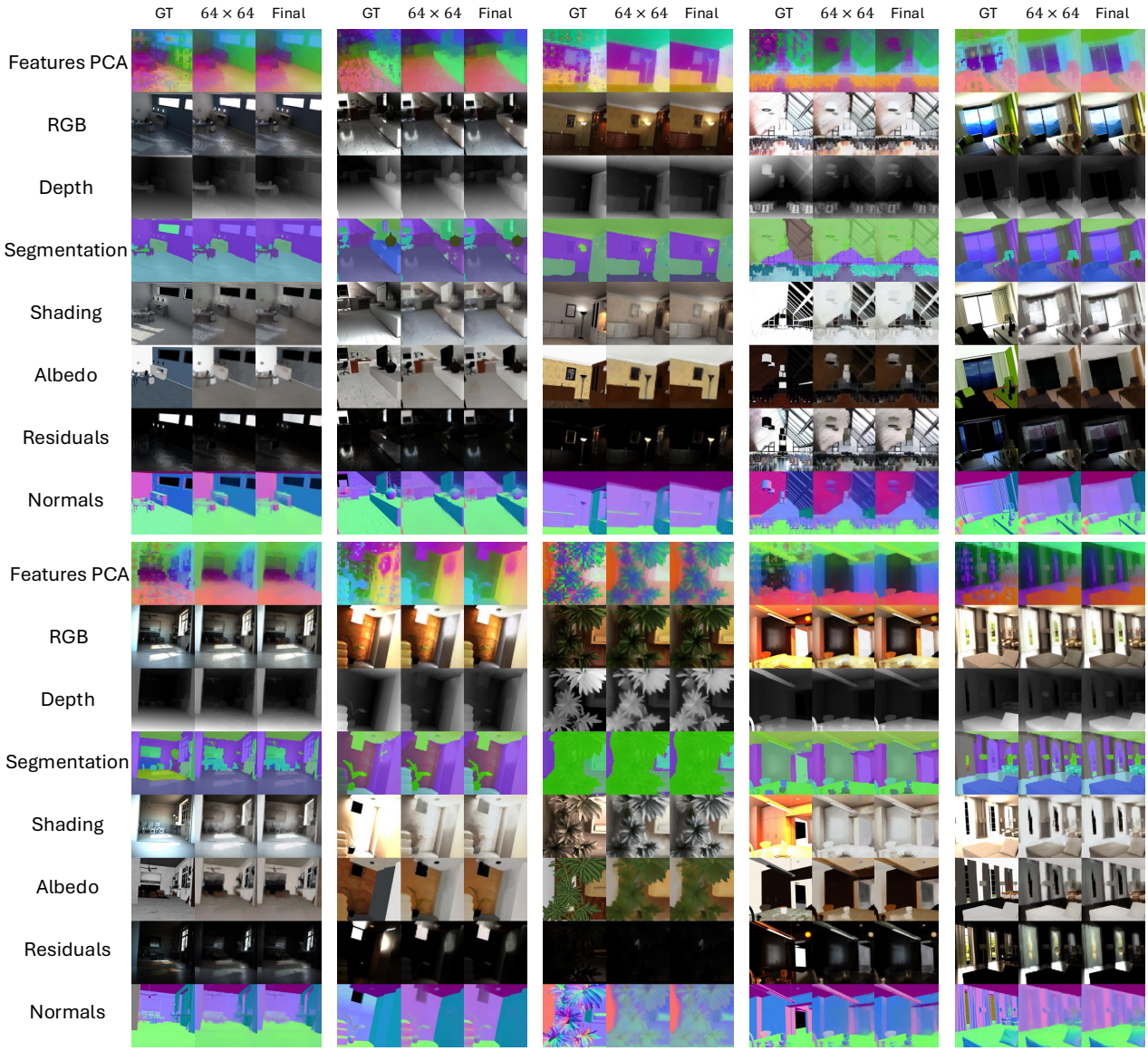


Figure 8. Qualitative evaluation of our auxiliary training tasks. During training, our encoder is trained various tasks, each with an individual head and the minimal receptive field of one pixel. Their results are demonstrated here. Top to bottom: PCA of the upscaled features, RGB reconstruction, depth estimation, semantic segmentation, and four tasks of intrinsic decomposition (shading, albeo, residuals, normals). For each image, we show (left) its GT labels (or the features before upscaling for the PCA), intermediate results of our heads after upscaling to size 64×64 (middle), and the final results of size 256×256 (right). Zoomed-in viewing recommended.

