
FlowFeat: Pixel-Dense Embedding of Motion Profiles

* Supplemental Material *

Nikita Araslanov^{1,2}

Anna Sonnweber¹

Daniel Cremers^{1,2}

¹TU Munich ²MCML

A Qualitative Examples

Our supplemental material is accompanied by a qualitative video comparison. We select multiple validation sequences from DAVIS-2017 [40] and run linear probing using DINO2-S14 [37] as the baseline encoder.

- `a_vos_featup.mp4` compares FlowFeat against FeatUp [15]. The videos also visualise the ground-truth segmentation masks and the first three principal components of FlowFeat representation. FlowFeat achieves compelling segmentation accuracy, remains sharp at object boundaries and more temporally stable than FeatUp.
- `b_pca_featup.mp4` inspects the first three principal components of FlowFeat and FeatUp for two input resolutions: 224×400 and 480×854 . We observe that FlowFeat scales gracefully and reveals a greater level of detail. By contrast, FeatUp struggles to adapt to the higher resolution and exhibits static artefacts. Note that both FlowFeat and FeatUp were trained on 224×224 image crops.
- `c_pca_video_models.mp4` compares FlowFeat to VideoMAE [48] based on ViT-B16 and V-JEPA [5] based on ViT-L16, which are pre-trained on videos in a self-supervised manner. We observe that neither of these two models is capable of producing a satisfying level of granularity in the feature maps. Additionally, V-JEPA exhibits peculiar artefacts, which make it unsuitable for downstream dense tasks. In contrast, FlowFeat leverages video datasets more effectively, producing fine-grained and temporally stable feature representations.
- Finally, `d_pca_vs_loftup.mp4` compares FlowFeat to a concurrent work, LoftUp [20], *pre-trained with mask supervision* via the Segment Anything model [25]. LoftUp offers an improved representation over FeatUp and exhibits a high discrimination level of the background regions. However, LoftUp suffers from artefacts, often producing ragged-looking boundaries of moving objects and background elements. Without relying on mask annotation, FlowFeat demonstrates consistent spatial and temporal precision when it comes to dynamic objects. For example, observe sharper boundaries in the “goat” sequence, and a more distinguishable representation of the car and the dancers. These observations explain the quantitative advantage of FlowFeat over LoftUp on the VOS benchmark (*cf.* Tab. 1).

B Further Implementation Details and Analyses

B.1 Video object segmentation

Linear Probing. Linear probing maps the learned representations to object masks. The linear layer has $(d+1) \times (C+1)$ parameters, accounting for the bias term and the background class. The training process of the linear probe is standardised across all evaluated methods. Using the cross-entropy loss, we employ the Adam solver [24] and train the linear projection for 500 iterations with a learning rate of 5×10^{-3} and a weight decay of 5×10^{-4} . We run inference and training of the linear probe by

Table 5: **(a) Effect of reduced feature resolution in linear probing (DAVIS-2017, val).** All features are downsampled to match the baseline encoder resolution. Δ indicates the absolute change in the VOS accuracy w.r.t. the original full-resolution setting of each model. Despite the lower resolution, FlowFeat retains strong VOS performance, confirming that high resolution alone does not explain the observed benefits; FlowFeat embeds a motion-based feature modality that is complementary to the encoder’s representation. **(b) FlowFeat generalizes to larger backbones (DAVIS-2017, val).** As expected, training FlowFeat with ViT-L leads to a consistent improvement of VOS accuracy over the baseline – both for MAE and DINOv2.

