International Journal of Computer Vision
https://doi.org/10.1007/s11263-024-02191-8

FADE: A Task-Agnostic Upsampling Operator for Encoder–Decoder Architectures

Hao Lu[1] · **Wenze Liu**[1] · **Hongtao Fu**[1] · **Zhiguo Cao**[1]

Received: 26 June 2023 / Accepted: 14 July 2024

Abstract
The goal of this work is to develop a task-agnostic feature upsampling operator for dense prediction where the operator is required to facilitate not only region-sensitive tasks like semantic segmentation but also detail-sensitive tasks such as image matting. Prior upsampling operators often can work well in either type of the tasks, but not both. We argue that task-agnostic upsampling should dynamically trade off between semantic preservation and detail delineation, instead of having a bias between the two properties. In this paper, we present FADE, a novel, plug-and-play, lightweight, and task-agnostic upsampling operator by fusing the assets of decoder and encoder features at three levels: (i) considering both the encoder and decoder feature in upsampling kernel generation; (ii) controlling the per-point contribution of the encoder/decoder feature in upsampling kernels with an efficient semi-shift convolutional operator; and (iii) enabling the selective pass of encoder features with a decoder-dependent gating mechanism for compensating details. To improve the practicality of FADE, we additionally study parameter- and memory-efficient implementations of semi-shift convolution. We analyze the upsampling behavior of FADE on toy data and show through large-scale experiments that FADE is task-agnostic with consistent performance improvement on a number of dense prediction tasks with little extra cost. For the first time, we demonstrate robust feature upsampling on both region- and detail-sensitive tasks successfully. Code is made available at: https://github.com/poppinace/fade

Keywords Feature upsampling · Dense prediction · Semantic segmentation · Image matting · Object detection · Instance segmentation · Depth estimation

1 Introduction

Feature quality, being an important yet hard-to-quantify indicator, significantly influences the performance of a vision system (Girshick et al., 2014). This is particularly true for dense prediction tasks such as semantic segmentation (Long et al., 2015) and object detection (Ren et al., 2015), where the predictions highly correlate with the responses of feature maps (Zhou et al., 2016). Prior art has proposed various ways to enhance the feature quality by operating features, including, but not limited to, spatial pooling (Chen et al., 2018; Zhao et al., 2017), feature pyramid fusion (Lin et al., 2017b; Liu et al., 2018), attention manipulation (Wang et al., 2018), context aggregation (Yuan et al., 2021), and feature alignment (Li et al., 2020b; Huang et al., 2021). Yet, the most famous segmentation model (Kirillov et al., 2023) so far still struggles to generate accurate boundary predictions, which suggests feature quality remains unsatisfactory. In this work, we delve into an easily overlooked yet fundamental component that closely relates to feature quality—feature upsampling.

Feature upsampling, which aims to recover the spatial resolution of features, is an indispensable stage in most dense prediction models (Ronneberger et al., 2015; Badrinarayanan et al., 2017; Xiao et al., 2018; Wang et al., 2020;

Communicated by Wanli Ouyang.

✉ Zhiguo Cao
zgcao@hust.edu.cn

Hao Lu
hlu@hust.edu.cn

Wenze Liu
wzliu@hust.edu.cn

Hongtao Fu
htfu@hust.edu.cn

[1] The Key Laboratory of Image Processing and Intelligent Control, Ministry of Education, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China

Published online: 22 July 2024

Zheng et al., 2021; Xie et al., 2021) as almost all dense prediction tasks prefer high-res predictions. Since feature upsampling is often close to the prediction head, the quality of upsampled features can provide a direct implication of the prediction quality. A good upsampling operator would therefore contribute to improved feature quality and prediction. Yet, conventional upsampling operators, such as nearest neighbor (NN) or bilinear interpolation (Lin et al., 2017a), deconvolution (Zeiler and Fergus, 2014), max unpooling (Badrinarayanan et al., 2017), and pixel shuffle (Shi et al., 2016), often have a preference of a specific task. For instance, bilinear interpolation is favored in semantic segmentation (Chen et al., 2018; Xie et al., 2021), while pixel shuffle is preferred in image super-resolution (Ignatov et al., 2021).

A main reason is that each dense prediction task has its own focus: some tasks like semantic segmentation (Long et al., 2015) and instance segmentation (He et al., 2017) are region-sensitive, while some tasks such as image super-resolution (Dong et al., 2015) and image matting (Xu et al., 2017; Lu et al., 2019) are detail-sensitive. If one expects an upsampling operator to generate semantically consistent features such that a region can share the same class label, it is often difficult for the same operator to recover boundary details simultaneously, and vice versa. Indeed empirical evidence shows that bilinear interpolation and max unpooling have inverse behaviors in segmentation and matting (Lu et al., 2019, 2022a), respectively.

In an effort to evade 'trials-and-errors' from choosing an upsampling operator for a certain task at hand, there has been a growing interest in developing a generic upsampling operator for dense prediction (Mazzini, 2018; Tian et al., 2019; Wang et al., 2019, 2021; Lu et al., 2019, 2022a; Dai et al., 2021). For example, CARAFE (Wang et al., 2019) shows its benefits on four dense prediction tasks, including object detection, instance segmentation, semantic segmentation, and image inpainting. IndexNet (Lu et al., 2019) also boosts performance on several tasks such as image matting, image denoising, depth prediction, and image reconstruction. However, a comparison between CARAFE and IndexNet (Lu et al., 2022a) indicates that neither CARAFE nor IndexNet can defeat its opponent on both region- and detail-sensitive tasks (CARAFE outperforms IndexNet on segmentation, while IndexNet can surpass CARAFE on matting), which can also be observed from the inferred segmentation masks and alpha mattes in Fig. 1. This raises a fundamental research question: *What makes for task-agnostic upsampling?*

After an apples-to-apples comparison between existing dynamic upsampling operators (Fig. 2), we hypothesize that it is the inappropriate and/or insufficient use of high-res encoder and low-res decoder features that leads to the task dependency of upsampling. We also believe that there should exist a unified form of upsampling operator that is truly task-agnostic. In particular, we argue that *a task-agnostic upsampling operator should dynamically trade off between semantic preservation and detail delineation in a content-aware manner, instead of having a bias between the two properties*. To this end, our main idea is to make the full use of encoder and decoder features in upsampling (kernels). We therefore introduce FADE, a novel, plug-and-play, lightweight, and task-agnostic upsampling operator for encoder-decoder architectures. The name also implies its working mechanism: upsampling features in a 'fade-in' manner, from recovering spatial structure to delineating subtle details. In the context of hierarchical encoder-decoder architectures such as feature pyramid networks (FPNs) (Lin et al., 2017b) and U-Net (Ronneberger et al., 2015), semantic information is rich in low-res decoder features, and detailed information is often abundant in high-res encoder features. To exploit both information in feature upsampling, FADE Fuses the Assets of Decoder and Encoder with three key observations and designs:

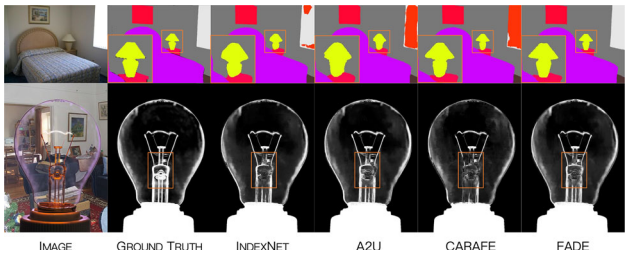

Fig. 1 Inferred segmentation masks and alpha mattes with different upsampling operators. The compared operators include IndexNet (Lu et al., 2019), A2U (Dai et al., 2021), CARAFE (Wang et al., 2019), and our proposed FADE. Among competitors, only FADE generates both the high-quality mask and the alpha matte

(i) By exploring why CARAFE works well on region-sensitive tasks but poorly on detail-sensitive tasks, and why IndexNet and A2U (Dai et al., 2021) behave conversely, we observe that what features (encoder or decoder) to use to generate the upsampling kernels matters. Using low-res decoder features preserves regional coherence, while using high-res encoder features helps recover details. It is thus natural to seek whether combining encoder and decoder features enjoys both merits, which underpins the core idea of FADE, as shown in Fig. 2.

(ii) To integrate high-res encoder and low-res decoder features, a subsequent obstacle is how to deal with the problem of resolution mismatch. A standard way is to implement U-Net-style fusion (Ronneberger et al., 2015), including feature interpolation, feature concatenation, and convolution. However, we show that this naive implementation can introduce artifacts into upsampling kernels. To solve this, we introduce a semi-shift convolutional operator that unifies channel compres-

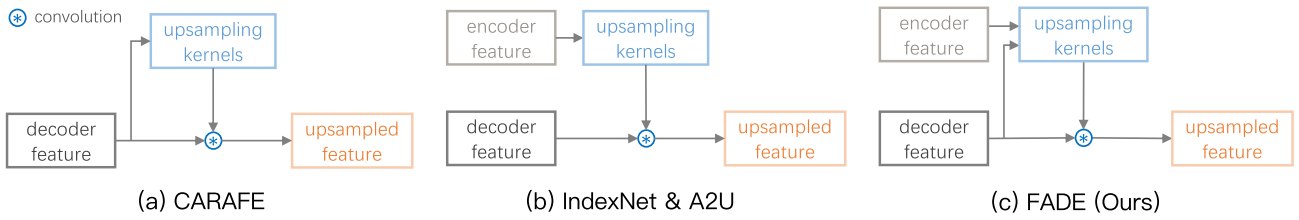

Fig. 2 Main difference between dynamic upsampling operators on the use of encoder and/or decoder features. **a** CARAFE (Wang et al., 2019) generates upsampling kernels conditioned on decoder features, while **b** IndexNet (Lu et al., 2022a) and A2U (Dai et al., 2021) generate kernels using encoder features only. By contrast, **c** FADE considers both encoder and decoder features in upsampling kernel generation

sion, concatenation, and kernel generation. Particularly, it allows granular control over how each feature point contributes to upsampling kernels.

(iii) Inspired by the gating mechanism used in FPN-like designs (Li et al., 2020c, 2023), we further refine upsampled features by enabling selective pass of high-res encoder features via a simple decoder-dependent gating unit.

To improve the practicality and efficiency of FADE, we also investigate parameter-efficient and memory-efficient implementations of semi-shift convolution. Such implementations lead to a lightweight variant of FADE termed FADE-Lite. We show that, even with one forth number of parameters of FADE, FADE-Lite still preserves the task-agnostic property and behaves reasonably well across different tasks. The memory-efficient implementation also enables direct execution of cross-resolution convolution, without explicit feature interpolation for resolution matching.

We conduct experiments on seven data sets covering six dense prediction tasks. We first validate our motivation and the rationale of our design via several toy-level and small-scale experiments, such as binary image segmentation on Weizmann Horse (Borenstein & Ullman, 2002), image reconstruction on Fashion-MNIST (Xiao et al., 2017), and semantic segmentation on SUN RGBD (Song et al., 2015). We then show through large-scale evaluations that FADE reveals its task-agnostic property by consistently boosting both region- and detail-sensitive tasks, for instance: (i) *semantic segmentation*: FADE improves SegFormer-B1 (Xie et al., 2021) by + 2.73 mask IoU and + 4.85 boundary IoU on ADE20K (Zhou et al., 2017) and steadily boosts the boundary IoU with stronger backbones, (ii) *image matting*: FADE outperforms the previous best matting-specific upsampling operator A2U (Dai et al., 2021) on Adobe Composition-1K (Xu et al., 2017), (iii) *object detection* and (iv) *instance segmentation*: FADE performs comparably against the best performing operator CARAFE over Faster R-CNN (Ren et al., 2015) (+ 1.1 AP for FADE vs. + 1.2 AP for CARAFE with ResNet-50) and Mask R-CNN (He et al., 2017) (+ 0.4 mask AP for FADE vs. + 0.7 mask AP for CARAFE with ResNet-50) baselines on Microsoft COCO (Lin et al., 2014), and (v) *monocular depth estimation*: FADE also surpasses the previous best upsampling operator IndexNet (Lu et al., 2022a) over the BTS (Lee et al., 2019) baseline on NYU Depth V2 (Silberman et al., 2012). In addition, FADE retains the lightweight property by introducing only a few amount of parameters and FLOPs. It has also good generality across convolutional and transformer architectures (Xiao et al., 2018; Xie et al., 2021).

