Learning Affinity-Aware Upsampling for Deep Image Matting*

Yutong Dai[1], Hao Lu[2], Chunhua Shen[1]
[1] The University of Adelaide [2] Huazhong University of Science and Technology

Abstract

We show that learning affinity in upsampling provides an effective and efficient approach to exploit pairwise interactions in deep networks. Second-order features are commonly used in dense prediction to build adjacent relations with a learnable module after upsampling such as non-local blocks. Since upsampling is essential, learning affinity in upsampling can avoid additional propagation layers, offering the potential for building compact models. By looking at existing upsampling operators from a unified mathematical perspective, we generalize them into a second-order form and introduce Affinity-Aware Upsampling (A^2U) where upsampling kernels are generated using a light-weight low-rank bilinear model and are conditioned on second-order features. Our upsampling operator can also be extended to downsampling. We discuss alternative implementations of A^2U and verify their effectiveness on two detail-sensitive tasks: image reconstruction on a toy dataset; and a large-scale image matting task where affinity-based ideas constitute mainstream matting approaches. In particular, results on the Composition-1k matting dataset show that A^2U achieves a 14% relative improvement in the SAD metric against a strong baseline with negligible increase of parameters ($< 0.5\%$). Compared with the state-of-the-art matting network, we achieve 8% higher performance with only 40% model complexity.

1. Introduction

The similarity among positions, *a.k.a.* affinity, is commonly investigated in dense prediction tasks [22, 4, 8, 36, 20]. Compared with directly fitting ground truths using first-order features, modeling similarity among different positions can provide second-order information. There currently exist two solutions to learn affinity in deep networks: i) learning an affinity map before a non-deep backend and ii) defining a learnable affinity-based module to propagate information. We are interested in end-to-end affinity learning, because classic methods often build upon some assumptions, rendering weak generalization in general cases. Existing approaches typically propagate or model affinity after upsampling layers or before the last prediction layer. While affinity properties are modeled, they sometimes may not be effective for the downstream tasks. For instance,the work in [20] requires a feature encoding block besides the encoder-decoder architecture to learn affinity. The work in [4] needs more iterations to refine the feature maps according to their affinity at the last stage. As shown in Fig. 1, one plausible reason is that pairwise similarity is damaged during upsampling. In addition, it is inefficient to construct interactions between high-dimensional feature maps. We therefore pose the question: *Can we model affinity earlier in upsampling in an effective and efficient manner*?

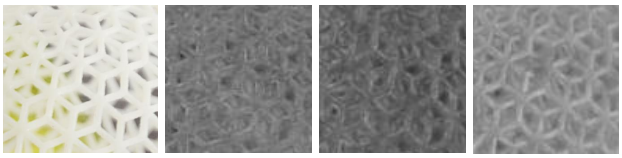

Figure 1 – Visualization of upsampled feature maps with various upsampling operators. From left to right, the input RGB image, feature maps after the last upsampling using nearest neighbor interpolation, bilinear upsampling, and our proposed affinity-aware upsampling, respectively. Our method produces better details with clear connectivity.

Many widely used upsampling operators interpolate values following a fixed rule at different positions. For instance, despite reference positions may change in bilinear upsampling, it always interpolates values based on relative spatial distances. Recently, the idea of learning to upsample emerges [24, 25, 35]. A learnable module is often built to generate upsampling kernels conditioned on feature maps to enable dynamic, feature-dependent upsampling behaviors. Two such representative operators include CARAFE [35] and IndexNet [25]. In our experiments, we find that CARAFE may not work well in low-level vision tasks where details need to be restored. IndexNet instead can recover details much better. We believe that one important reason is that IndexNet encodes, stores, and delivers spatial information prior to downsampling. But computation can be costly when the network goes deep. This mo-

*HL's contribution was made when he was with The University of Adelaide. YD and HL contributed equally. Correspondence should be addressed to CS (email: chunhua@icloud.com).

tivates us to pursue not only flexible but also light-weight designs of the upsampling operator.

In this paper, we propose to model affinity into upsampling and introduce a novel learnable upsampling operator, *i.e.*, affinity-aware upsampling (A^2U). As we show later in Section 4, *A^2U is a generalization of first-order upsampling operators*: in some conditions, the first-order formulation in [35] and [24] can be viewed as special cases of our second-order one. In addition, by implementing A^2U in a low-rank bilinear formulation, we can achieve efficient upsampling with few extra parameters.