(a)								(b)			
Method	Scale	$\mathcal{J}\mathcal{F}_m$	$/\Delta$	\mathcal{J}_m	$/\Delta$	\mathcal{F}_m	$/\Delta$	Method	$\mathcal{J}\mathcal{F}_m$	\mathcal{J}_m	\mathcal{F}_m
DINO2-S14 [37]	–	57.5	–	54.2	–	60.7	–	DINOv2-L14 [37]	59.4	55.8	63.0
+ FeatUp [15]	$\downarrow \times 14$	59.5	–1.0	56.4	–1.0	62.6	–1.0	+ FlowFeat-YT	66.9	63.4	70.4
+ FlowFeat-YT	$\downarrow \times 14$	62.2	–3.6	58.2	–3.8	66.3	–3.4	MAE-L14 [18]	46.7	44.4	49.0
+ FlowFeat-K	$\downarrow \times 14$	62.3	–2.3	58.1	–2.9	66.4	–1.8	+ FlowFeat-YT	55.4	52.0	58.9

fixing the image height at $480p$ and adjusting the width to be divisible by 64, as done by Caron et al. [9]. We apply linear probing on the native feature resolution produced by the model. If necessary, we resize the predicted masks to the original image resolution to compute the metrics w.r.t. the original ground truth masks.

Both FlowFeat and FeatUp [15] utilise a two-stage architecture: a pre-trained encoder that produces low-resolution features, followed by a decoder that generates high-resolution representations. To evaluate the contribution of the high-resolution features, we combine them with the encoder’s low-resolution feature map. In detail, we bilinearly upsample the backbone features and concatenate them with the high-resolution feature tensors. The linear layer projects this joint feature representation to the segmentation logits for each pixel.

Note that linear probing is not auto-regressive – in contrast to local KNN. Specifically, we run inference with the same pre-trained linear projection on all remaining frames in the video. Intuitively, linear probing in the context of VOS is akin to few-shot learning, and provides a more interpretable measure of the spatio-temporal representation quality.

To assess whether the improved VOS performance of FlowFeat is due to the higher feature resolution alone, we downsample the output features of FlowFeat and FeatUp to the same resolution as the baseline encoder (DINO2-S14 [37]). This enables us to directly compare the models under the identical conditions of the feature resolution. Tab. 5a reports the results. While all methods experience some drop in performance – as expected – both FlowFeat variants significantly outperform the baseline and FeatUp. This result demonstrates that FlowFeat yields accuracy gains not merely due to the higher resolution, but also due to the complementary properties derived from motion, which the encoder’s representation lacks.

To evaluate generalization of FlowFeat to the backbone architectures with larger capacity, we consider ViT-L and train FlowFeat by bootstrapping it from DINO2-L14 [37] and MAE-L14 [18]. We evaluate the models using the same linear probing protocol on VOS. As shown in Tab. 5b, FlowFeat leads to substantial performance gains for both models. These results confirm that FlowFeat is effective across different encoder architectures, with varying model capacities and pre-training schemes.

Local KNN. We adopt the label propagation approach from Caron et al. [9]. This protocol requires downsampling the high-resolution feature maps to match the encoder feature resolution. We apply label propagation independently on both the backbone and the downsampled features, and compute the mean of the resulting logits. This ensures hyperparameter consistency of our local KNN evaluation with previous work and across model architectures.

Since downsampling the high-resolution features goes against our motivation, we analyse the impact of the increased feature resolution in Tab. 6. Here, we keep the hyperparameters of the local KNN from the base setting above, but increase the resolution of the feature maps by a factor of 2. As a reference, we provide the VOS accuracy of CRW [21], a CNN-based approach producing the feature maps at the required resolution. Surprisingly, neither the encoder nor FeatUp benefit from the

Table 6: **Scaling up the feature resolution in local KNN probing (DAVIS-2017, val).** We increase the feature resolution by a factor of two and re-run the local KNN unchanged otherwise. The Δ reports the absolute difference in the corresponding metric w.r.t. the base setting of local KNN. CRW [21] is a CNN-based approach provided for a reference. The encoder and FeatUp do not benefit from the higher feature resolution. By contrast, FlowFeat considerably improves its VOS accuracy.

Method	Scale	\mathcal{JF}_m / Δ	\mathcal{J}_m / Δ	\mathcal{F}_m / Δ
CRW-Res18 [21]	$\times 2$	65.2	63.1	67.3
DINO2-S14 [37]	$\times 2$	63.3	-1.8	62.8
+ FeatUp [15]	$\times 2$	64.6	-0.9	64.5
+ FlowFeat-YT	$\times 2$	70.3	+2.7	68.0
+ FlowFeat-K	$\times 2$	70.0	+1.5	67.5

resolution increase. By contrast, FlowFeat improves the results further by significant margins and substantially outperforms the CNN-based reference.