Overall, our contributions include the following:

- For the first time, we show that task-agnostic upsampling is made possible on both high-level region-sensitive and low-level detail-sensitive tasks;
- We present FADE, one of the first task-agnostic upsampling operator, that fuses encoder and decoder features in generating upsampling kernels, uses an efficient semi-shift convolutional operator to control per-point contribution, and optionally applies a gating mechanism to compensate details;
- We provide a comprehensive benchmarking on state-of-the-art upsampling operators across five mainstream dense prediction tasks, which facilitates future study.

A preliminary conference version of this work appeared in (Lu et al., 2022b). We extend (Lu et al., 2022b) from the following aspects: (i) to highlight the task-agnostic property, we validate FADE comprehensively on more baseline models, e.g., UPerNet (Xiao et al., 2018), Faster RCNN (Ren et al., 2015), Mask RCNN (He et al., 2017), and BTS (Lee et al., 2019), on different network scales, from SegFormer-B1 to -B5 (Xie et al., 2021) and from R50 to R101 (He et al., 2016), and on three additional vision tasks including object detection, instance segmentation, and monocular depth estimation; (ii) we carefully benchmark the performance of state-of-the-art dynamic upsampling operators on the evaluated tasks to provide a basis for future studies; (iii) we further explore parameter-efficient and memory-efficient implementations of semi-shift convolution to enhance the practicality of FADE, which also leads to a lightweight variant called FADE-Lite; (iv) by observing some unexpected phenomena

in experiments, we rethink the value of the gating mechanism in FADE and provide additional analyses and insights on when to use the gating unit, particularly for instance-level tasks; (v) we extend the related work by comparing feature upsampling with other closely related techniques such as feature alignment and boundary processing; (vi) we also extend our discussion on the general value of feature upsampling to dense prediction.

2 Literature Review

We review upsampling operators in deep networks, techniques that share a similar spirit to upsampling including feature alignment and boundary processing, and typical dense prediction tasks in vision.

2.1 Feature Upsampling

Unlike joint image upsampling (Tomasi & Manduchi, 1998; He et al., 2010), feature upsampling operators are mostly developed in the era of deep learning, to respond to the need for recovering spatial resolution of encoder features (decoding). Conventional upsampling operators typically use fixed/hand-crafted kernels. For instance, the kernels in the widely used NN and bilinear interpolation are defined by the relative distance between pixels. Deconvolution (Zeiler and Fergus, 2014), *a.k.a.* transposed convolution, also applies a fixed kernel during inference, despite the kernel parameters are learned. Pixel Shuffle (Shi et al., 2016) first employs convolution to adjust feature channels and then reduces the depth dimension to increase the spatial dimension. While the main purpose of resolution increase is achieved, the operators above also introduce certain artifacts into features. For instance, it is well-known that, interpolation smooths boundaries, and deconvolution generates checkerboard artifacts (Odena et al., 2016). Several recent work has shown that unlearned upsampling has become a bottleneck behind architectural design (Liu et al., 2023), and dynamic upsampling behaviors are more expected (Lu et al., 2019). Among hand-crafted operators, unpooling (Badrinarayanan et al., 2017) perhaps is the only operator that implements dynamic upsampling, i.e., each upsampled position is data-dependent conditioned on the max operator. The importance of such a dynamic property has been exemplified by some recent dynamic kernel-based upsampling operators (Wang et al., 2019; Lu et al., 2019; Dai et al., 2021; Lu et al., 2022c), which leads to a new direction from considering generic feature upsampling across tasks and architectures. In particular, CARAFE (Wang et al., 2019) implements context-aware reassembly of features with decoder-dependent upsampling kernels, IndexNet (Lu et al., 2019) provides an indexing perspective of upsampling and executes upsampling by learning a soft index (kernel) function, and A2U (Dai et al., 2021) introduces affinity-aware upsampling kernels by exploiting second-order information. At the core of these operators is the data-dependent upsampling kernels whose kernel parameters are not learned but dynamically predicted by a sub-network.

However, while being dynamic, CARAFE, A2U, and IndexNet still exhibit a certain degree of bias on specific tasks. In this work, we show through FADE that the devil is in the use of encoder and decoder features in generating upsampling kernels.

2.2 Feature Alignment and Boundary Processing

Different from dynamic upsampling that aims to enhance feature quality *during* resolution change, much existing work also attempts to enhance the feature quality *after* matching resolution. Two closely related techniques are feature alignment and boundary processing. Feature alignment explores to align multi-level feature maps by warping features with, for example, either sampling offsets (Wu et al., 2022; Huang et al., 2021) or a dense flow field (Li et al., 2020b, 2023), which has been found effective in reducing semantic aliasing during cross-resolution feature fusion. Another idea is to use a gating unit to align and refine features (Li et al., 2020c), which prevents encoder noise from entering decoder feature maps. FADE has also a similar design as post-processing, but is much simpler. Considering that, most fragile predictions in segmentation are along object boundaries, boundary processing techniques are developed to optimize boundary quality. In particular, PointRend (Kirillov et al., 2020) views segmentation as a rendering problem and adaptively selects points to predict crisp boundaries by an iterative subdivision algorithm. Li et al. (2020a) improves boundary prediction with decoupled body and edge supervision. Boundary-preserving Mask R-CNN (Cheng et al., 2020) presents a boundary-preserving mask head to improve mask localization accuracy. Gated-SCNN (Takikawa et al., 2019) introduces a two-stream architecture that wires shape information as a separate processing branch to process boundary-related information specifically.

Compared with dynamic upsampling, feature alignment and boundary processing are typically executed *after* naive feature upsampling. Since feature upsampling is inevitable, it would be interesting to see whether one could enhance the feature quality *during* upsampling, which is exactly one of the goals of dynamic upsampling. In this work, we show that FADE is capable of mitigating semantic aliasing as feature alignment and of improving boundary predictions as boundary processing. FADE also demonstrates universality across a number of tasks more than segmentation.

2.3 Dense Prediction

Dense prediction covers a broad class of per-pixel labeling tasks, ranging from mainstream object detection (Ren et al., 2015), semantic segmentation (Long et al., 2015), instance segmentation (He et al., 2017), and depth estimation (Eigen et al., 2014) to low-level image restoration (Mao et al., 2016), image matting (Xu et al., 2017), edge detection (Xie and Tu, 2015), and optical flow estimation (Teed & Deng, 2020), to name a few. An interesting property about dense prediction is that a task could be region-sensitive or detail-sensitive. The sensitivity is closely related to what metric is used to assess the task. In this sense, semantic/instance segmentation is region-sensitive, because the standard Mask Intersection-over-Union (IoU) metric (Everingham et al., 2010) is mostly affected by regional mask prediction quality, instead of boundary quality. On the contrary, image matting can be considered detail-sensitive, because the error metrics (Rhemann et al., 2009) are mainly computed from trimap regions that are full of subtle details or transparency. Note that, when we emphasize region sensitivity, we do not mean that details are not important, and vice versa. In fact, the emergence of the Boundary IoU metric (Cheng et al., 2021) implies that the limitation of a certain evaluation metric has been noticed by our community.

Feature upsampling can play important roles in dense prediction, not only for generating high-resolution predictions but also for improving the quality of predictions. The goal of developing a task-agnostic and content-aware upsampling operator capable of both regional preservation and detail delineation can have a broad impact on a number of dense prediction tasks. In this work, we evaluate FADE and other upsampling operators on both types of tasks using both region-aware and detail-aware metrics.

3 Task-Agnostic Upsampling: A Trade-off Between Semantic Preservation and Detail Delineation?

Before we present FADE, we share some of our view points towards task-agnostic upsampling, which may be helpful to understand our designs in FADE.

Remark 1 Encoder and decoder features play different roles in upsampling, particularly in the generation of upsampling kernels.

In dense prediction models, downsampling stages are involved to reduce computational burden or to acquire a large receptive field, bringing the need of peer-to-peer upsampling stages to recover the spatial resolution, which together constitutes the basic encoder-decoder architecture. During downsampling, details of high-res features are impaired or even lost, but the resulting low-res encoder features often have good semantic meanings that can pass to decoder features. Hence, we believe an ideal upsampling operator should appropriately resolve two issues: (1) preserve the semantic information already extracted; (2) compensate as many lost details as possible without deteriorating the semantic information. NN or bilinear interpolation only meets the former. This conforms to our intuition that interpolation often smooths features. A reason is that low-res decoder features have no prior knowledge about missing details. Other operators that directly upsample decoder features, such as deconvolution and pixel shuffle, can have the same problem with poor detail compensation. Compensating details requires high-res encoder features. This is why unpooling that stores indices before downsampling has good boundary delineation (Lu et al., 2019), but it hurts the semantic information due to zero-filling.

Dynamic upsampling operators, including CARAFE (Wang et al., 2019), IndexNet (Lu et al., 2019), and A2U (Dai et al., 2021), alleviate the problems above with data-dependent upsampling kernels. Their upsampling modes are shown in Fig. 2a, b. From Fig. 2, it can be observed that, CARAFE generates upsampling kernels conditioned on decoder features, while IndexNet (Lu et al., 2019) and A2U (Dai et al., 2021) generate kernels via encoder features. This may explain the inverse behavior between CARAFE and IndexNet/A2U on region- or detail-sensitive tasks (Lu et al., 2022a). Yet, we find that generating upsampling kernels using either encoder or decoder features can lead to suboptimal results, and it is critical *to leverage both encoder and decoder features for task-agnostic upsampling*, as implemented in FADE (Fig. 2c).

Remark 2 How each feature point contributes to upsampling matters.

After deciding what the features to use, the follow-up question is how to use the features effectively and efficiently. The main obstacle is the mismatched resolution between encoder and decoder features. Per Fig. 3, one may consider simple interpolation for resolution matching, but this can lead to sub-optimal upsampling. Considering the case of applying $\times 2$ NN interpolation to decoder features, if we use 3×3 convolution to generate the upsampling kernel, the effective receptive field of the kernel can reduce to be $< 50\%$: before interpolation there are 9 valid points in a 3×3 window, but only 4 valid points are left after interpolation. Besides this, another more important issue remains. Still in the $\times 2$ upsampling in Fig. 3, the four windows which control the variance of upsampling kernels w.r.t. the 2×2 neighbors of high resolution are affected by the naive interpolation. Controlling a high-res upsampling kernel map, however, is blind with the low-res decoder feature. It contributes little to the variance of the four neighbors. A more reasonable choice may be to *let*

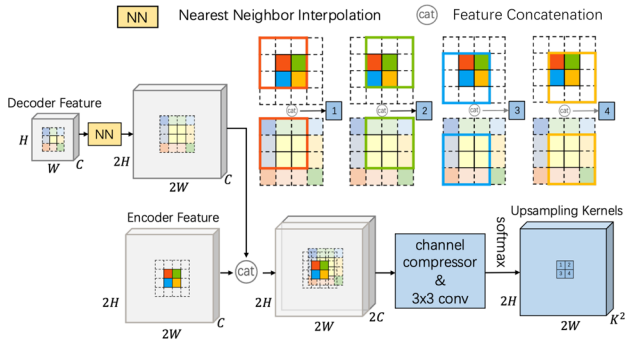

Fig. 3 Naive implementation for generating upsampling kernels using encoder and decoder features. The kernel prediction using high-res encoder and low-res decoder features requires matching resolution with explicit feature interpolation and concatenation, followed by channel compression and convolution

encoder and decoder features cooperate to control the overall upsampling kernel, but let the encoder feature alone control the variance of the four neighbors. This insight exactly motivates the design of semi-shift convolution (Sect. 4.3).

Remark 3 High-res encoder features can be leveraged for further detail refinement.

Besides helping structural recovery via upsampling kernels, there remains useful information in encoder features. Since encoder features only go through a few layers of a network, they preserve 'fine details' of high resolution. In fact, nearly all dense prediction tasks require fine details, e.g., despite regional prediction dominates in instance segmentation, accurate boundary prediction can significantly boost performance (Tang et al., 2021), not to mention the stronger request of fine details in detail-sensitive tasks. *The demands of fine details in dense prediction need further exploitation of encoder features*. Following existing ideas (Cho et al., 2014; Li et al., 2020c, 2023), we explore the use of a gating mechanism by leveraging low-res decoder features to guide where the high-res encoder features can pass through. Yet, in some instance-aware tasks, we find that the gate is better left fully open (more discussion can be found in Sect. 4.4).

4 FADE: Fusing the Assets of Decoder and Encoder

Here we elaborate our designs in FADE. We first revisit the framework of dynamic upsampling, then present from three aspects on how to fuse the assets of decoder and encoder features in upsampling, particularly discussing the principle and the efficient implementations of the semi-shift convolution.

