Benchmarking Feature Upsampling Methods for Vision Foundation Models using Interactive Segmentation

Supplementary Material

A. Further Architectural Explorations

A.1. Segmentation Head Ablation Studies

To determine the optimal segmentation head, three configurations were evaluated, aiming to maintain simplicity while ensuring sufficient expressiveness.

- *Linear Head* – A single 1×1 convolutional layer.
- *Simple Conv Head* – Three 1×1 convolutional layers with an inner channel dimension of 384.
- *Conv Head* – Similar to the *Simple Conv Head*, but with the first two layers using a kernel size of 3×3 instead of 1×1.

Tab. 2 presents a comparison of these architectures using ViT [1] backbone without feature upsampling, a symmetric patch embedding click encoder, and early click injection. The *Conv Head* achieves the best balance between performance and complexity and is therefore selected for all subsequent experiments.

Head	GrabCut			
	NoC80	NoC85	NoC90	IoU@1
Linear	6.06	9.02	14.06	36.33
Simple Conv	4.76	7.94	13.20	48.56
Conv	4.40	6.14	9.80	56.03

Table 2. **Comparison of segmentation heads.** Models were trained on the SBD dataset for 20 epochs with a batch size of 16, using ViT [1] backbone, a symmetric patch embedding click encoder with early injection, and no upsampler.

A.2. Alternative Design Explorations

In this work, we explored two alternative architectures for benchmarking VFMs that did not yield the desired performance. Below, we describe these approaches in more detail and discuss potential reasons for their limitations, which may inform future improvements.

Multiscale Architecture. SOTA IS models [7, 11] typically leverage multiscale features, which are extracted from vision backbone outputs via FPN and subsequently processed by a multiscale segmentation head. While our primary architecture focused on removing all multiscale components, here we investigate whether upsamplers can be beneficially used in FPNs.

The architecture of our *Upsampler-based FPN* follows a design similar to the FPN in ViTDet [6], which constructs multiscale feature maps exclusively from the final feature map of the vision backbone. Specifically, the DINOv2 (S/14) [8] backbone produces a final feature map at a $\frac{1}{14}$ resolution, from which we generate feature maps at resolutions $1, \frac{1}{4}, \frac{1}{14}, \frac{1}{28}$. The highest resolutions (1 and $\frac{1}{4}$) are obtained via upsampling, followed by a single convolutional layer. The $\frac{1}{14}$ resolution is derived directly through a convolutional layer, while the $\frac{1}{28}$ resolution is obtained by applying a 2×2 max pooling operation prior to convolution. All convolutional layers employ 1×1 kernels, followed by normalization layers, primarily to adjust the channel dimensions of each feature map. The final multiscale feature maps have channel dimensions of $\{C, 2C, 4C, 8C\}$ for resolutions $\{1, \frac{1}{4}, \frac{1}{14}, \frac{1}{28}\}$, respectively, where $C = 128$.

As a baseline, we also adapted the FPN from SimpleClick [7], which maps the feature map at $\frac{1}{14}$ resolution to scales $\{\frac{2}{7}, \frac{1}{7}, \frac{1}{14}, \frac{1}{28}\}$. The resolutions $\frac{2}{7}$ and $\frac{1}{7}$ are generated using two and one transposed convolutional layers, respectively. The smallest resolution, $\frac{1}{28}$, is obtained by applying a convolutional layer with a 2×2 kernel and a stride of 2. The feature map at the original resolution remains unchanged. At the final stage, all scales are processed by convolutional layers with a 1×1 kernel, followed by normalization layers. The output channel dimensions are consistent with those used in the Upsampler-based FPN.

FPN	GrabCut				Berkeley				DAVIS			
	NoC80	NoC85	NoC90	IoU@1	NoC80	NoC85	NoC90	IoU@1	NoC80	NoC85	NoC90	IoU@1
SimpleClick FPN	**3.88**	5.52	**9.74**	**67.00**	**5.18**	**8.58**	**13.61**	**60.83**	10.81	14.25	17.51	54.50
Upsampler-based (Bilinear)	4.84	7.02	11.16	61.42	6.29	10.11	15.81	55.60	11.80	15.08	17.88	53.63
Upsampler-based (LoftUp*)	4.14	**5.44**	10.16	63.01	5.26	8.85	13.64	59.39	**10.40**	**13.86**	**17.20**	**60.49**

Table 1. **Evaluation of the multiscale IS architecture.** The backbone used is DINOv2 (S/14) [8], with a symmetric patch embedding click encoder and early injection. The segmentation head is an adapted version of SegFormer's head [10]. * Indicates results obtained from non-final checkpoints.