We demonstrate the effectiveness of A^2U on two detail-sensitive tasks: an image reconstruction task on a toy dataset with controllable background and a large-scale image matting task with subtle foregrounds. Image matting is a desirable task to justify the usefulness of affinity, because affinity-based matting approaches constitute one of prominent matting paradigms in literatures. Top matting performance thus can suggest appropriate affinity modeling. In particular, we further discuss alternative design choices of A^2U and compare their similarities and differences. Compared with a strong image matting baseline on the Composition-1k matting dataset, A^2U exhibits a significant improvement ($\sim 14\%$) with negligible increase of parameters ($< 0.5\%$), proffering a light-weight image matting architecture with state-of-the-art performance.

2. Related work

Upsampling Operators in Deep Networks. Upsampling is often necessary in dense prediction to recover spatial resolution. The mostly used upsampling operators are bilinear interpolation and nearest neighbor interpolation. Since they are executed only based on spatial distances, they may be sub-optimal in detail-oriented tasks such as image matting where distance-based similarity can be violated. Compared with distance-based upsampling, max-unpooling is feature-dependent and has been shown to benefit detail-oriented tasks [24, 25], but it must match with max-pooling. In recent literatures, learning-based upsampling operators [32, 23, 35, 25] emerge. The Pixel Shuffle (P.S.) [32] upsamples feature maps by reshaping. The deconvolution (Deconv) [23], an inverse version of convolution, learns the upsampling kernel via back-propagation. Both P.S. and Deconv are data-independent during inference, because the kernel is fixed once learned. By contrast, CARAFE [35] and IndexNet [24] learn the upsampling kernel dynamically conditioned on the data. They both introduce additional modules to learn upsampling kernels. Since the upsampling kernel is directly related to the feature maps, these upsampling operators are considered first-order.

Following the learning-based upsampling paradigm, we also intend to learn dynamic upsampling operators but to condition on second-order features to enable affinity-informed upsampling. We show that, compared with first-order upsampling, affinity-informed upsampling not only achieves better performance but also introduces a light-weight learning paradigm.

Deep Image Matting. Affinity dominates the majority of classic image matting approaches [19, 3, 6, 9]. The main assumption in propagation-based matting is that, similar alpha values can be propagated from known positions to unknown positions, conditioned on affinity. This assumption, however, highly depends on the color distribution. Such methods can perform well on cases with clear color contrast but more often fail in cases where the color distribution assumption is violated. Recently, deep learning is found effective to address ill-posed image matting. Many deep matting methods arise [5, 36, 39, 33, 12, 24, 20, 2]. This field has experienced from a semi-deep stage [5, 36] to a fully-deep stage [39, 12, 24, 20, 2]. Here 'semi-deep' means that the matting part still relies on classic methods [19, 3] to function, while 'fully-deep' means that the entire network does not resort to any classic algorithms. Among fully-deep matting, DeepMatting [39] first applied the encoder-decoder architecture and reported improved results. Targeting this strong baseline, several deep matting methods were proposed. AlphaGAN matting [26] and IndexNet matting [24] explored adversarial learning and index generating module to improve matting performance, respectively. In particular, works in [12, 20, 2, 33] imitated classic sampling-based and propagation-based ideas into deep networks to ease the difficulty of learning. Therein, GCA matting [20] first designed an affinity-based module and demonstrated the effectiveness of affinity in fully-deep matting. It treats alpha propagation as an independent module and adds it to different layers to refine the feature map, layer by layer.

Different from the idea of 'generating then refining', we propose to directly incorporate the propagation-based idea into upsampling for deep image matting. It not only benefits alpha propagation but also shows the potential for light-weight module design.

3. A Mathematical View of Upsampling

The work in [25] unifies upsampling from an indexing perspective. Here we provide an alternative mathematical view. To simplify exposition, we discuss the upsampling of the one-channel feature map. Without loss of generality, the one-channel case can be easily extended to multi-channel upsampling, because most upsampling operators execute per-channel upsampling. Given a one-channel local feature map $\mathbf{Z} \in \mathbb{R}^{k \times k}$ used to generate an upsampled feature point, it can be vectorized to $\mathbf{z} \in \mathbb{R}^{k^2 \times 1}$. Similarly, the vectorization of an upsampling kernel $\mathbf{W} \in \mathbb{R}^{k \times k}$ can be denoted by $\mathbf{w} \in \mathbb{R}^{k^2 \times 1}$. If $g(\mathbf{w}, \mathbf{z})$ defines the output of