4.1 Dynamic Upsampling Revisited

Here we review some basic operations in recent dynamic upsampling operators such as CARAFE (Wang et al., 2019), IndexNet (Lu et al., 2019), and A2U (Dai et al., 2021). Figure 2 briefly summarizes their upsampling modes. They share an identical pipeline, i.e., first generating data-dependent upsampling kernels, and then reassembling the decoder features using the kernels. Typical dynamic upsampling kernels are content-aware, but channel-shared, which means each position has a unique upsampling kernel in the spatial dimension, but the same ones are shared in the channel dimension.

CARAFE learns upsampling kernels directly from decoder features and then reassembles them to high resolution. Specifically, the decoder features pass through two consecutive convolutional layers to generate the upsampling kernels, of which the former is a channel compressor implemented by 1×1 convolution used to reduce the computational complexity and the latter is a content encoder with 3×3 convolution. IndexNet and A2U, however, adopt more sophisticated modules to leverage the merit of encoder features. Further details can be referred to (Wang et al., 2019; Lu et al., 2019; Dai et al., 2021).

FADE is designed to maintain the simplicity of dynamic upsampling. Hence, we mainly optimize the process of kernel generation with semi-shift convolution, and the channel compressor will also function as a way of pre-fusing encoder and decoder features. In addition, FADE also includes a gating mechanism for detail refinement. The overall pipeline of FADE is summarized in Fig. 4. In what follows, we explain our three key designs and present our efficient implementations.

4.2 Generating Upsampling Kernels from Encoder and Decoder Features

We first showcase a few visualizations on some small-scale or toy-level data sets to highlight the importance of both encoder and decoder features for task-agnostic upsampling. We choose semantic segmentation on SUN RGBD (Song et al., 2015) as the region-sensitive task and image reconstruction on Fashion MNIST (Xiao et al., 2017) as the detail-sensitive one. We follow the network architectures and the experimental settings in (Lu et al., 2022a). Since we focus on upsampling, all downsampling stages use max pooling. Specifically, to show the impact of encoder and decoder features, in the segmentation experiments, we use CARAFE as the baseline but only modify the source of features used for generating upsampling kernels. We build three baselines: (1) *decoder-only*, the standard implementation of CARAFE; (2) *encoder-only*, where the upsampling kernels are generated from encoder features; (3) *encoder-decoder*, where the upsampling kernels are generated from the con-

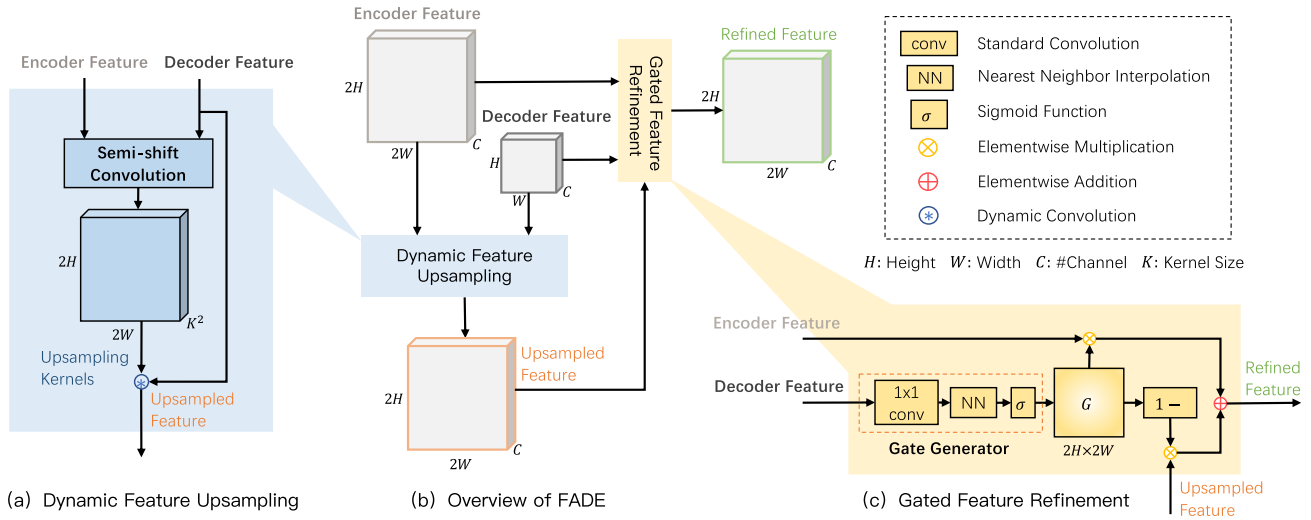

Fig. 4 Technical pipeline of FADE. From **b** the overview of FADE, FADE upsamples the low-res decoder feature with the help of the high-res encoder features. The two types of features are fed into two key modules. In **a** dynamic feature upsampling, the features are used to generate upsampling kernels using a semi-shift convolutional operator (Fig. 6). The kernels are then applied to the decoder feature to generate the upsampled feature. In **c** gated feature refinement, the encoder and upsampled features are modulated by a decoder-dependent gating mechanism to enhance detail delineation before outputting the final refined feature

Table 1 Results of semantic segmentation on SUN RGBD and image reconstruction on Fashion MNIST

	Segmentation Accuracy ↑		Reconstruction Accuracy ↑		Error ↓	
	mIoU	bIoU	PSNR	SSIM	MAE	MSE
Decoder-only	37.00	25.61	24.35	87.19	0.0357	0.0643
Encoder-only	36.71	27.89	32.25	97.73	0.0157	0.0257
Encoder-decoder	**37.59**	**28.80**	**33.83**	**98.47**	**0.0122**	**0.0218**

Best performance is in boldface

catenation of encoder and NN-interpolated decoder features. We report Mask IoU (mIoU) (Everingham et al., 2010) and Boundary IoU (bIoU) (Cheng et al., 2021) for segmentation, and Peak Signal-to-Noise Ratio (PSNR), Structural SIMilarity index (SSIM), Mean Absolute Error (MAE), and root Mean Square Error (MSE) for reconstruction. From Table 1, one can observe that the encoder-only baseline outperforms the decoder-only one in image reconstruction, but in semantic segmentation the trend is on the contrary. To understand why, we visualize the segmentation masks and reconstructed results in Fig. 5. We find that in segmentation the decoder-only model tends to produce regionally coherent masks, while the encoder-only one generates clear mask boundaries but blocky regions; in reconstruction, by contrast, the decoder-only model almost fails and can only generate low-fidelity reconstructions. It thus can be inferred that, high-res encoder features help to predict details, while low-res decoder features contribute to semantic preservation of regions. Indeed, by considering both encoder and decoder features, the resulting mask seems to integrate the merits of the former two, and the reconstructions are also full of details.

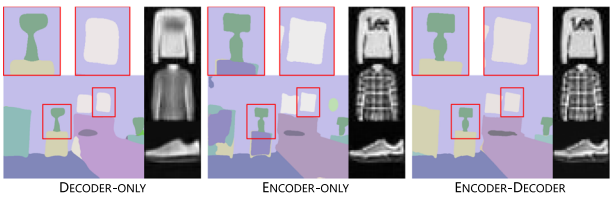

Fig. 5 Visualizations of inferred mask and reconstructed results on SUN RGBD and Fashion-MNIST. The decoder-only model generates semantically consistent mask predictions but poor reconstructions, while the encoder-only one is on the contrary. When both encoder and decoder features are considered, the model generates reasonable masks as the decoder-only model and clear reconstructions as the encoder-only one (cf. the table lamp and the stripes on clothes)

Therefore, albeit a simple tweak, FADE significantly benefits from generating upsampling kernels with both encoder and decoder features, as illustrated in Fig. 2c.

4.3 Semi-shift Convolution

Given encoder and decoder features, we next address how to use them to generate upsampling kernels. We investigate two

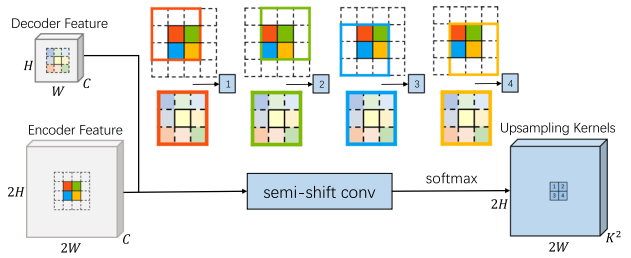

Fig. 6 Upsampling kernel generation using semi-shift convolution with both encoder and decoder features. In contrast to naive implementation (Fig. 3), semi-shift convolution carefully controls the per-point contribution to the kernel (see how each decoder feature point corresponds to each encoder feature point) and unifies feature interpolation, concatenation, channel compression, and kernel prediction

implementations: the naive one presented in Fig. 3 and our customized one–semi-shift convolution. We first illustrate the principle of semi-shift convolution and then present its efficient implementations. Finally, we compare the computational workload and memory occupation among different implementations.

4.3.1 Principle of Semi-shift Convolution

The key difference between naive and semi-shift convolution is how each decoder feature point spatially corresponds to each encoder feature point. The naive implementation shown in Fig. 3 includes five operations: (i) feature interpolation, (ii) concatenation, (iii) channel compression, (iv) standard convolution for kernel generation, and (v) softmax normalization. As aforementioned in Sect. 3, naive interpolation can have a few problems. To address them, we propose semi-shift convolution that simplifies the first four operations above into a unified operator, which is illustrated in Fig. 6. Note that the 4 convolution windows in encoder features all correspond to the same window in decoder features. This design has the following advantages: (1) the role of control in the kernel generation is made clear where the control of the variance of 2×2 neighbors is moved to encoder features completely; (2) the receptive field of decoder features is kept consistent with that of encoder features; (3) memory cost is reduced, because semi-shift convolution directly operates on low-res decoder features, without feature interpolation; (4) channel compression and kernel generation can also be merged in semi-shift convolution.

Mathematically, the single window processing with naive implementation or semi-shift convolution has an identical form if ignoring the content of feature maps. For example, considering the top-left window w.r.t. the index '1' in Figs. 3 and 6, the (unnormalized) upsampling kernel takes the form

$$w_m = \sum_{l=1}^{d}\sum_{i=1}^{h}\sum_{j=1}^{h}\beta_{ijlm}\left(\sum_{k=1}^{2C}\alpha_{kl}x_{ijk} + a_l\right) + b_m$$

$$= \sum_{l=1}^{d}\sum_{i=1}^{h}\sum_{j=1}^{h}\beta_{ijlm}\left(\sum_{k=1}^{C}\alpha_{kl}^{\mathrm{en}}x_{ijk}^{\mathrm{en}} + \sum_{k=1}^{C}\alpha_{kl}^{\mathrm{de}}x_{ijk}^{\mathrm{de}} + a_l\right) + b_m$$

$$= \sum_{l=1}^{d}\sum_{i=1}^{h}\sum_{j=1}^{h}\beta_{ijlm}\sum_{k=1}^{C}\alpha_{kl}^{\mathrm{en}}x_{ijk}^{\mathrm{en}} + \sum_{l=1}^{d}\sum_{i=1}^{h}\sum_{j=1}^{h}\beta_{ijlm}\left(\sum_{k=1}^{C}\alpha_{kl}^{\mathrm{de}}x_{ijk}^{\mathrm{de}} + a_l\right) + b_m, \quad (1)$$

where $w_m, m = 1, ..., K^2$, is the weight of the upsampling kernel, K the upsampling kernel size, h the convolution window size, C the number of input channel dimension of encoder and decoder features, and d the number of compressed channel dimension. $\alpha_{kl}^{\mathrm{en}}$ and $\{\alpha_{kl}^{\mathrm{de}}, a_l\}$ are the parameters of 1×1 convolution specific to encoder and decoder features, respectively, and $\{\beta_{ijlm}, b_m\}$ the parameters of 3×3 convolution. Following CARAFE, we set $h = 3$, $K = 5$, and $d = 64$.

4.3.2 Efficient Implementations of Semi-shift Convolution

Given the formulation above, here we discuss the efficient implementations of semi-shift convolution. According to Eq. (1), by the linearity of convolution, the two standard convolutions on $2C$-channel features are equivalent to applying two distinct 1×1 convolutions to C-channel encoder and C-channel decoder features, respectively, followed by a shared 3×3 convolution and summation. Such decomposition allows us to process encoder and decoder features without matching their resolution explicitly. However, we still need to address the mismatch implicitly. There are two strategies: i) downsampling the high-res encoder output to match the low-res decoder one, or ii) upsampling the low-res decoder output to match the high-res encoder one.