B.2 Semantic segmentation

We keep the DPT decoder in FlowFeat frozen and only train a shallow, one-layer probe. For the low-resolution encoder features and the high-dimensional FeatUp representation we use linear probing. For FlowFeat, we complement the predictions from the encoder with the predictions provided by attention probing. Our implementation of attention probing is inspired by previous work [5]. In our evaluation, we extend this technique to dense prediction tasks. We initialise $C = 27$ (the number of semantic categories) learnable queries in the probe. Each query has the dimensionality of d , matching that of FlowFeat. We use a single block of cross-attention to condition the queries on the FlowFeat representation. Finally, we compute the dot product of the conditioned queries with the spatial feature grid produced by FlowFeat. As a result, we obtain a prediction of size $C \times H \times W$. We also found that downsampling the FlowFeat maps in the cross-attention block significantly improves the probe efficiency without detriment to the probing accuracy.

We train the models with Adam [24] using the cross-entropy loss. We sample mini-batches of size 32, setting the learning rate to 10^{-4} and weight decay to 0.001. All models tend to converge within 100K iterations, which takes less than a day on a single GPU – except for FeatUp, which runs longer, as we discuss in the next section.

For the post-hoc refinement (the “++” variants of FlowFeat), we use a simplified PAMR implementation [1]. Specifically, we use a single kernel of size 11×11 and a fixed scaling factor of 0.1 to produce the local affinity distribution. Crucially, we do not leverage the image intensities to compute the distribution, but replace it with the FlowFeat representation. The refinement runs for 10 steps.

B.3 Monocular depth estimation

The experiments on monocular depth largely follow the setting of the semantic segmentation above. The only conceptual difference is the definition of C , which in monocular depth stands for the number of depth bins. Specifically, to predict the AdaBins [6] representation with $C = 256$ bins (*i.e.* a tensor of size $C \times H \times W$), we initialise $C = 256$ learnable queries in the probe and pass them through a single block of cross-attention. As in semantic segmentation, we compute the dot product of the conditioned queries with the FlowFeat tensor. We train the probes with Adam optimizer [24], batch size 16 and set the learning rate to 10^{-4} and weight decay to 10^{-5} . Following previous work [4, A.3.1], we use a combination of the scale-invariant depth loss [57] and the gradient matching loss [58]. We train all models for up to 100K iterations.

C Efficiency Analysis

We analyse the computational and runtime efficiency of FlowFeat and compare those to the baseline encoder DINO2-S14 [37] and FeatUp [15]. Specifically, we set the input resolution to 224×224 and measure floating-point operations (FLOPs) as well as the FPS rate on an RTX 8000 GPU with

Table 7: **Computational and runtime complexity.** We compare the total and decoder-only floating-point operations (FLOPs), as well as the throughput measured by the frames per second (FPS) rate. Here, we use the DINO2-S14 [37] baseline, the input resolution of 224×224 and RTX 8000 GPU.

Method	Total FLOPs	Decoder FLOPs	FPS
DINO2-S14 [37]	6.14B	–	176.79
+ FeatUp [15]	16.54B	10.33B	25.12
+ FlowFeat	23.43B	17.3B	105.82

48GB of memory. Tab. 7 summarises the benchmark results. In fact, FlowFeat incurs more FLOPs in total than FeatUp. However, all operations in the DPT decoder of FlowFeat are highly parallelisable, hence efficient. As a result, FlowFeat achieves a significantly higher throughput. In the practical terms of FPS, FlowFeat runs four times faster than FeatUp. Indeed, FeatUp’s implementation of the bilateral upsampler, though impressively more efficient than previous work, still falls short of the DPT runtime efficiency.

References

- [57] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.
- [58] Z. Li and N. Snavely. MegaDepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018.