For the multiscale segmentation head, we adopt the architecture from Segformer [10]. Feature maps at four different scales are processed through independent convolutional layers with a 1×1 kernel and an output channel dimension of 256. The features are then bilinearly interpolated to match the resolution of the largest feature map (either 1 or $\frac{2}{7}$ in our setup), concatenated along the channel dimension, and passed through an additional convolutional layer with the same output channels and a 1×1 kernel. Finally, a classification layer is applied. Normalization layers are included as intermediate steps in the process.

We conducted experiments using DINOv2 (S/14) [8] with a symmetric patch embedding click encoder and early injection. Models were trained on the SBD dataset [3] for 20 epochs with the batch size of 8. Similar to other experiments, VFM and upsampler modules are kept frozen. The results, shown in Tab. 1, indicate that the Upsampler-based FPN, when combined with LoftUp [5], outperforms the one with bilinear interpolation. However, its advantage over the original FPN baseline remains inconclusive. Furthermore, our primary single-scale architecture consistently outperforms the multiscale approach. We hypothesize that this performance gap stems from the fixed, predefined channel dimensions used in the multiscale setup to maintain computational efficiency, which may cause information loss in the upsampled features. One potential solution would be to retain all channels from the upsampled features, which we leave for future exploration. Another likely reason for the poor performance is that the multiscale structure becomes redundant, as the upsampler module already performs scaling effectively.

Multi-granular Architecture. A common challenge in IS is the ambiguity in determining the desired object scale based on a given input click, as the click may correspond to objects of varying granularities. Recently, GraCo [11] addressed this issue by introducing a granularity scale as an additional input parameter. The granularity scale, a continuous value between 0 and 1, specifies the intended object scale. To incorporate this concept, the authors extended the pre-trained SimpleClick model [7] by injecting learnable granularity embeddings into the segmentation pipeline. To enhance granularity control learning, LoRA fine-tuning [4] was applied.

To construct our multi-granular IS architecture, we closely follow the methodology introduced in GraCo [11]. Our approach builds upon our primary single-scale pipeline, which comprises VFM, click encoder, upsampler, and segmentation head. The click encoder and segmentation head are initialized using the results from previous experiments and, along with VFM and upsampler, remain frozen during training. Both the fixed granularity embeddings and the LoRA parameters are introduced as learnable components. The granularity embeddings are incorporated into the network by adding them element-wise to both image and click features, following either an early or late injection strategy. Simultaneously, LoRA fine-tuning is applied to **Q** and **K** projection layers in each attention block of the VFM, enabling efficient adaptation to multi-granular setup.

Tab. 3 presents the results for DINOv2 (S/14) using a symmetric patch embedding click encoder with early injection. The models were trained using the extended, multi-granular version of SBD dataset [3] for 20 epochs with the batch size of 16. The dataset generation, training, and evaluation protocols strictly follow those established by GraCo.

Upsampler	GrabCut			
	NoC80	NoC85	NoC90	IoU@1
Low-res	5.28	6.92	10.80	59.39
FeatUp [2]	**3.68**	**5.02**	**8.02**	**69.61**
Low-res + GRA	12.20	15.38	18.88	27.64
FeatUp [2] + GRA	9.78	11.28	13.92	12.30

Table 3. **Evaluation of the multi-granular IS architecture.** The backbone used is DINOv2 (S/14) [8], with a symmetric patch embedding click encoder and early injection.

Here, we observe that the multi-granular setup performs considerably worse than the primary single-scale pipeline. We attribute the reduced performance to potential shifts in the backbone feature distribution caused by LoRA fine-tuning, which may alter the upsampler's input features. Addressing this limitation would require joint fine-tuning of the backbone and the upsampler, a strategy not explored in this work.

B. Additional Visualizations

We provide further visualizations of features after upsampling in Fig. 1, as well as segmentation masks after the first and third clicks in Figs. 2 and 3, respectively. All visualizations are generated on GrabCut dataset [9] using the DINOv2 (S/14) [8] backbone and a symmetric patch embedding click encoder with early injection.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*. OpenReview.net, 2021. 1

[2] 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 *ICLR*. OpenReview.net, 2024. 2

[3] Bharath Hariharan, Pablo Arbeláez, Lubomir D. Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, pages 991–998. IEEE Computer Society, 2011. 2