To process the whole feature map following the first strategy, the window can move s steps on encoder features but only $\lfloor s/2 \rfloor$ steps on decoder features. This is why the operator is given the name 'semi-shift convolution'. We split the process to 4 sub-processes; each sub-process focuses on the top-left, the top-right, the bottom-left, and the bottom-right window, respectively. Different sub-processes have similar prepossessing strategies. For example, for the top-left sub-process, we add full zero padding to the decoder feature, but only pad the top and left side of the encoder feature. Then all the top-left window correspondences can be satisfied by setting convolutional stride of 1 for the decoder feature and of 2 for the encoder feature. Finally, after a few memory operations, the four sub-outputs can be reassembled to the (unnormalized) upsampling kernel. This process is illustrated in the left of Fig. 7, which can be called the high-to-low (H2L) implementation.

The H2L implementation above is provided in our conference version (Lu et al., 2022b). We later notice that

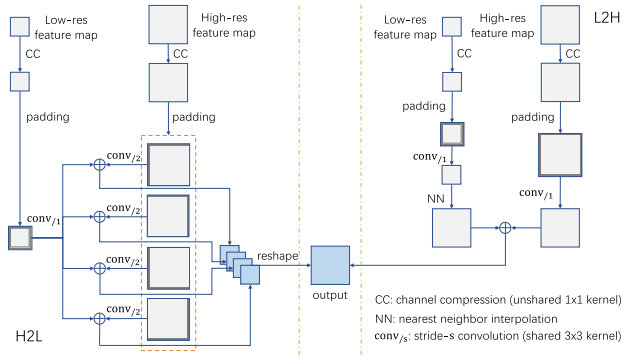

Fig. 7 Fast implementations of semi-shift convolution. We present two forms of fast implementations: (left: H2L) high resolution matches low resolution, which is presented in our conference version (Lu et al., 2022b), and (right: L2H) low resolution matches high resolution, which is more memory efficient

the key characteristic of semi-shift convolution lies in *the same decoder feature point corresponds to* 4 *encoder feature points*, which shares the same spirit of NN interpolation. Following this interpretation, we provide a more efficient implementation with less use of memory, as shown in the right of Fig. 7, named the low-to-high (L2H) implementation. First, unshared 1×1 convolutions are used to compress the encoder and decoder features, respectively. Then the shared 3×3 convolution is applied, of which the decoder feature is NN-interpolated to the size of the encoder one. Finally they are summed to obtain the (unnormalized) kernel.

Both implementations can be implemented within the standard `PyTorch` library. In the H2L implementation, the kernel $\mathcal{W}_i$ of the i-th sub-process (with specific padding applied), $i = 1, 2, 3, 4$, takes the form

$$\begin{aligned}\mathcal{W}_i &= \texttt{conv}_{/2}(\texttt{CC}(\mathcal{X}_{\texttt{en}}, \theta_{en}), \theta) \\ &\quad + \texttt{conv}_{/1}(\texttt{CC}(\mathcal{X}_{\texttt{de}}, \theta_{\texttt{de}}), \theta),\end{aligned} \tag{2}$$

where $\texttt{conv}_{/s}(\mathcal{X}, \theta)$ denotes the stride-s 3×3 convolution over the feature map $\mathcal{X}$, parameterized by θ. `CC` is the channel compressor implemented by 1×1 convolution. $\mathcal{X}_{\texttt{en}}$ and $\mathcal{X}_{\texttt{de}}$ are the encoder and the decoder feature, respectively. Note that, the parameters $\theta_{\texttt{en}}$ and $\theta_{\texttt{de}}$ in `CC` are different, while the parameters in $\texttt{conv}_{/1}$ and $\texttt{conv}_{/2}$ are the same θ. The four $\mathcal{W}_i$'s need to be aggregated and reshaped to form the full kernel $\mathcal{W}$.

In contrast, the L2H implementation does not require sub-process division and computes the full kernel $\mathcal{W}$ directly. It can be formulated as

$$\begin{aligned}\mathcal{W} &= \texttt{conv}_{/1}(CC(\mathcal{X}_{en}, \theta_{en}), \theta) \\ &\quad + NN(\texttt{conv}_{/1}(\texttt{CC}(\mathcal{X}_{\texttt{de}}, \theta_{\texttt{de}}), \theta)),\end{aligned} \tag{3}$$

where `NN` is the $\times 2$ NN interpolation operator.

Table 2 Results on the Weizmann Horse dataset

SegNet – baseline	mIoU
Unpooling	93.42
IndexNet (Lu et al., 2019)	93.00
NN	89.15
CARAFE (Wang et al., 2019)	89.29
NN + Gate	95.26
CARAFE + Gate	95.25

SemiShift-Lite and FADE-Lite. We also investigate a simplified variant of semi-shift convolution, which uses depthwise convolution to further reduce the computational complexity, named SemiShift-Lite. Specifically, SemiShift-Lite sets $d = K^2$ and adopts 3×3 depthwise convolution to encode the local information. Its whole number of parameters is $2CK^2 + 9K^2$. The use of SemiShift-Lite also leads to a lightweight variant of FADE, i.e., FADE-Lite. We use this variant to show that the task-agnostic property indeed comes with the careful treatment of encoder and decoder features, even with much less parameters. When $C = 256$, $d = 64$, and $K = 5$, despite FADE-Lite only includes 27.6% parameters of its standard version FADE, we observe that FADE-Lite is still task-agnostic and outperforms most upsampling operators (see Sect. 5 for details).

4.4 Extracting Fine Details from Encoder Features

Here we further introduce a gating mechanism to complement fine details from encoder features to upsampled features. We again use some experimental observations to motivate our design. We use a binary image segmentation dataset, Weizmann Horse (Borenstein & Ullman, 2002). The reasons for choosing this dataset are two-fold: (1) the visualization is made simple; (2) the task is simple such that the impact of feature quality can be neglected. When all baselines have nearly perfect region predictions, the difference in detail prediction can be amplified. We use SegNet pretrained on ImageNet as the baseline and alter only the upsampling operators. Results are listed in Table 2. An interesting phenomenon is that CARAFE works almost the same as NN interpolation and even falls behind the default unpooling and IndexNet. An explanation is that the dataset is too simple such that the region smoothing property of CARAFE is wasted, but recovering details matters.

A common sense in segmentation is that, the interior of a certain class would be learned fast, while mask boundaries are difficult to predict. This can be observed from the gradient maps w.r.t. an intermediate decoder layer, as shown in Fig. 8. During the middle stage of training, most responses are near boundaries. Now that gradients reveal the demand of

Fig. 8 Gradient maps and gate maps of horses

detail information, feature maps would also manifest this requisite with some distributions, e.g., in multi-class semantic segmentation a confident class prediction in a region would be a unimodal distribution along the channel dimension, and an uncertain prediction around boundaries would likely be a bimodal distribution. Hence, we assume that all decoder layers have gradient-imposed distribution priors and can be encoded to inform the requisite of detail or semantic information. In this way fine details can be chosen from encoder features without hurting the semantic property of decoder features. Hence, instead of directly skipping encoder features as in feature pyramid networks (FPNs) (Lin et al., 2017b), we introduce a naive gating mechanism following existing ideas (Cho et al., 2014; Li et al., 2020c, 2023) to refine upsampled features using encoder features, conditioned on decoder features. The gate is generated through a 1×1 convolution layer, a NN interpolation layer, and a `sigmoid` function. As shown in Fig. 4c, the decoder feature first goes through the gate generator, and the generator then outputs a gate map instantiated in Fig. 8. Finally, the gate map G modulates the encoder feature $\mathcal{F}_{\texttt{encoder}}$ and the upsampled feature $\mathcal{F}_{\texttt{upsampled}}$ to generate the final refined feature $\mathcal{F}_{\texttt{refined}}$ as

$$\mathcal{F}_{\texttt{refined}} = \mathcal{F}_{\texttt{encoder}} \cdot G + \mathcal{F}_{\texttt{upsampled}} \cdot (1 - G). \quad (4)$$

From Table 2, the gate works on both NN and CARAFE.

We remark that our initial motivation for developing the gating mechanism comes from semantic segmentation and image matting tasks. In semantic segmentation, the model outputs a set of logits and uses argmax to select one channel as the predicted class. This form of prediction renders the model working in a one-class-one-value manner. To preserve this manner, we expect the gate to extract only the details that require from the encoder (Fig. 8) and to influence the decoder feature as less as possible. Similarly in matting, despite the number of classes can be considered to be infinity, the model still follows the one-class-one-value paradigm. However, in instance-sensitive tasks, such as object detection, given the one-class-one-value feature maps, one cannot tell the instance difference with argmax. In addition, object detection is rather different from semantic segmentation, where high-res features are responsible for precise localization, so in (Lin et al., 2017b) the FPN is adopted to improve Faster-RCNN (Ren et al., 2015). For the reasons above, gating, as a mechanism strengthening decoder features, may not tackle the improvement for localization. In this case, FADE without gating, denoted by FADE (G=1), would be a better choice. We will discuss more in the experiments on object detection (Sect. 5.3) and instance segmentation (Sect. 5.4).

5 Applications

Here we demonstrate the applications and the task-agnostic property of FADE on various dense prediction tasks, including semantic segmentation, image matting, object detection, instance segmentation, and monocular depth estimation. In particular, we focus our experiments on segmentation to analyze the the upsampling behaviors of FADE from different aspects and design ablation studies to justify our design choice on FADE.

5.1 Semantic Segmentation

Semantic segmentation is region-sensitive. To prove that FADE is architecture-independent, SegFormer (Xie et al., 2021) and UPerNet (Xiao et al., 2018) are chosen as transformer and convolutional baselines, respectively.

5.1.1 Data Set, Metrics, Baseline, and Protocols

We use the ADE20K dataset (Zhou et al., 2017). It covers 150 fine-grained semantic concepts, including 20, 210 training images and 2, 000 validation images. In addition to reporting the standard mask IoU (mIoU) (Everingham et al., 2010), we also report the boundary IoU (bIoU) (Cheng et al., 2021) to assess the boundary quality.

SegFormer-B1 (Xie et al., 2021) is first evaluated. We keep the default model architecture in SegFormer except for modifying the upsampling stages in the MLP head. In particular, feature maps of each scale need to be upsampled to 1/4 of the original image. Therefore, there are $3+2+1 = 6$ upsampling stages in all. All training settings and implementation details are kept the same as in (Xie et al., 2021). Since SegFormer follows a 'fuse-and-concatenate' manner, where the feature maps are all upsampled to the max-resolution one, we verify two styles of upsampling strategies: direct upsampling and 2 by 2 iterative upsampling. We also test B3, B4, and B5 versions of SegFormer to see if a similar boost could be observed on stronger backbones. In addition, considering that stronger backbones often produce better feature quality, this also allows to see whether feature upsampling still contributes to improved feature quality on stronger backbones.

For UPerNet (Xiao et al., 2018), we use the implementation provided by `mmsegmentation`.[1] We use the ResNet-

[1] https://github.com/open-mmlab/mmsegmentation.

Table 3 Semantic segmentation and image matting results on the ADE20K and Adobe Composition-1K data sets

SegFormer-B1/ A2U Matting-R34	Segmentation – accuracy metric ↑				Matting – error metric ↓					
	mIoU	bIoU	GFLOPs	Params	SAD	MSE	Grad	Conn	GFLOPs	Params
Bilinear	41.68	27.80	15.9	13.7M	37.31	0.0103	21.38	35.39	8.6	8.1M
CARAFE (Wang et al., 2019)	42.82	29.84	+1.5	+0.4M	41.01	0.0118	21.39	39.01	+6.0	+0.3M
IndexNet (Lu et al., 2019)	41.50	28.27	+30.7	+12.6M	34.28	0.0081	15.94	31.91	+31.7	+12.3M
A2U (Dai et al., 2021)	41.45	27.31	+0.4	+0.1M	32.15	0.0082	16.39	29.25	+0.7	+38K
SAPA (Lu et al., 2022c)	43.20	30.96	+1.7	+0.2M	31.19	0.0079	15.48	28.30	+1.8	+0.1M
FADE	**44.41**	**32.65**	+2.7	+0.3M	**31.10**	**0.0073**	**14.52**	**28.11**	+8.9	+0.1M
FADE-Lite	43.49	31.55	+0.9	+80K	31.36	0.0075	14.83	28.21	+1.5	+27K

For IndexNet, we use the 'M2O' version in matting and 'HIN' in segmentation following (Lu et al., 2022a). GFLOPs and Param indicate the additional floating-point calculations and additional number of parameters compared with the bilinear baseline. Best performance is in boldface and second best is underlined

Table 4 Semantic segmentation results on the ADE20K data set with different SegFormer backbones

SegFormer	Backbone	Params	mIoU	bIoU
Bilinear	B1	13.7 M	41.68	27.80
FADE	B1	+0.4 M	44.41 (+2.73)	32.65 (+4.85)
Bilinear	B3	47.3 M	49.04	35.24
FADE	B3	+0.7 M	49.05 (+0.01)	36.79 (+1.55)
Bilinear	B4	64.1 M	49.93	35.63
FADE	B4	+0.7 M	50.11 (+0.18)	37.39 (+1.76)
Bilinear	B5	84.7 M	51.00	37.81
FADE	B5	+0.7 M	50.90 (−0.10)	38.91 (+1.10)

50 and ResNet-101 backbones and modify the upsampling operators in the FPN and train the model with 80 K iterations. The original skip connection is removed due to the inclusion of the gating mechanism. Because FADE upsamples by $\times 2$ times of the input at once, we use the aligned resizing in inference to match the resolution. Other settings are kept the same.