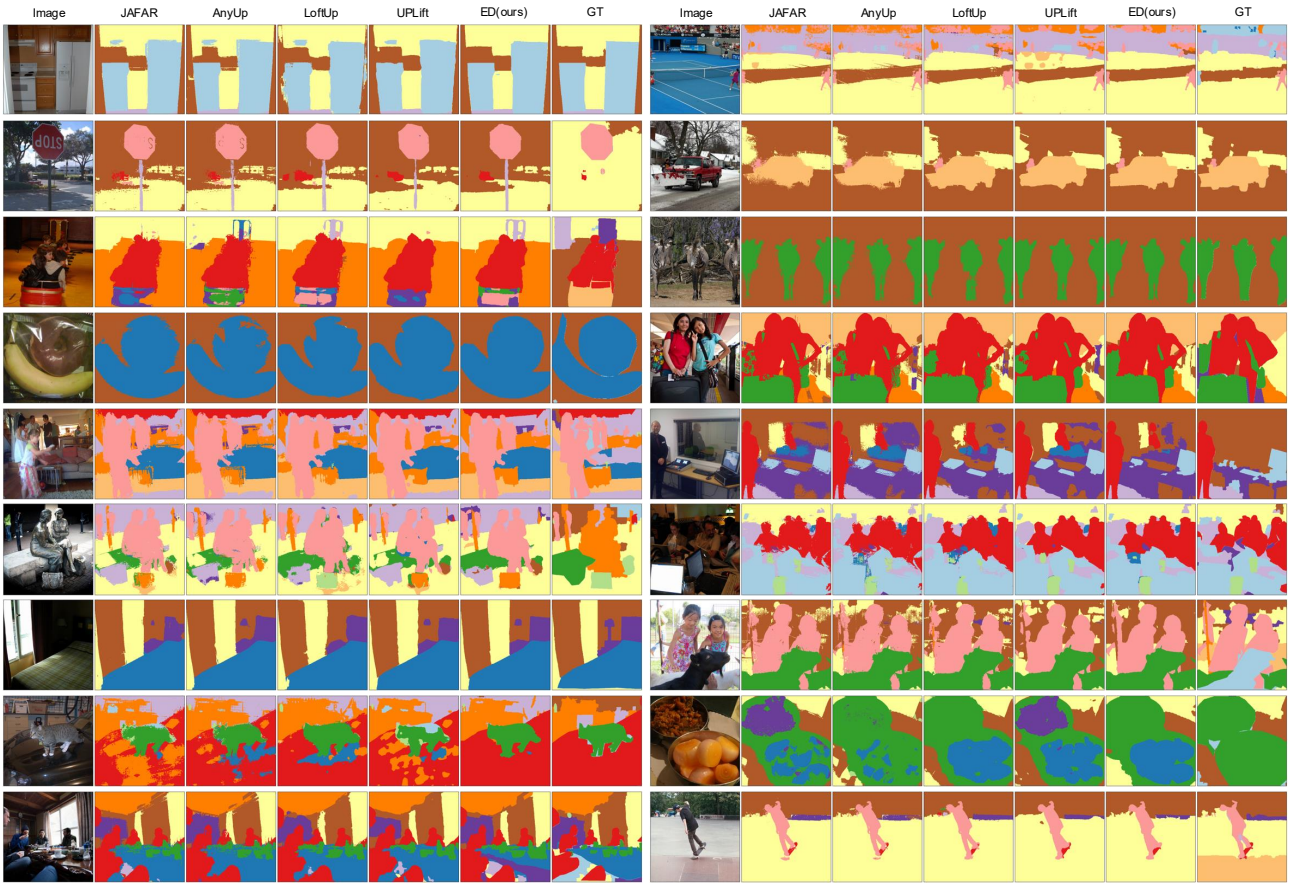


Figure 9. Additional qualitative comparisons for the segmentation task over the COCO dataset. From left to right: Input image, JAFAR [8], Anyup [31], LoftUp [13], UPLift [28], ours, Ground Truth. Zoomed-in viewing recommended.