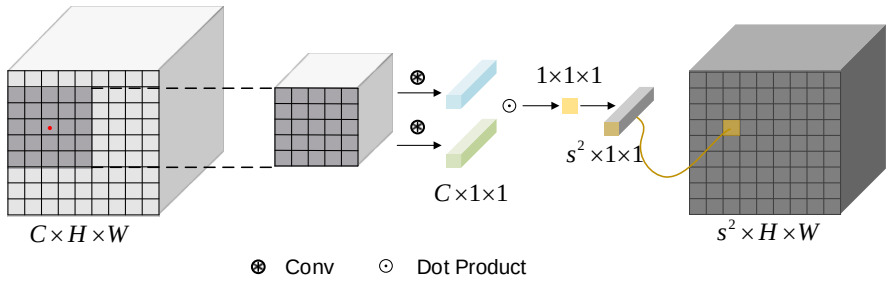

Figure 2 – Kernel generation of A^{2}U. Given a feature map of size $C \times H \times W$, an $s \times s$ upsampling kernel is generated at each spatial position conditioned on the feature map. The rank d is 1 here.

upsampling, most existing upsampling operations follow

$$g(\mathbf{w}, \mathbf{z}) = \mathbf{w}^T \mathbf{z}\,. \tag{1}$$

Note that $g(\mathbf{w}, \mathbf{z})$ indicates an upsampled point. In practice, multiple such points can be generated to form an upsampled feature map. $\mathbf{w}$ may be either shared or unshared among channels depending on the upsampling operator. Further, even the same $\mathbf{w}$ can be applied to different $\mathbf{z}$'s. According to how the upsampling kernel $\mathbf{w}$ is generated, we categorize the kernel into two types: the universal kernel and the customized kernel. The universal kernel is input-independent. One example is deconvolution [23]. The customized kernel, however, is input-dependent. Based on what input is used to generate the kernel, the customized kernel can be further divided into distance-based and feature-based. We elaborate as follows.

Distance-based Upsampling. Distance-based upsampling is implemented according to spatial distances, such as nearest neighbor and bilinear interpolation. The difference between them is the number of positions taken into account. Under the definition of Eq. (1), the upsampling kernel is a function of the relative distance between points.

Feature-based Upsampling. Feature-based upsampling is feature-dependent. They are developed in deep networks, including max-unpooling [1], CARAFE [35], and IndexNet [25]:

i) *Max-unpooling* interpolates values following the indices returned from max-pooling. In a 2×2 region of the feature layer after upsampling, only one position recorded in the indices has value, and other three are filled with 0. We can define $\mathbf{w}$ by a 1×1 vector $\mathbf{w} = [w]$, where $w \in \mathbb{R}^{1\times 1}$, and $\mathbf{z}$ is also the 1×1 point at the low-resolution layer.

ii) *CARAFE* learns an upsampling kernel $\mathbf{w} \in \mathbb{R}^{k^2\times 1}$ ($k = 5$ in [35]) via a kernel generation module given a decoder feature map ready to upsample. It also conforms to Eq. (1), where $\mathbf{z} \in \mathbb{R}^{k^2\times 1}$ is obtained from the low-resolution decoder feature map. The kernel size of $\mathbf{w}$ depends on the size of $\mathbf{z}$. In multi-channel cases, the same $\mathbf{w}$ is shared among channels.

iii) *IndexNet* also learns an upsampling kernel dynamically from features. The difference is that IndexNet learns from high-resolution encoder feature maps. Under the formulation of Eq. (1), the upsampling kernel follows a similar spirit like max-unpooling. But here $w \in [0, 1]$ instead of $\{0, 1\}$.

Hence, different operators correspond to different $\mathbf{w}$'s and $\mathbf{z}$'s, where $\mathbf{w}$ can be heuristically defined or dynamically generated. In particular, existing operators define/generate $\mathbf{w}$ according to distances or first-order features, while *second-order information remains unexplored in upsampling*.

4. Learning Affinity-Aware Upsampling

Here we explain how we exploit second-order information to formulate the affinity idea in upsampling using a bilinear model and how we apply a low-rank approximation to reduce computational complexity.