5.1.2 Semantic Segmentation Results

Quantitative results of different upsampling operators are reported in Table 3. FADE is the best performing operator on both mIoU and bIoU metrics. In particular, it improves over the Bilinear baseline by a large margin, with +2.73 mIoU and +4.85 bIoU. Qualitative results are shown in Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. FADE generates high-quality predictions both within mask regions and near mask boundaries.

Stronger Backbones. We also test stronger backbones on SegFormer, including the B3, B4, and B5 versions. From Table 4, when stronger backbones are used, we observe both mIoU and bIoU improve (B1→B3, B3→B4, and B4→B5). However, on B3, B4, and B5, the benefits of FADE are almost invisible in terms of mIoU, which suggests improved feature quality brought by improved backbones have addressed many misclassifications that upsampling can amend, particularly for interior regions. Yet, steady boosts in bIoU (> 1) can still be observed. This means improved features only address the boundary errors to a certain degree (cf. bIoU improvements in B1→B3 vs. that in B3→B4), and FADE can still improve feature quality near mask boundaries. Our evaluations connote improved feature upsampling indeed makes a difference, particularly being useful for resource-constrained applications where a model has limited capacity.

Upsampling Styles. We also explore two styles of upsampling in SegFormer: direct upsampling and iterative $\times 2$ upsampling. From Table 5 we can see that iterative upsampling is better than the direct one in performance. Compared with CARAFE, FADE is more sensitive to the upsampling style, which implies the occurrence of features of different scales matters.

Applicability to CNN Architecture. We further evaluate FADE on UPerNet. Results are shown in Table 6. Compared with Bilinear, FADE boosts around +1% mIoU and outperforms the strong baseline CARAFE with ResNet-50, which confirms the efficacy of FADE for the FPN architecture. On the ResNet-101 backbone, FADE also works, and we observe a even more significant improvement in bIoU, which suggests FADE is good at amending boundary errors.

Visualization of Learned Upsampling. We also visualize the learning process of CARAFE and FADE with increased iterations. From Fig. 9, we can see that the two upsampling operators have different behaviors: FADE first learns to delin-

Table 5 SegFormer with direct or iterative upsampling of FADE and CARAFE

SegFormer-B1	mIoU	
	Direct	Iterative
CARAFE (Wang et al., 2019)	42.67	42.82
FADE	42.89	44.41

Table 6 Semantic segmentation results with UPerNet

UPerNet	Backbone	mIoU	bIoU
Bilinear	R50	41.09	28.04
CARAFE (Wang et al., 2019)	R50	41.49	28.29
FADE	R50	**42.18**	**28.72**
Bilinear	R101	43.33	30.21
FADE	R101	**44.27**	**31.31**

Best performance is in boldface

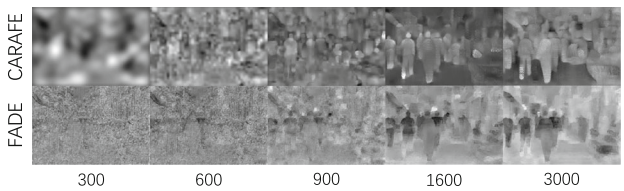

Fig. 9 Learned upsampled feature maps with increased iterations. The learning process between CARAFE and FADE is different. FADE first delineates the outlines of objects and then fills the interior regions, while CARAFE starts from the interior and then spreads outside

eate the outlines of objects and then fills the interior regions, while CARAFE focuses on the interior initially and then spreads outside slowly. We think the reason is that the gating mechanism is relatively simple and learns fast. By the way, one can see that there are checkerboard artifacts in the visualizations of CARAFE (on the leg of the bottom left person) due to the adoption of Pixel Shuffle. Such visualizations suggest that upsampling can significantly affect the quality of features. While there is no principal rule on what could be called 'good features', feature visualizations still proffer a good basis of the feature quality, and one at least can sense where is wrong when clear artifacts present in visualizations.

5.2 Image Matting

Our second task is image matting (Xu et al., 2017). Image matting is a typical detail-sensitive task. It requires a model to estimate an accurate alpha matte that smoothly splits foreground from background. Since ground-truth alpha mattes can exhibit significant differences among local regions, estimations are sensitive to a specific upsampling operator used (Lu et al., 2019; Dai et al., 2021).

5.2.1 Data Set, Metrics, Baseline, and Protocols

We conduct experiments on the Adobe Image Matting dataset (Xu et al., 2017), whose training set has 431 unique foreground objects and ground-truth alpha mattes. Following (Dai et al., 2021), instead of compositing each foreground with fixed 100 background images chosen from MS COCO (Lin et al., 2014), we randomly choose background images in each iteration and generate composited images on-the-fly. The Composition-1K testing set has 50 unique foreground objects, and each is composited with 20 background images from PASCAL VOC (Everingham et al., 2010). We report the widely used Sum of Absolute Differences (SAD), Mean Squared Error (MSE), Gradient (Grad), and Connectivity (Conn) (Rhemann et al., 2009).

A2U Matting (Dai et al., 2021) is adopted as the baseline. Following (Dai et al., 2021), the baseline network adopts a backbone of the first 11 layers of ResNet-34 with in-place activated batchnorm (Bulo et al., 2018) and a decoder consisting of a few upsampling stages with shortcut connections. Readers can refer to (Dai et al., 2021) for the detailed architecture. We use max pooling in downsampling stages when applying FADE as the upsampling operator to train the model, and cite the results of other upsampling operators from A2U Matting (Dai et al., 2021). We strictly follow the training configurations and data augmentation strategies used in (Dai et al., 2021).

5.2.2 Image Matting Results

We compare FADE with other state-of-the-art upsampling operators. Quantitative results are also shown in Table 3. Akin to segmentation, FADE consistently outperforms other competitors in all metrics, with also few additional parameters. Note that IndexNet and A2U are strong baselines that are delicately designed upsampling operators for image matting. Also the worst performance of CARAFE indicates that upsampling with only decoder features is not sufficient to recover details. Compared with standard bilinear upsampling, FADE invites 16–32% relative improvements, which suggests a simple upsampling operator can make a difference. Our community may shift more attention to upsampling. Additionally, it is worth noting that FADE-Lite also outperforms other prior operators, and particularly, surpasses the strong baseline A2U with even less parameters. Qualitative results are shown in Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. FADE generates high-fidelity alpha mattes.

Task-Agnostic Property. By comparing different upsampling operators across both segmentation and matting, FADE is the only operator that exhibits the task-agnostic property. A2U is the previous best operator in matting, but turns out to be the worst one in segmentation. CARAFE is the previous best operator in segmentation, but the worst one in mat-

Table 7 Object detection results with Faster R-CNN on MS-COCO

Faster RCNN (Ren et al., 2015)	Backbone	Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FA2M (Wu et al., 2022)	R50	+23K	37.9	58.8	40.9	22.1	41.7	48.8
FAM (Li et al., 2020b)	R50	+0.8 M	37.8	58.6	41.0	21.8	41.2	48.8
GD-FAM (Li et al., 2023)	R50	+0.5 M	38.1	59.2	41.3	22.7	41.5	49.6
Nearest	R50	46.8 M	37.4	58.1	40.4	21.2	41.0	48.1
CARAFE (Wang et al., 2019)	R50	+0.3 M	**38.6**	**59.9**	**42.2**	**23.3**	**42.2**	**49.7**
IndexNet (Lu et al., 2022a)	R50	+8.4 M	37.6	58.4	40.9	21.5	41.3	49.2
A2U (Dai et al., 2021)	R50	+0.1 M	37.3	58.7	40.0	21.7	41.1	48.5
SAPA (Lu et al., 2022c)	R50	+0.1 M	37.8	59.2	40.6	22.4	41.4	49.1
FADE	R50	+0.2 M	37.8	58.8	40.8	21.2	41.2	49.4
FADE (G=1)	R50	+0.2 M	38.5	59.6	41.8	23.1	**42.2**	49.3
FADE-Lite (G=1)	R50	+52K	38.3	59.5	41.7	22.4	41.9	**49.7**
Nearest	R101	65.8 M	39.4	60.1	43.1	22.4	43.7	51.1
FADE (G=1)	R101	+0.2 M	40.0	61.0	43.3	23.6	44.0	51.1

The 'HIN' version of IndexNet is used. Best performance is in boldface and second best is underlined

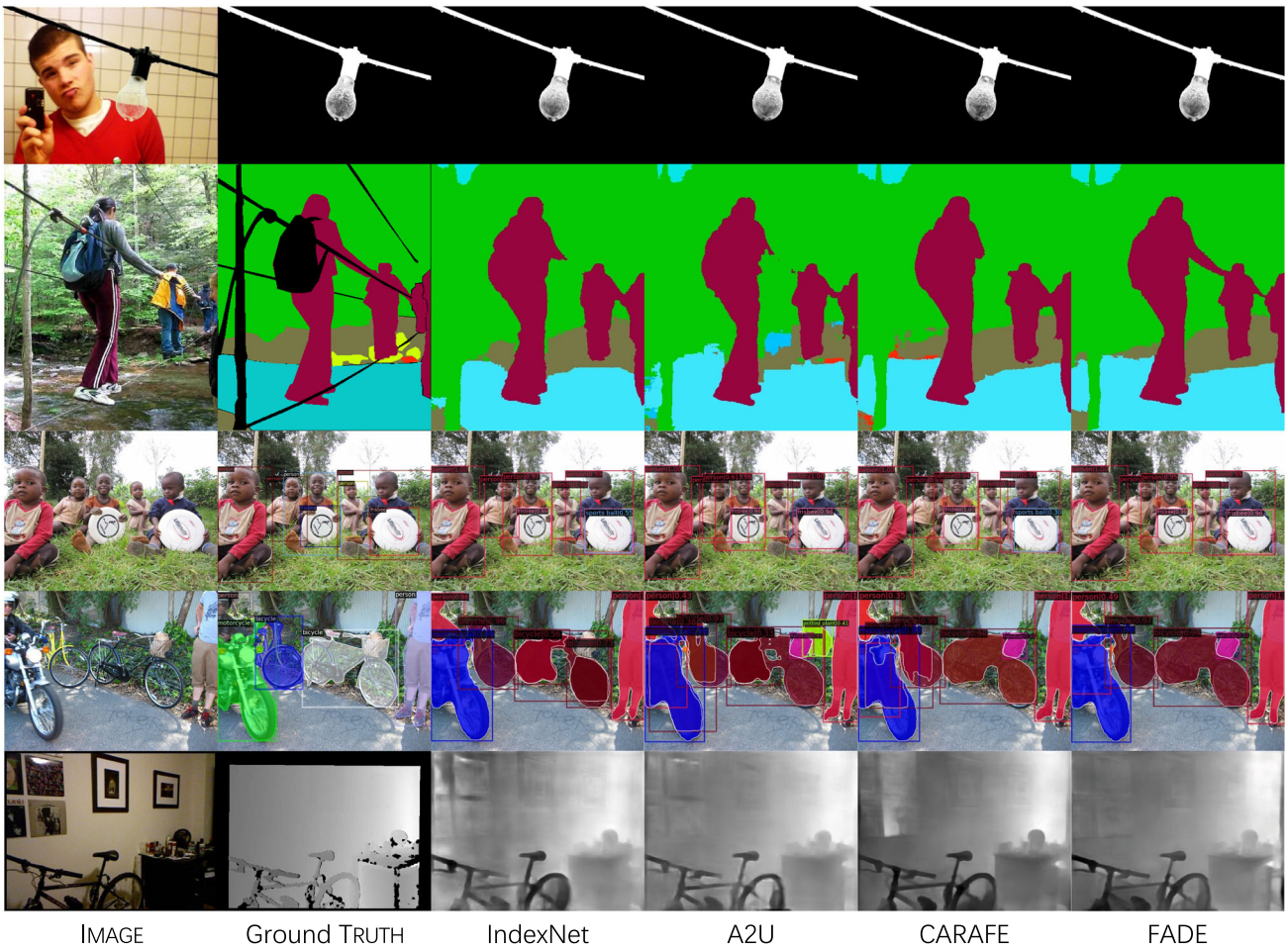

Fig. 10 Qualitative results of different upsampling operators on different dense prediction tasks. Among all competitors, only FADE produces visually pleasing visualizations on both region- and detail-sensitive tasks, e.g., the water drops under the bulb, the hand on the rope, and the (generally) smooth depth values of the wall

Fig. 11 Upsampled feature maps of different upsampling operators on Faster R-CNN

ting. This implies that current dynamic operators still have certain weaknesses to achieve task-agnostic upsampling. In addition, FADE-Lite also exhibits the task-agnostic property (being the consistent second best in both tasks in all metrics), which suggests such a property is insensitive to the number of parameters.