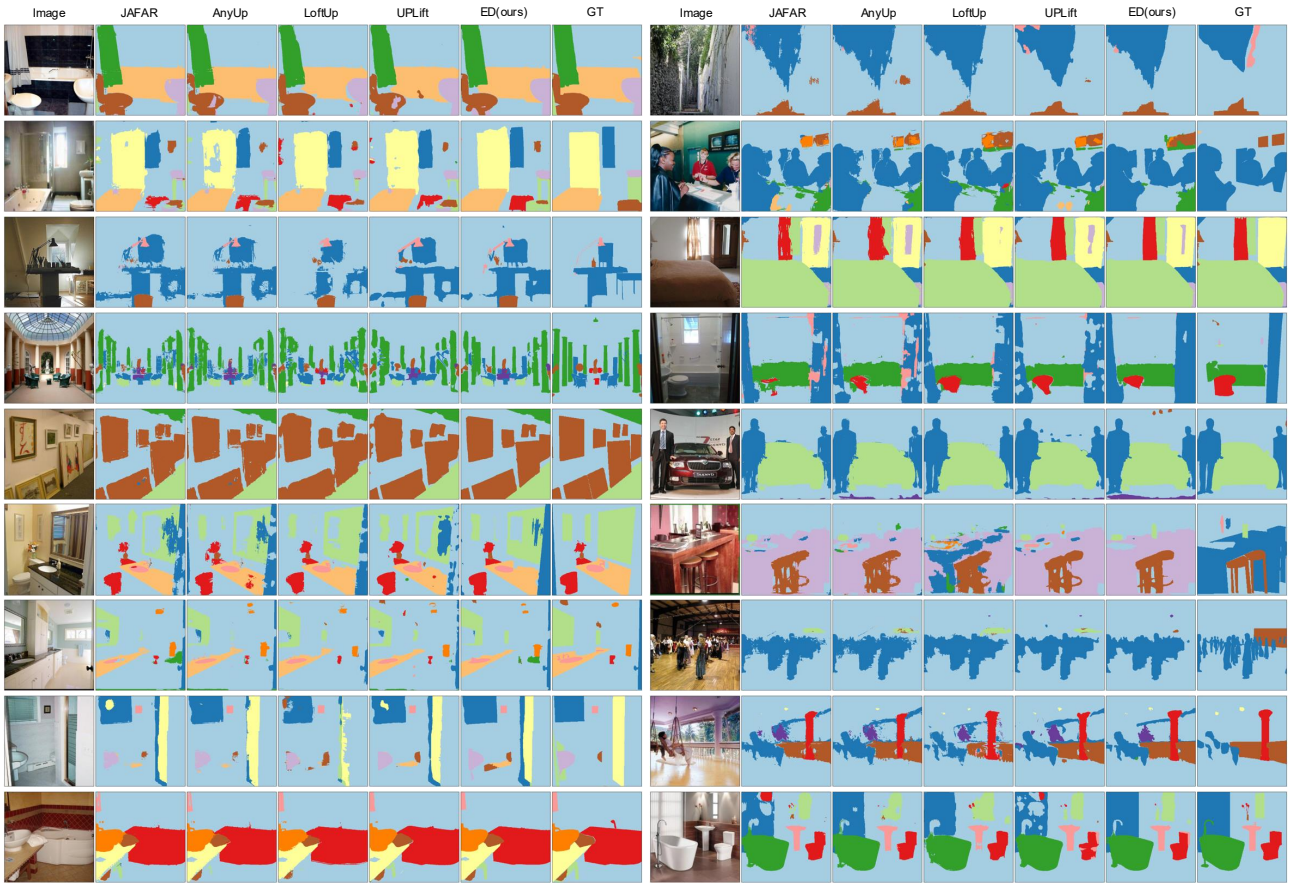


Figure 10. Additional qualitative comparisons for the segmentation task over the ADE20K dataset. From left to right: Input image, JAFAR [8], Anyup [31], LoftUp [13], UPLift [28], ours, Ground Truth. Zoomed-in viewing recommended.



Figure 11. Additional qualitative comparisons for the segmentation task over the VOC dataset. From left to right: Input image, JAFAR [8], Anyup [31], LoftUp [13], UPLift [28], ours, Ground Truth. Zoomed-in viewing recommended.