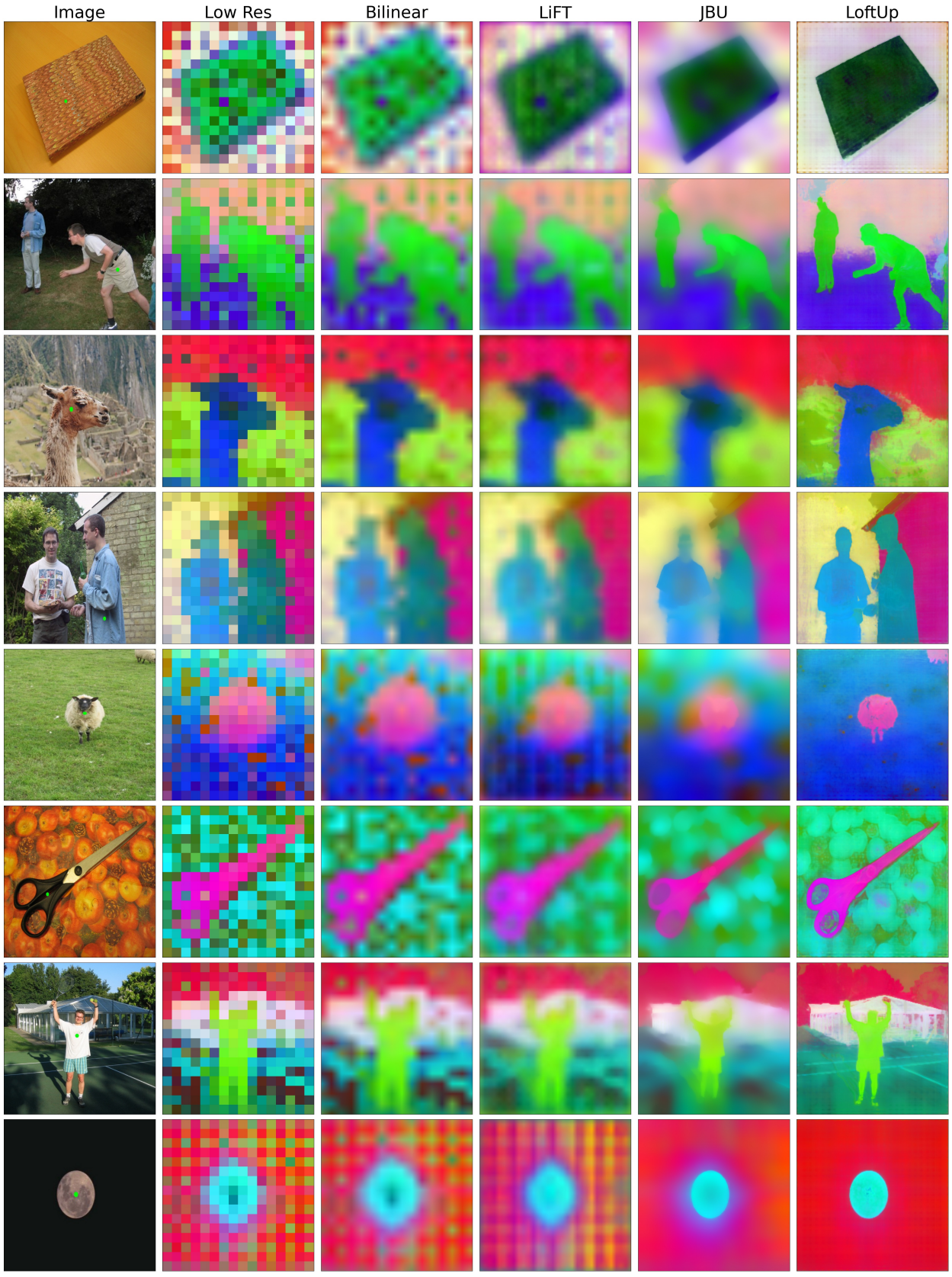

Figure 1. **Additional visualizations of upsampler features with a single input click.** The click is indicated by a green dot on the original images.

Figure 2. **Additional segmentation results with a single input click.** The click is indicated by a green dot.