5.3 Object Detection

The third task is object detection (Ren et al., 2015). Object detection addresses where and what objects are with category-specific bounding boxes. It is a mainstream dense prediction problem. Addressing 'what' is a recognition problem, while addressing 'where' requires precise localization in feature pyramids. Upsampling is therefore essential to acquire high-res feature maps.

5.3.1 Data Set, Metrics, Baseline, and Protocols

We use the MS COCO dataset (Lin et al., 2014) and report the standard AP, AP_{50}, AP_{75}, AP_S, AP_M, and AP_L. We use Faster R-CNN as the baseline and replace the default NN interpolation with other upsampling operators. We follow the Faster R-CNN implementation provided by `mmdetection`[2] and only modify the upsampling stages in FPN. Note that, the original skip connection in FPN is removed due to the inclusion of the gating mechanism. All other settings remain unchanged. We evaluate on both ResNet-50 and ResNet-101 backbones. Moreover, since the FPN is used, in addition to the dynamic upsampling operators, we also compare with some feature alignment modules designed for FPN, including the FA^{2}M used in FSANet (Wu et al., 2022), the FAM used in SFNet (Li et al., 2020b), and the GD-FAM used in SFNet-Lite (Li et al., 2023).

5.3.2 Object Detection Results

Quantitative and qualitative results are shown in Table 7 and Fig. 10, respectively. We find that, while FADE still improves detection performance, it is not at a level comparable to CARAFE. However, when setting the gate $G = 1$ in FADE, the performance improves from 37.8 to 38.5 AP, approaching to CARAFE. We are interested to know why. After a careful check at the upsampled feature map (Fig. 11), we see that the detector favors more detailed upsampled features than blurry ones (CARAFE vs. FADE). Perhaps details in features can benefit precise localization of bounding boxes. In the use of CARAFE, high-res encoder features are directly skipped in the FPN. In contrast, FADE uses a gate to control of pass the encoder features. The resulting features of FADE show that the gate does not work as expected: the decoder features dominate in the output. Why does not the gate work? We believe this can boil down to how the detector is supervised. Since the gate predictor has few parameters, the generated gate is mostly affected by the feature map. In semantic segmentation and image matting where per-pixel ground truths are provided, the features can be updated delicately. Yet, in detection where the ground truth bounding boxes are sparse, the feature learning could be coarse, therefore affecting the prediction of the gate. Fortunately, the gating mechanism works in FADE as a post-processing step and can be disabled when unnecessary. In addition, we observe FADE (G=1) outperforms feature alignment modules, which suggests manipulating kernels seems more effective than manipulating features. A plausible explanation is that, feature alignment needs to correct additional artifacts introduced by naive feature upsampling (NN or bilinear upsampling is typically executed before feature alignment is performed). Moreover, with a stronger backbone ResNet-101, FADE can also boost the performance. This implies that, while a better backbone is often favored, there are still feature issues that cannot be addressed with increased model capacity. In this case, some improved components within the architecture such as improved upsampling may help.

5.4 Instance Segmentation

The forth task is instance segmentation (He et al., 2017). Instance segmentation is an extended task of semantic segmentation. In addition to labelling object/scene categories, it needs to further discriminate instances of the same category. It can also be considered a region-sensitive task.

5.4.1 Data Set, Metrics, Baseline, and Protocols

Akin to object detection, we use the MS COCO dataset (Lin et al., 2014) for instance segmentation and report box AP, mask AP, and boundary AP. Following (Wang et al., 2019), we select Mask R-CNN as our baseline and only replace the default NN interpolation with other upsampling operators in the FPN. Since the gate in FADE would reduce to the skip connection when $G = 1$ according to Eq. (4), the original skip connection in FPN is removed. We also follow the Mask R-CNN implementation provided by `mmdetection` and

[2] https://github.com/open-mmlab/mmdetection.

Table 8 Instance segmentation results with Mask R-CNN (ResNet50 as the backbone) on MS-COCO

Mask R-CNN (He et al., 2017)		Bbox metric					
Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FA2M (Wu et al., 2022)	R50	38.8	59.7	42.1	22.7	42.2	50.3
FAM (Li et al., 2020b)	R50	38.4	59.2	41.7	21.9	41.6	49.9
GD-FAM (Li et al., 2023)	R50	38.7	59.4	42.2	22.4	41.7	50.7
Nearest	R50	38.3	58.7	42.0	21.9	41.8	50.2
CARAFE (Wang et al., 2019)	R50	**39.2**	**60.3**	**42.9**	**23.4**	42.5	**51.2**
IndexNet (Lu et al., 2022a)	R50	38.4	59.1	42.1	22.1	42.0	50.2
A2U (Dai et al., 2021)	R50	37.9	59.0	40.8	22.0	41.5	49.4
SAPA (Lu et al., 2022c)	R50	38.7	59.7	42.2	23.1	41.8	49.9
FADE	R50	38.7	59.3	42.0	22.6	42.4	50.4
FADE (G=1)	R50	**39.2**	**60.3**	42.7	23.2	**42.6**	51.0
FADE-Lite (G=1)	R50	38.9	60.0	42.3	23.2	42.2	50.9
Nearest	R101	40.0	60.4	43.7	22.8	43.7	52.0
FADE (G=1)	R101	40.6	61.5	44.3	24.1	44.4	53.1
		Segm metric					
Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FA2M (Wu et al., 2022)	R50	35.3	56.4	37.6	16.8	37.6	**52.1**
FAM (Li et al., 2020b)	R50	34.8	56.0	36.9	15.9	37.1	50.9
GD-FAM (Li et al., 2023)	R50	35.0	56.1	37.3	16.6	37.1	51.4
Nearest	R50	34.7	55.8	37.2	16.1	37.3	50.8
CARAFE (Wang et al., 2019)	R50	**35.5**	**57.1**	**37.8**	**17.3**	**37.9**	51.9
IndexNet (Lu et al., 2022a)	R50	34.8	55.9	37.0	16.3	37.3	51.3
A2U (Dai et al., 2021)	R50	34.4	55.7	36.8	15.7	37.1	50.6
SAPA (Lu et al., 2022c)	R50	35.3	56.7	37.6	16.9	**37.9**	50.7
FADE	R50	34.7	55.9	36.9	15.9	37.2	51.0
FADE (G=1)	R50	35.2	56.8	37.5	16.8	37.7	51.4
FADE-Lite (G=1)	R50	35.3	56.7	37.6	17.2	37.7	51.7
Nearest	R101	36.0	57.6	38.5	16.5	39.3	52.2
FADE (G=1)	R101	36.4	58.0	38.9	17.4	39.3	53.3
		Boundary metric					
Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FA2M (Wu et al., 2022)	R50	21.1	46.2	16.8	16.8	31.4	**20.6**
FAM (Li et al., 2020b)	R50	20.7	45.8	16.4	15.9	31.0	19.9
GD-FAM (Li et al., 2023)	R50	20.9	46.2	16.3	16.5	30.9	20.1
Nearest	R50	20.7	45.5	16.6	16.0	31.2	20.0
CARAFE (Wang et al., 2019)	R50	**21.3**	**47.0**	**16.9**	**17.3**	31.5	**20.6**
IndexNet (Lu et al., 2022a)	R50	20.8	45.8	16.4	16.3	31.2	20.3
A2U (Dai et al., 2021)	R50	20.4	45.1	15.9	15.7	30.8	19.7
SAPA (Lu et al., 2022c)	R50	21.1	46.4	16.6	16.8	**31.7**	19.6
FADE	R50	20.7	45.7	16.3	15.9	31.1	20.3
FADE (G=1)	R50	21.2	46.6	16.7	16.8	31.4	20.4
FADE-Lite (G=1)	R50	21.2	46.7	16.6	17.1	31.5	**20.6**
Nearest	R101	22.1	48.1	17.6	16.4	33.0	21.5
FADE (G=1)	R101	22.2	48.5	17.6	17.4	33.0	21.6

Upsampling operators are replaced in FPN. 'HIN' version of IndexNet is used. The parameter increment is the same as in Faster R-CNN. Best performance is in boldface and second best is underlined

the training setting used in (Wang et al., 2019). We test on both ResNet-50 and ResNet-101 backbones. In addition, we also compare against the feature alignment modules as in detection, because Mask R-CNN uses the FPN as well.

5.4.2 Instance Segmentation Results

Quantitative and qualitative results are shown in Table 8 and Fig. 10, respectively. We have similar observations to object detection: i) the standard implementation of FADE only shows marginal improvements; ii) FADE without gating works better than FADE and is on par with CARAFE. Compared with other tasks, all upsampling operators have limited improvements (< 1) in terms of mask AP. A reason may be the limited output resolution (28×28) of the mask head. In this case, the benefits of improved boundary delineation of upsampling may not be revealed, which can also be observed from the marginal improvements on the boundary AP. Indeed the more significant relative improvements on box AP than mask AP indicate that the improved mask AP could be mostly due to the improved detection performance. Nevertheless, FADE without gating could still be a preferable choice if taking its task-agnostic property into account. With a stronger backbone ResNet-101, FADE invites an improvement of 0.6 box AP and 0.4 mask AP, which provides a similar boost as ResNet-50. Compared with feature alignment modules, dynamic upsampling operators generally work better. From the visualizations of feature maps in Fig. 12, one can see that, despite being empirical, the quality of the feature maps generally seems an good indicator of final performance: feature maps more resembling to the ground truth at the relatively low resolution (the second row) generally have better performance (cf. the feature maps of NN and A2U).

5.5 Monocular Depth Estimation

Our final task is monocular depth estimation (Xian et al., 2018). This task aims to infer the depth from a single image. Compared with other tasks, depth estimation is a mixture of region- and detail-sensitive dense predictions. In a local region, depth values could remain constant (an object plane parallel to the image plane), could be gradually varied (an object plane oblique to the image plane), or could be suddenly changed (on the boundary between different depth planes). The recovery of details in depth estimation is also critical for human perception, because boundary artifacts can be easily perceived by human eyes in many depth-related applications such as 3D ken burns (Niklaus et al., 2019) and bokeh rendering (Peng et al., 2022).

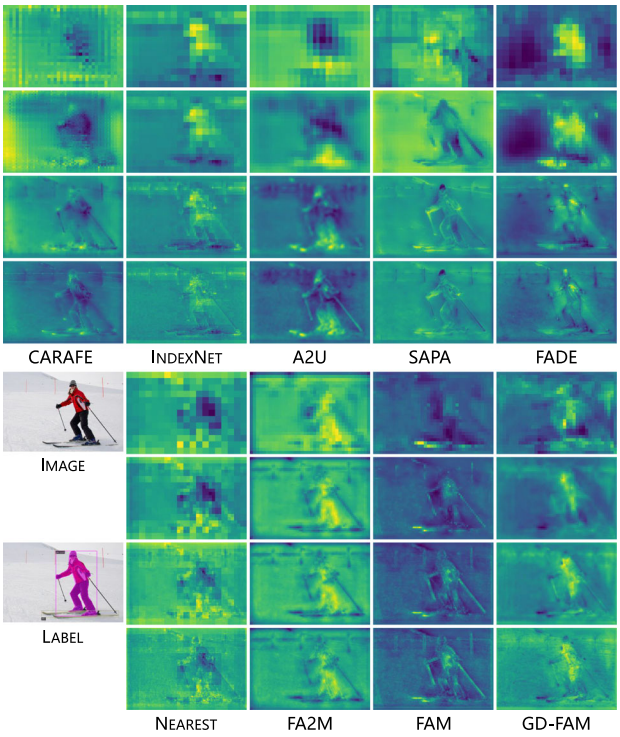

Fig. 12 Visualizations of upsampled feature maps generated by different methods. The feature maps are extracted from the output of upsamplers in Mask R-CNN-R50 (He et al., 2017). The quality of feature maps generally provides an implication of performance

5.5.1 Data Set, Metrics, Baseline, and Protocols

We use the NYU Depth V2 (Silberman et al., 2012) dataset and standard depth metrics used by previous work to evaluate the performance, including root mean squared error (RMS) and its log version (RMS (log)), absolute relative error (Abs Rel), squared relative error (Sq Rel), average $\log_{10}$ error (log10), and the accuracy with threshold thr ($\delta < thr$). Readers can refer to (Lee et al., 2019) for definitions of the metrics. We use BTS[3] as our baseline and modify all the upsampling stages except for the last one, because there is no guiding feature map at the last stage. We follow the default training setting provided by the authors but set the batch size as 4 in our experiments (due to limited computational budgets).