General Formulation of Upsampling. Given a feature map $\mathcal{M} \in \mathbb{R}^{C\times H\times W}$ to be upsampled, the goal is to generate an upsampled feature map $\mathcal{M}' \in \mathbb{R}^{C\times rH\times rW}$, where r is the upsampling ratio. For a position (i', j') in $\mathcal{M}'$, the corresponding source position (i, j) in $\mathcal{M}$ is derived by solving $i = \lfloor i'/r \rfloor$, $j = \lfloor j'/r \rfloor$. We aim to learn an upsampling kernel $\mathbf{w} \in \mathbb{R}^{k^2\times 1}$ for each position in $\mathcal{M}'$. By applying the kernel to a channel of the local feature map $\mathcal{X} \in \mathbb{R}^{C\times k\times k}$ centered at position l on $\mathcal{M}$, denoted by $\mathbf{X} \in \mathbb{R}^{1\times k\times k}$, the corresponding upsampled feature point $m'_{l'} \in \mathcal{M}'$ of the same channel at target position l' can be obtained by $m'_{l'} = \mathbf{w}^T\mathbf{x}$ according to Eq. (1), where $\mathbf{x} \in \mathbb{R}^{k^2\times 1}$ is the vectorization of $\mathbf{X}$.

General Meaning of Affinity. Affinity is often used to indicate pairwise similarity and is considered second-order features. An affinity map can be constructed in different ways such as using a Gaussian kernel. In self-attention, the affinity between the position l and the enumeration of all possible positions p at a feature map $\mathcal{M}$ is denoted by $\underset{\forall p}{softmax}\left(sim\left(\mathbf{m}_l, \mathbf{m}_p\right)\right)$, where $\mathbf{m}_l$ and $\mathbf{m}_p$ represent two vectors at position l and p, respectively, and

$sim(\mathbf{m}_l, \mathbf{m}_p)$ measures the similarity between $\mathbf{m}_l$ and $\mathbf{m}_p$ with the inner product ${\mathbf{m}_l}^T\mathbf{m}_p$.

Affinity-Aware Upsampling via Bilinear Modeling. Given a local feature map $\mathcal{X} \in \mathbb{R}^{C\times h_1\times w_1}$, $\mathcal{X}$ has an equivalent matrix form $\mathbf{X} \in \mathbb{R}^{C\times N}$, where $N = h_1 \times w_1$. We aim to learn an upsampling kernel conditioned on $\mathbf{X}$. Previous learning-based upsampling operators [35, 24, 25] generate the value of the upsampling kernel following a linear model by $w = \sum_{i=1}^{C}\sum_{j=1}^{N} a_{ij}x_{ij}$, where a_{ij} and x_{ij} are the weight and the feature at the channel i and position j of $\mathbf{X}$, respectively. Note that $w \in \mathbb{R}^{1\times 1}$. To encode second-order information, a natural generalization of the linear model above is bilinear modeling where another feature matrix $\mathbf{Y} \in \mathbb{R}^{C\times M}$ transformed from the feature map $\mathcal{Y} \in \mathbb{R}^{C\times h_2\times w_2}$ ($M = h_2 \times w_2$), is introduced to pair with $\mathcal{X}$ to model affinity. Given each $\mathbf{x}_i \in \mathbb{R}^{C\times 1}$ in $\mathbf{X}$, $\mathbf{y}_j \in \mathbb{R}^{C\times 1}$ in $\mathbf{Y}$, the bilinear weight a_{ij} of the vector pair, and the embedding weights q_k and t_k for each channel of $\mathbf{x}_i$ and $\mathbf{y}_j$, we propose to generate each value of the upsampling kernel from embedded pairwise similarity, *i.e.*,

$$\begin{aligned} w &= \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}\varphi(\mathbf{x}_i)^T\phi(\mathbf{y}_j) = \sum_{k=1}^{C}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}q_k x_{ik}t_k y_{jk} \\ &= \sum_{k=1}^{C}\sum_{i=1}^{N}\sum_{j=1}^{M} a'_{ijk}x_{ik}y_{jk} = \sum_{k=1}^{C} {\mathbf{x}_k}^T\mathbf{A}_k\mathbf{y}_k\,, \end{aligned} \tag{2}$$

where $\mathbf{x}_k \in \mathbb{R}^{N\times 1}$ and $\mathbf{y}_k \in \mathbb{R}^{M\times 1}$ are the k-th channel of $\mathbf{X}$ and $\mathbf{Y}$, respectively, $\mathbf{A}_k \in \mathbb{R}^{N\times M}$ is the affinity matrix for k-th channel, $a'_{ijk} = a_{ij}q_k t_k$, and φ and ϕ represent the embedding function.