[4] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*. OpenReview.net, 2022. 2

[5] Haiwen Huang, Anpei Chen, Volodymyr Havrylov, Andreas Geiger, and Dan Zhang. Loftup: Learning a coordinate-based feature upsampler for vision foundation models, 2025. 2

[6] Yanghao Li, Hanzi Mao, Ross B. Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *ECCV (9)*, pages 280–296. Springer, 2022. 1

[7] Qin Liu, Zhenlin Xu, Gedas Bertasius, and Marc Niethammer. Simpleclick: Interactive image segmentation with simple vision transformers. In *ICCV*, pages 22233–22243. IEEE, 2023. 1, 2

[8] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024, 2024. 1, 2

[9] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004. 2

[10] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, José M. Álvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, pages 12077–12090, 2021. 1, 2

[11] Yian Zhao, Kehan Li, Zesen Cheng, Pengchong Qiao, Xiawu Zheng, Rongrong Ji, Chang Liu, Li Yuan, and Jie Chen. Graco: Granularity-controllable interactive segmentation. In *CVPR*, pages 3501–3510. IEEE, 2024. 1, 2

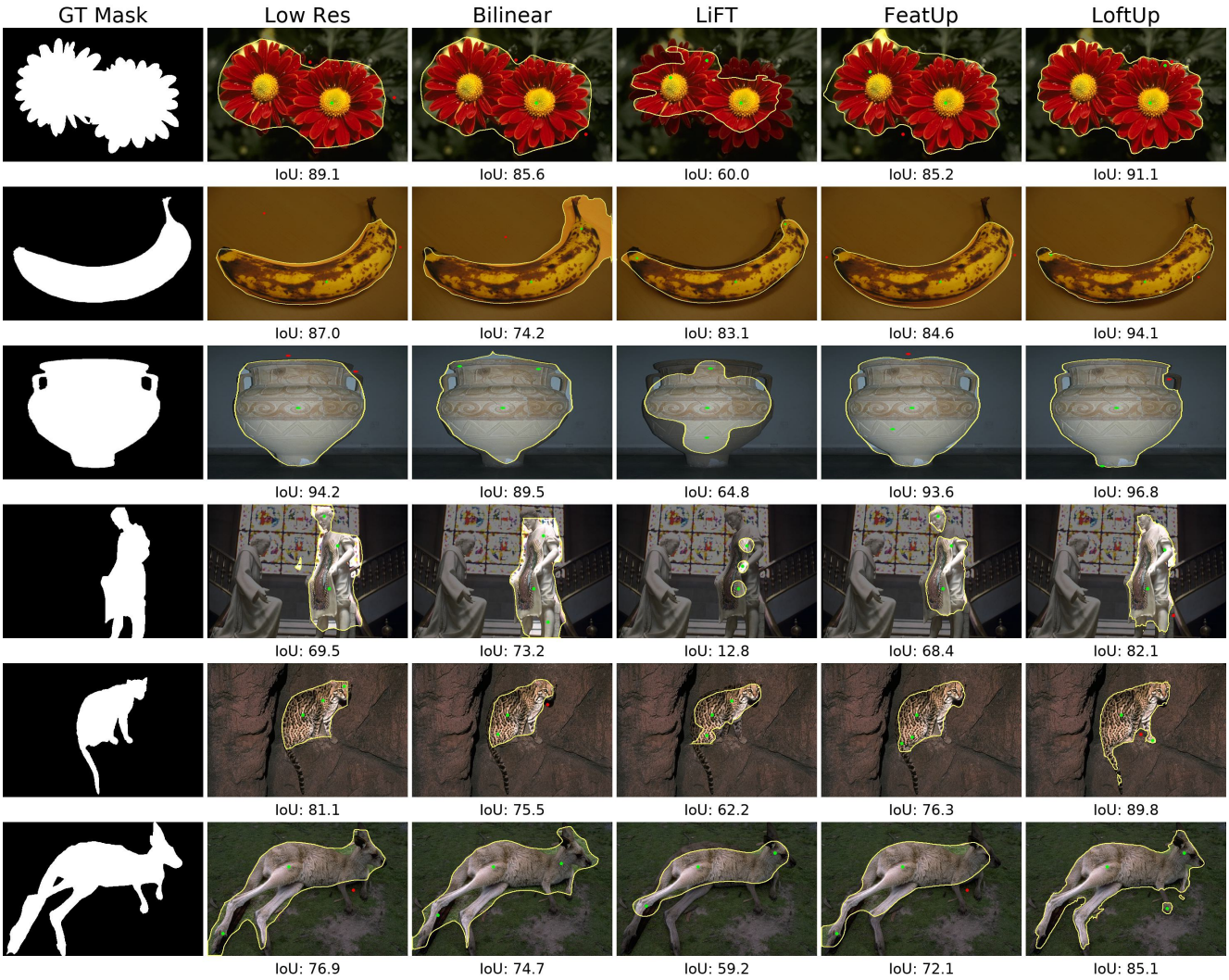

Figure 3. **Additional segmentation results with three input clicks.** Positive and negative clicks are indicated by green and red dots, respectively.