5.5.2 Monocular Depth Estimation Results

Quantitative and qualitative results are shown in Table 9 and Fig. 10, respectively. Note that FADE requires more number of parameters in this task. The reason is that the number of channels in encoder and decoder features are different, and we need a few 1×1 convolutions to adjust the channel

[3] https://github.com/cleinc/bts.

Table 9 Monocular depth estimation results on NYU Depth V2 with BTS

BTS-ResNet50 (Lee et al., 2019)	Params	Accuracy metric ↑			Error metric ↓				
		$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Abs Rel	Sq Rel	RMS	RMS (log)	log10
Nearest	49.5 M	0.865	0.975	0.993	0.119	0.075	0.419	0.152	0.051
CARAFE (Wang et al., 2019)	+0.4 M	0.864	0.974	0.994	0.117	0.071	0.418	0.152	0.051
IndexNet (Lu et al., 2022a)	+44.2 M	0.866	0.976	**0.995**	0.117	0.071	0.416	0.151	0.050
A2U (Dai et al., 2021)	+0.2 M	0.860	0.973	0.993	0.121	0.077	0.429	0.156	0.052
SAPA (Lu et al., 2022c)	+0.2 M	0.871	0.977	0.994	0.116	0.070	0.410	0.151	0.050
FADE	+2.8 M	**0.875**	**0.978**	**0.995**	**0.114**	**0.068**	**0.404**	**0.147**	**0.049**
FADE-Lite	+2.5 M	0.870	0.977	**0.995**	0.115	0.069	0.411	0.150	0.050

'HIN' version of IndexNet is used. Best performance is in boldface and second best is underlined

number for the gating mechanism. Overall, FADE reports consistently better performance in all metrics than other competitors, and FADE-Lite is also the steady second best. It is worth noting that A2U degrades the performance, which suggests only improving detail delineation is not sufficient for depth estimation. FADE, however, fuses the benefits of both detail- and region-aware upsampling capable of simultaneous detail delineation and regional preservation. We believe this is the reason why FADE behaves remarkably on this task.

5.6 Ablation Study

Here we conduct ablation studies to justify our three design choices. We follow the settings in segmentation and matting, because they are sufficiently representative to indicate region- and detail-sensitive tasks. In particular, we explore how performance is affected by the source of features, the way for upsampling kernel generation, and the use of the gating mechanism. We build six baselines:

(1) b1: *encoder-only*. Only encoder features go through 1×1 convolution for channel compression (64 channels), followed by 3×3 convolution layer for kernel generation;
(2) b2: *decoder-only*. This is the CARAFE baseline (Wang et al., 2019). Only decoder features go through the 1×1 and 3×3 convolution for kernel generation, followed by Pixel Shuffle;
(3) b3: *encoder-decoder-naive*. NN-interpolated decoder features are first concatenated with encoder features, and then the same two convolutional layers are applied;
(4) b4: *encoder-decoder-semi-shift*. Instead of using NN interpolation and standard convolutional layers, we use semi-shift convolution to generate kernels as in FADE;
(5) b5: b4 with *skipping*. We directly skip the encoder features as in feature pyramid networks (Lin et al., 2017b);
(6) b6: b4 with *gating*. The full implementation of FADE.

Results are shown in Table 10. By comparing b1, b2, and b3, the results confirm the importance of both encoder and decoder features for upsampling kernel generation. By comparing b3 and b4, semi-shift convolution is superior than naive implementation in the way of generating upsampling kernels. As aforementioned, the rationale behind such a superiority can boil down to the granular control on the per-point contribution in the kernel (Sect. 4). We also note that, even without gating, the performance of FADE already surpasses other upsampling operators (b4 vs. Table 3), which means the task-agnostic property is mainly due to the joint use of encoder and decoder features and the semi-shift convolution. In addition, skipping in these two task is clearly not the optimal way to move encoder details to decoder features, at least worse than the gating mechanism (b5 vs. b6). Hence, we think gating is generally beneficial.

5.7 Limitations and Further Discussions

Computational Overhead. Despite FADE outperforms CARAFE in 4 out of 6 tasks, FADE processes 5 times data more than CARAFE and thus consumes more FLOPs due to the involvement of high-res encoder features. Our efficient implementations do not change this fact but only help prevent extra calculations on interpolated decoder features. A thorough comparison of the computational complexity and inference time of different dynamic upsampling operators can be found in Appendix A.

Prerequisite of Using FADE. The use of the gating mechanism in FADE requires an equal number of channels of encoder and decoder features. Therefore, if the channel number differs, one needs to add a 1×1 convolution layer to align the channel number. However, this would introduce additional parameters, for example depth estimation with BTS. If the gate is not used, i.e., FADE (G=1), this trouble could be saved. In addition, if there is no high-res feature guidance, for instance, the last upsampling stage in BTS or in image super-resolution tasks, FADE cannot be applied as well.

When to Use the Gating Mechanism. At our initial design (Lu et al., 2022b), we mainly consider the one-class-one-value mapping as in semantic segmentation or regressing

Table 10 Ablation study on the source of features, the way for upsampling kernel generation, and the effect of the gating mechanism

No	SegFormer / A2U Matting			Segm – accuracy ↑		Matting – error ↓			
	source of feat	kernel gen	fusion	mIoU	bIoU	SAD	MSE	Grad	Conn
b1	en			42.75	31.00	34.22	0.0087	15.90	32.03
b2	de			42.82	29.84	41.01	0.0118	21.39	39.01
b3	en & de	naive		43.27	31.55	32.41	0.0083	16.56	29.82
b4	en & de	semi-shift		43.33	32.06	31.78	0.0075	15.12	28.95
b5	en & de	semi-shift	skipping	43.22	31.85	32.64	0.0076	15.90	29.92
b6	en & de	semi-shift	gating	**44.41**	**32.65**	**31.10**	**0.0073**	**14.52**	**28.11**

'en' is for encoder, and 'de' for decoder. Best performance is in boldface

a dense 2D map as in image matting, but do not explore instance-level tasks like object detection and instance segmentation, where the situation differs from what we initially claim. We find that the high-res encoder feature plays an important role in localization. If forcing the feature map to be alike to that in semantic segmentation, the model cannot learn instance-aware information effectively. In this case the gating mechanism can fail, and we propose to use direct addition ($G = 1$) as a substitution. One should also be aware that, semi-shift convolution can introduce encoder noise in the generated kernel such that the precise localization of bounding box could be affected (the obviously lower AP_{75} of FADE than CARAFE in object detection and instance segmentation).

General Value of Upsampling to Dense Prediction. As closing remarks, here we tend to share our insights on the general value of upsampling to dense prediction. Compared with other operators or modules studied in dense prediction models, upsampling operators have received less attention. While we have conducted extensive experiments to demonstrate the effectiveness of upsampling, one may still raise the question: Is upsampling an intrinsic factor to influence the dense prediction performance? Indeed current mainstream ideas are to scale the model (Tan & Le, 2019; Zhai et al., 2022), and results from Table 4 also indicate that, under a certain evaluation metric, a strong backbone with a simple bilinear upsampling is sufficient. Yet, we remark that, if one keep pursuing the increment of a certain metric in a specific task, e.g., mIoU in semantic segmentation, some other important things would be overlooked such as the boundary quality. From also Table 4, we can observe that enhanced upsampling steadily boosts the bIoU metric. This is only in segmentation. From a broad view across different dense prediction tasks, the value of upsampling can even be greater, particularly for low-level tasks. For instance, it has been reported that, with learned upsampling, the Deep Image Prior model can use 95% fewer parameters to achieve superior denoising results than existing methods (Liu et al., 2023). Our previous experience in matting also suggests inappropriate upsampling even cannot produce a reasonable alpha prediction (Lu et al., 2022a). From the perspective of architecture design, different operators or modules function differently, but their ultimate goal is alike, i.e., learning high-quality features. If enabling an upsampling operator that has a high probability of being used in an encoder-decoder architecture to have equivalent or even better functions implemented by other optional modules, the architecture design could be simplified. Task-agnostic upsampling at least demonstrates such a potential. Indeed upsampling matters. We believe the value of upsampling is not only about improved performance but also about the design of new, effective, efficient, and generic encoder-decoder architectures.

Another closely-related question is that: Does one still need new fundamental (upsampling) operators, particularly in the era of vision foundation models (Caron et al., 2021; Radford et al., 2021; Rombach et al., 2022; Kirillov et al., 2023) when the idea of scaling typically wins? Indeed current foundation models are made of standard operators such as convolutional layers (Radford et al., 2021) and self-attention blocks (Caron et al., 2021). The classic U-Net architecture (Ronneberger et al., 2015) is also used in StableDiffusion (Rombach et al., 2022). The adoption of sophisticated operators or architectures seem unnecessary if the model capacity reaches to a certain level. Yet, we note a phenomenon that the SAM model (Kirillov et al., 2023) still cannot generate accurate mask boundaries. We believe one of the reasons is that it still uses the deconvolution upsampling in the decoder, which smoothes boundaries. Hence, we think designing fundamental and task-agnostic network operators would remain to be an active research area. Here we make a tentative prediction: a real sense of the vision foundation model should be made of task-agnostic operators. We expect this work can inspire the new design of such operators.

6 Conclusion

In this paper, we provide feature upsampling with three levels of meanings: i) being basic, the ability to increase spatial resolution; ii) being effective, the capability of improving performance; and iii) being task-agnostic, the generality across tasks. In particular, to achieve the third property, we propose FADE, a novel, plug-and-play, and task-agnostic upsampling operator by fully fusing the assets of encoder and decoder features. For the first time, FADE demonstrates that task-agnostic upsampling is made possible across both region- and detail-sensitive dense prediction tasks, outperforming or at least being comparable with the previous best upsampling operators. We explain the rationale of our design with step-to-step analyses and also share our view points from considering what makes for generic feature upsampling. Our core insight is that an upsampling operator should be able to dynamically trade off between detail delineation and semantic preservation in a content-aware manner.

We encourage others to try this operator on many more dense prediction tasks, particularly on low-level tasks such as image restoration. So far, FADE is designed to maintain the simplicity by only implementing linear upsampling, which leaves ample room for further improvement, e.g., by exploring additional nonlinearity.