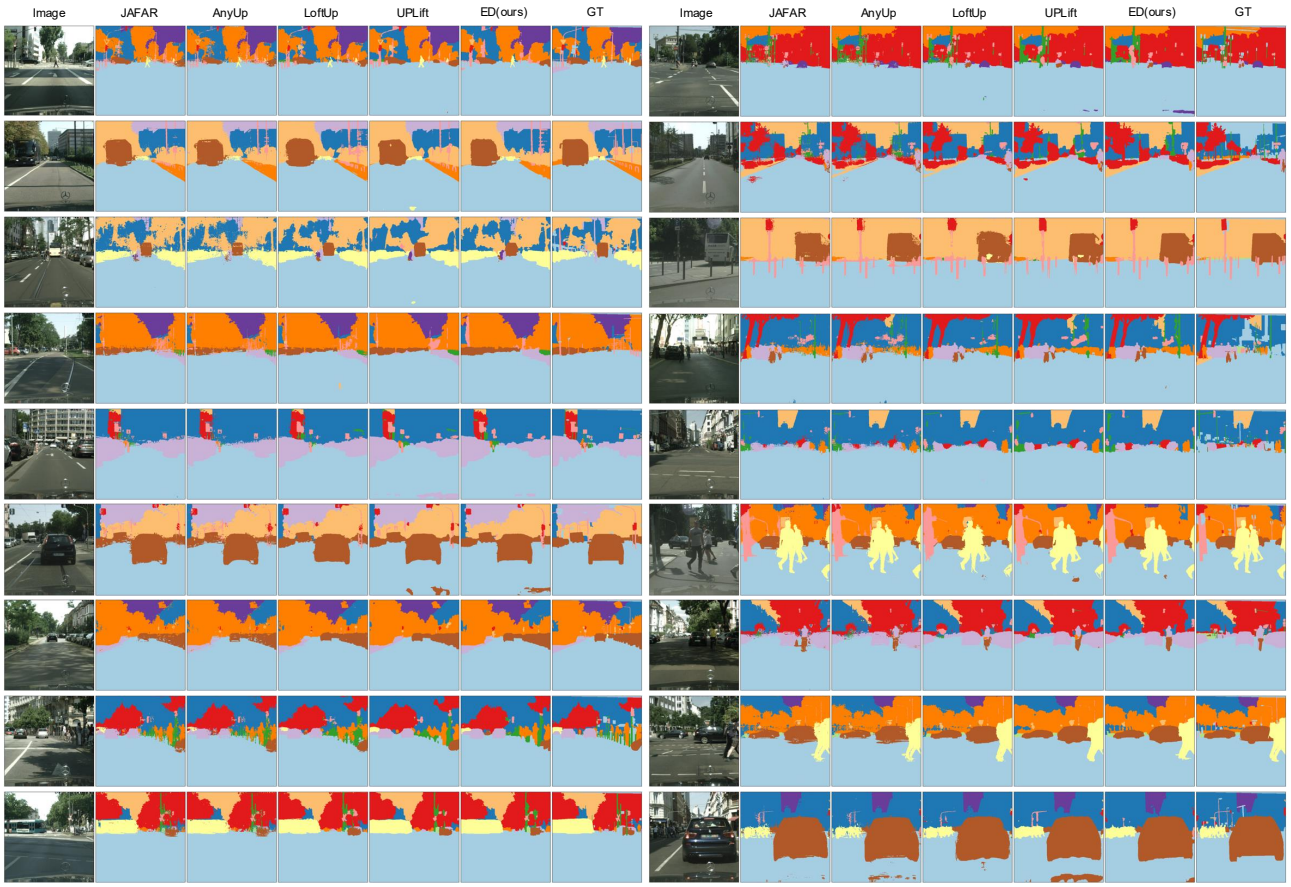


Figure 12. Additional qualitative comparisons for the segmentation task over the CityScapes dataset. From left to right: Input image, JAFAR [8], Anyup [31], LoftUp [13], UPLift [28], ours, Ground Truth. Zoomed-in viewing recommended.