Appendix A Comparison of Computational Complexity

A favorable upsampling operator, being part of overall network architecture, should not significantly increase the computation cost. This issue is not well addressed in IndexNet as it introduces many parameters and much computational overhead (Lu et al., 2019). In this part we analyze the computational workload and memory occupation among different dynamic upsampling operators. We first compare the FLOPs and number of parameters in Table 11. FADE requires more FLOPs than CARAFE (note that FADE processes 5

Table 11 Computational complexity and parameters of FADE and other upsampling operators

Module	Operation	FLOPs ($\times HW$)	Params
CARAFE	Kernel generation	$Cd+36K^2d$	$Cd+36K^2d$
	Feature assembly	$4K^2C$	0
	Total	$Cd+36K^2d+4K^2C$	$Cd+36K^2d$
IndexNet	Kernel generation	$32C^2+8C$	$32C^2+8C$
HIN	Feature assembly	$4C$	0
	Total	$32C^2+12C$	$32C^2+8C$
IndexNet	Kernel generation	$68C^2$	$68C^2$
M2O	Feature assembly	$4C$	0
	Total	$68C^2+4C$	$68C^2$
A2U	Kernel generation	$73C+4K^2$	$4K^2C+2C$
	Feature assembly	$4K^2C$	0
	Total	$73C+4K^2+4K^2C$	$4K^2C+2C$
SAPA	Kernel generation	$5Cd+4K^2d$	$2Cd$
	Feature assembly	$4K^2C$	0
	Total	$5Cd+4K^2d+4K^2C$	$2Cd$
FADE	Kernel generation	$5Cd+45K^2d$	$2Cd+9K^2d$
	Feature assembly	$4K^2C$	0
	Gated fusion	$9C$	C
	Total	$5Cd+4K^2C+45K^2d+9C$	$2Cd+9K^2d+C$
	Total (G=1)	$5Cd+45K^2d+4K^2C$	$2Cd+9K^2d$
FADE	Kernel generation	$5CK^2+45K^2$	$2CK^2+9K^2$
Lite	Feature assembly	$4K^2C$	0
	Gated fusion	$9C$	C
	Total	$5CK^2+4K^2C+45K^2+9C$	$2CK^2+9K^2+C$
	Total (G=1)	$5CK^2+45K^2+4K^2C$	$2CK^2+9K^2$

C number of channels of encoder and decoder features, d number of compressed channels, K upsampling kernel size, H, W height and width of the **decoder** feature map. G=1 indicates no gating mechanism

Table 12 Comparison of inference time among different upsampling operators

Upsampler	Bilinear	CARAFE	IndexNet	A2U	FADE	FADE-Lite
Time (ms)	1.5	11.1	26.4	24.2	20.2	17.6

Time is tested on a single Nvidia GTX 3090 GPU on a server with Intel Xeon Gold 6226R @ 2.90 GHz CPUs

times more feature data than CARAFE), but less parameters when the number of channels is small. For example, when $C = 256$, $d = 64$, $K = 5$, and $H = W = 112$, CARAFE and FADE cost 2.50 and 4.56 GFLOPs, respectively; the number of parameters are 74 K and 47 K, respectively. FADE-Lite, in the same setting, costs only 1.53 GFLOPs and 13 K parameters. In addition, we also test the inference speed by upsampling a random feature map of size $256 \times 120 \times 120$ (a guiding map of size 256×240 is used if required). The inference time is shown in Table 12. Among compared dynamic upsampling operators, FADE and FADE-Lite are relatively efficient given that they process five times more data than CARAFE. We also test the practical memory occupation of FADE on SegFormer-B1 (Xie et al., 2021), with 6 upsampling stages. Under the default training setting, SegFormer-B1 with bilinear upsampling costs 22, 157 MB GPU memory. With the H2L implementation of FADE, it consumes 24, 879 MB, 2722 MB more than the original one. The L2H one reduces the memory cost by 24.2% (from 2722 to 2064 MB), and is within an acceptable range compared with the decoder-only upsampling operator CARAFE (664 MB) if taking the five times more data into account.

Funding This work is supported by the National Natural Science Foundation of China Under Grant No. 62106080 and the Hubei Provincial Natural Science Foundation of China Under Grant No. 2024AFB566.

References

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans Pattern Anal Mach Intell, 39*(12), 2481–2495.

Borenstein, E., & Ullman, S. (2002). Class-specific, top-down segmentation. *Proc* (pp. 109–122). European Conference on Computer Vision: Springer.

Bulo, S.R., Porzi, L., & Kontschieder, P. (2018). In-place activated batchnorm for memory-optimized training of dnns. In: Proceedings of the IEEE conference on computer vision pattern Recognition, pp. 5639–5647.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE International Conference Computer Vision, pp. 9650–9660.

Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European Conference on Computer Vision, pp. 801–818.

Cheng, B., Girshick, R., Dollár, P., Berg, A.C., & Kirillov, A, (2021). Boundary iou: Improving object-centric image segmentation evaluation. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 15334–15342.

Cheng, T., Wang, X., Huang, L., & Liu, W. (2020). Boundary-preserving mask r-cnn. *Proc* (pp. 660–676). European Conference on Computer Vision: Springer.

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv Comput Res Repository.

Dai, Y., Lu, H., & Shen, C. (2021). Learning affinity-aware upsampling for deep image matting. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 6841–6850.

Dong, C., Loy, C. C., He, K., & Tang, X. (2015). Image super-resolution using deep convolutional networks. *IEEE Trans Pattern Anal Mach Intell, 38*(2), 295–307.

Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In: Annual Conference on Neural Information Processing Systems, pp. 2366–2374.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision, 88*(2), 303–338.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 580–587.

He, K., Sun, J., & Tang, X. (2010). Guided image filtering. *Proc* (pp. 1–14). European Conference on Computer Vision: Springer.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 770–778.

He. K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In: Proceedings of the IEEE conference on computer vision, pp. 2961–2969

Huang, S., Lu, Z., Cheng, R., & He, C. (2021). Fapn: Feature-aligned pyramid network for dense image prediction. In: Proceedings of the IEEE conference on computer vision, pp. 864–873.

Ignatov, A., Timofte, R., Denna, M., & Younes, A. (2021). Real-time quantized image super-resolution on mobile npus, mobile ai 2021 challenge: Report. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, Workshops, pp. 2525–2534.

Kirillov, A., Wu, Y., He, K., & Girshick, R. (2020). Pointrend: Image segmentation as rendering. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 9799–9808.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollar, P., & Girshick, R. (2023). Segment anything. In: Proceedings of the IEEE conference on computer vision, pp. 4015–4026.

Lee, J.H., Han, M.K., Ko, D.W., & Suh, I.H. (2019). From big to small: Multi-scale local planar guidance for monocular depth estimation. arXiv Comput Res Repository.

Li, X., Li, X., Zhang, L., Cheng, G., Shi, J., Lin, Z., Tan, S., & Tong, Y. (2020). Improving semantic segmentation via decoupled body and edge supervision. *Proc* (pp. 435–452). European Conference on Computer Vision: Springer.

Li, X., You, A., Zhu, Z., Zhao, H., Yang, M., Yang, K., Tan, S., & Tong, Y. (2020). Semantic flow for fast and accurate scene parsing. *Proc* (pp. 775–793). European Conference on Computer Vision: Springer.

Li, X., Zhao, H., Han, L., Tong, Y., Tan, S., & Yang, K. (2020). Gated fully fusion for semantic segmentation. *Proceedings of the AAAI Conference on Artificial Intelligence, 34*, 11418–11425.

Li, X., Zhang, J., Yang, Y., Cheng, G., Yang, K., Tong, Y., & Tao, D. (2023). SFNet: Faster and accurate semantic segmentation via semantic flow. *International Journal of Computer Vision, 132*(2), 1–24.

Lin, G., Milan, A., Shen, C., & Reid, I. (2017a). RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 1925–1934.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan. D., Dollár, P., & Zitnick, C.L. (2014). Microsoft coco: Common objects in context. In: Proceedings of the European Conference on Computer Vision, pp. 740–755.

Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017b). Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 2117–2125.

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 8759–8768.

Liu, Y., Li, J., Pang, Y., Nie, D., & Yap, P.T. (2023). The devil is in the upsampling: Architectural decisions made simpler for denoising with deep image prior. In: Proceedings of the IEEE conference on computer vision, pp. 12408–12417.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 3431–3440.

Lu, H., Dai, Y., Shen, C., & Xu, S. (2019). Indices matter: Learning to index for deep image matting. In: Proceedings of the IEEE conference on computer vision, pp. 3266–3275.

Lu, H., Dai, Y., Shen, C., & Xu, S. (2022). Index networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(1), 242–255.

Lu, H., Liu, W., Fu, H., & Cao, Z. (2022b). Fade: Fusing the assets of decoder and encoder for task-agnostic upsampling. In: Proceedings of the European Conference on Computer Vision, pp. 231–247.

Lu, H., Liu, W., Ye, Z., Fu, H., Liu, Y., & Cao, Z. (2022c). SAPA: Similarity-aware point affiliation for feature upsampling. In: Proceedings of the Annual Conference on Neural Information Processing Systems, pp. 20889–20901.

Mao, X., Shen, C., & Yang, Y.B. (2016). Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In: Proceedings of the Annual Conference on Neural Information Processing Systems, pp. 2802–2810.

Mazzini, D. (2018). Guided upsampling network for real-time semantic segmentation. In: Proceedings of British Machine Vision Conference (BMVC), pp. 1–12.

Niklaus, S., Mai, L., Yang, J., & Liu, F. (2019). 3D ken burns effect from a single image. *ACM Trans Graph, 38*(6), 1–15.

Odena, A., Dumoulin, V., & Olah, C. (2016). *Deconvolution and checkerboard artifacts. Distill, 1*(10), e3.

Peng, J., Cao, Z., Luo, X., Lu, H., Xian, K., & Zhang, J. (2022). BokehMe: When neural rendering meets classical rendering. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 16283–16292.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *Proc* (pp. 8748–8763). PMLR: International Conference on Machine Learning

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. Annual Conference on Neural Information Processing Systems 28.

Rhemann, C., Rother, C., Wang, J., Gelautz, M., Kohli, P., & Rott, P. (2009). A perceptually motivated online benchmark for image matting. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 1826–1833.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 10684–10695.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 234–241.

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 1874–1883.

Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In: Proceedings of the European Conference on Computer Vision, pp. 746–760.

Song, S., Lichtenberg, S.P., & Xiao, J. (2015). SUN RGB-D: A RGB-D scene understanding benchmark suite. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 567–576.

Takikawa, T., Acuna, D., Jampani, V., & Fidler, S. (2019). Gated-scnn: Gated shape cnns for semantic segmentation. In: Proceedings of the IEEE conference on computer vision, pp. 5229–5238.

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *Proceedings of the International Conference on Machine Learning, 97*, 6105–6114.

Tang, C., Chen, H., Li, X., Li, J., Zhang, Z., & Hu, X. (2021). Look closer to segment better: Boundary patch refinement for instance segmentation. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 13926–13935.

Teed, Z., & Deng, J. (2020). Raft: Recurrent all-pairs field transforms for optical flow. *Proc* (pp. 402–419). European Conference on Computer Vision: Springer.

Tian, Z., He, T., Shen, C., & Yan, Y. (2019). Decoders matter for semantic segmentation: Data-dependent decoding enables flexible feature aggregation. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 3126–3135.

Tomasi, C., & Manduchi, R. (1998). Bilateral filtering for gray and color images. *Proc* (pp. 839–846). IEEE International Conference on Computer Vision.

Wang, J., Chen, K., Xu, R., Liu, Z., Loy, C.C., & Lin, D. (2019). CARAFE: Context-aware reassembly of features. In: Proc. IEEE/CVF International Conference on Computer Vision (ICCV), pp. 3007–3016.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al. (2020). Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 43*(10), 3349–3364.

Wang, J., Chen, K., Xu, R., Liu, Z., Loy, C. C., & Lin, D. (2021). CARAFE++: Unified Content-Aware ReAssembly of FEatures. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(9), 4674–4687.

Wang, X., Girshick, R., Gupta, A., & He, K. (2018). Non-local neural networks. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 7794–7803.

Wu, J., Pan, Z., Lei, B., & Hu, Y. (2022). Fsanet: Feature-and-spatial-aligned network for tiny object detection in remote sensing images. *Transactions on Geoscience and Remote Sensing, 60*, 1–17.

Xian, K., Shen, C., Cao, Z., Lu, H., Xiao, Y., Li, R., & Luo, Z. (2018). Monocular relative depth perception with web stereo data supervision. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 311–320.

Xiao, H., Rasul, K., & Vollgraf, R. (2017), Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv Comput Res Repository.

Xiao, T., Liu, Y., Zhou, B., Jiang, Y., & Sun, J. (2018). Unified perceptual parsing for scene understanding. In: Proceedings of the European Conference on Computer Vision, pp. 418–434.

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., & Luo, P. (2021). SegFormer: Simple and efficient design for semantic segmentation with transformers. In: Proceedings of the Annual Conference on Neural Information Processing Systems, pp. 12077–12090.

Xie, S., & Tu, Z. (2015). Holistically-nested edge detection. In: Proceedings of the European Conference on Computer Vision, pp. 1395–1403.

Xu, N., Price, B., Cohen, S., & Huang, T. (2017). Deep image matting. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 2970–2979.

Yuan, Y., Huang, L., Guo, J., Zhang, C., Chen, X., & Wang, J. (2021). Ocnet: Object context for semantic segmentation. *International Journal of Computer Vision, 129*(8), 2375–2398.

Zeiler, M.D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In: Proceedings of the European Conference on Computer Vision, pp. 818–833.

Zhai, X., Kolesnikov, A., Houlsby, N., & Beyer, L. (2022). Scaling vision transformers. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 12104–12113.

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 2881–2890.

Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., & Torr, P.H., et al. (2021). Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 6881–6890

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 2921–2929.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., & Torralba, A. (2017). Scene parsing through ade20k dataset. In: Proceedings of the IEEE conference on computer vision and pattern Recognition, pp. 633–641.