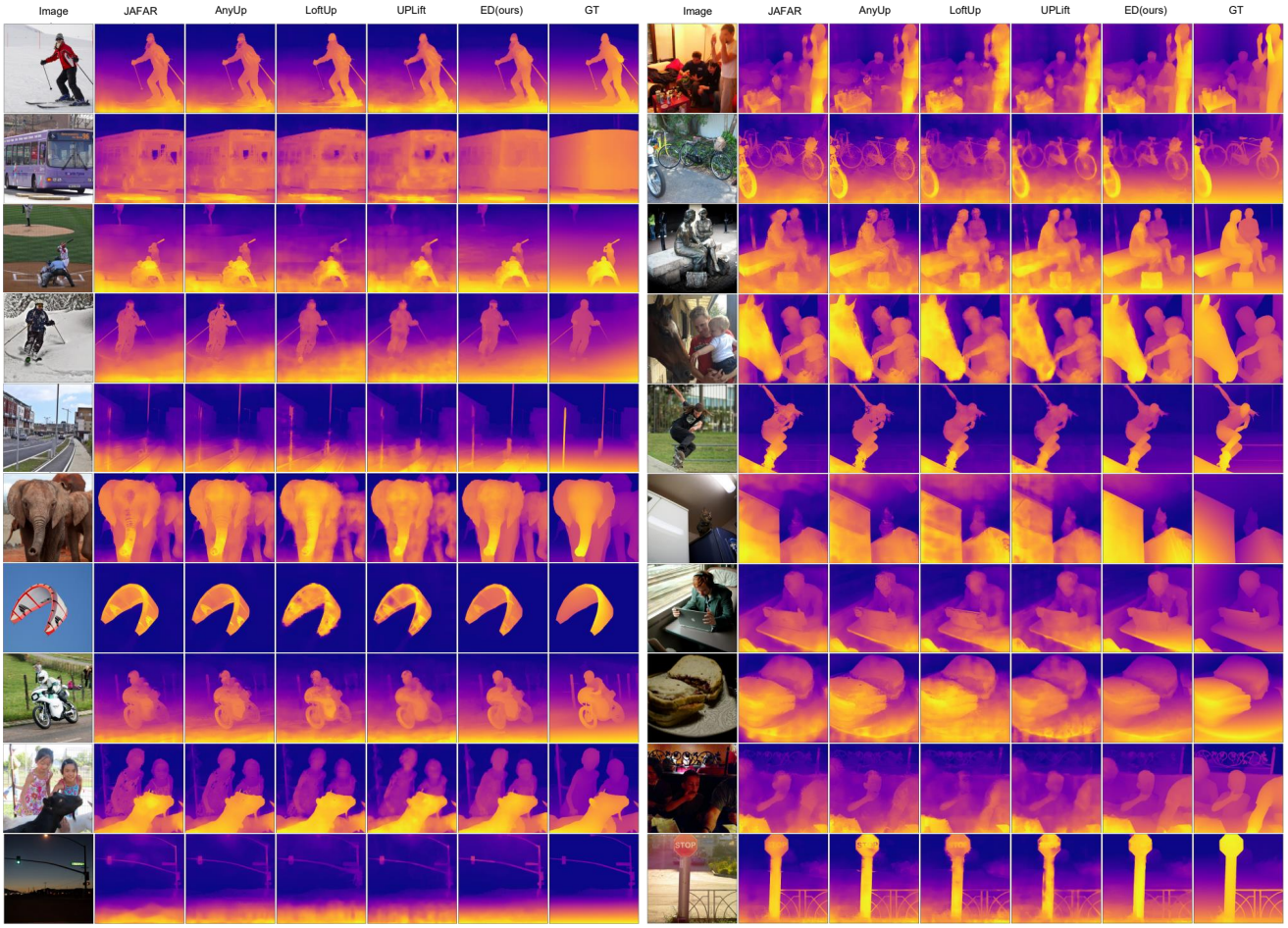


Figure 13. Additional qualitative comparisons for the depth estimation task over the COCO dataset. From left to right: Input image, JAFAR [8], Anyup [31], LoftUp [13], UPLift [28], ours, Ground Truth. Zoomed-in viewing recommended.

References

[41] Simon Baker, Daniel Scharstein, James P Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. International journal of computer vision, 92(1):1–31, 2011.

[5] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. arXiv, 2021. 1, 2, 5

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017. 4

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016. 5

[8] Paul Couairon, Loick Chambon, Louis Serrano, Jean-Emmanuel Haugeard, Matthieu Cord, and Nicolas Thome. Jafar: Jack up any feature at any resolution, 2025. 1, 3, 5, 7, 12, 13

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009. 5

[13] Haiwen Huang, Anpei Chen, Volodymyr Havrylov, Andreas Geiger, and Dan Zhang. Loftup: Learning a coordinate-based feature upsampler for vision foundation models. arXiv preprint arXiv:2504.14032, 2025. 1, 3, 5, 7

[14] Jitesh Jain, Jiachen Li, MangTik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. Oneformer: One transformer to rule universal image segmentation, 2022. 1, 2, 5

[49] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. arXiv preprint arXiv:1910.09700, 2019.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 5

[51] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[19] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193, 2023. 1, 2

[24] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding, 2021. 5, 6

[25] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025. 1, 2, 5

[28] Matthew Walmer, Saksham Suri, Anirud Aggarwal, and Abhinav Shrivastava. Uplift: Efficient pixel-dense feature upsampling with local attenders. arXiv preprint arXiv:2601.17950, 2026. 1, 3, 5, 7

[31] Thomas Wimmer, Prune Truong, Marie-Julie Rakotosaona, Michael Oechsle, Federico Tombari, Bernt Schiele, and Jan Eric Lenssen. Anyup: Universal feature upsampling. arXiv preprint arXiv:2510.12764, 2025. 1, 3, 5, 7, 12

[34] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data, 2024. 1, 2

[58] Chong Zhou, Chen Change Loy, and Bo Dai. Extract free dense labels from clip. In European Conference on Computer Vision (ECCV), 2022.