Factorized Affinity-Aware Upsampling. Learning $\mathbf{A}_k$ can be expensive when M and N are large. Inspired by [14, 41], a low-rank bilinear method can be derived to reduce computational complexity of Eq. (2). Specifically, $\mathbf{A}_k$ can be rewritten by $\mathbf{A}_k = \mathbf{U}_k\mathbf{V}_k^T$, where $\mathbf{U}_k \in \mathbb{R}^{N\times d}$ and $\mathbf{V}_k \in \mathbb{R}^{M\times d}$. d represents the rank of $\mathbf{A}_k$ under the constraint of $d \le \min(N, M)$. Eq. (2) therefore can be rewritten by

$$\begin{aligned} w &= \sum_{k=1}^{C} {\mathbf{x}_k}^T\mathbf{U}_k{\mathbf{V}_k}^T\mathbf{y}_k = \sum_{k=1}^{C} \mathbb{1}^T({\mathbf{U}_k}^T\mathbf{x}_k \circ {\mathbf{V}_k}^T\mathbf{y}_k) \\ &= \mathbb{1}^T\sum_{k=1}^{C}({\mathbf{U}_k}^T\mathbf{x}_k \circ {\mathbf{V}_k}^T\mathbf{y}_k) \end{aligned}\,, \tag{3}$$

where $\mathbb{1} \in \mathbb{R}^d$ is a column vector of ones, and $\circ$ denotes the Hadamard product. Since we need to generate a $s \times s$ upsampling kernel, $\mathbb{1}$ in Eq. (3) can be replaced with $\mathbf{P} \in \mathbb{R}^{d\times s^2}$. Note that, Eq. (3) is applied to each position of a feature map, so the inner product here can be implemented by convolution. The full upsampling kernel therefore can be generated by

$$\begin{aligned} \mathbf{w} &= \mathbf{P}^T\sum_{k=1}^{C}({\mathbf{U}_k}^T\mathbf{x}_k \circ {\mathbf{V}_k}^T\mathbf{y}_k) \\ &= \mathbf{P}^T\overset{d}{\underset{r=1}{\texttt{cat}}}\Big(\sum_{k=1}^{C}({\mathbf{u}_{kr}}^T\mathbf{x}_k \circ {\mathbf{v}_{kr}}^T\mathbf{y}_k)\Big) \\ &= \texttt{conv}\Big(\mathcal{P}, \overset{d}{\underset{r=1}{\texttt{cat}}}\big(\texttt{gpconv}(\mathcal{U}_r, \mathcal{X}) \odot \texttt{gpconv}(\mathcal{V}_r, \mathcal{Y})\big)\Big) \end{aligned}\,, \tag{4}$$

where $\mathbf{u}_{kr} \in \mathbb{R}^{N\times 1}$, $\mathbf{v}_{kr} \in \mathbb{R}^{M\times 1}$. The convolution kernels $\mathcal{P} \in \mathbb{R}^{d\times s^2\times 1\times 1}$, $\mathcal{U} \in \mathbb{R}^{d\times C\times h_1\times w_1}$, and $\mathcal{V} \in \mathbb{R}^{d\times C\times h_2\times w_2}$ are reshaped tensor versions of $\mathbf{P}$, $\mathbf{U}$ and $\mathbf{V}$, respectively. $\texttt{conv}(\mathcal{K}, \mathcal{M})$ represents a convolution operation on the feature map $\mathcal{M}$ with the kernel $\mathcal{K}$; $\texttt{gpconv}(\mathcal{K}, \mathcal{M})$ defines a group convolution operation (C groups) with the same input. `cat` is the concatenate operator. This process is visualized in Fig. 2.

Alternative Implementations. Eq. (4) is a generic formulation. In practice, many design choices can be discussed in implementation:

i) The selection of $\mathcal{X}$ and $\mathcal{Y}$ can be either same or different. In this paper, we only discuss self-similarity, i.e., $\mathcal{X} = \mathcal{Y}$;

ii) The rank d can be chosen in the range $[1, \min(N, M)]$. For example, if $\mathcal{X}$ and $\mathcal{Y}$ are extracted in 5×5 regions, the range will be $[1, 25]$. In our experiments, we set $d = 1$ to explore the most simplified and light-weight case.

iii) $\mathcal{U}$ and $\mathcal{V}$ can be considered two encoding functions. They can be shared, partly-shared, or unshared among channels. We discuss two extreme cases in the experiments: 'channel-shared' ('*cs*') and 'channel-wise' ('*cw*').

iv) Eq. (4) adjusts the kernel size of $\mathbf{w}$ only using $\mathcal{P}$. Since the low-rank approximation has less parameters, fixed $\mathcal{P}$, $\mathcal{U}$, and $\mathcal{V}$ may not be sufficient to model all local variations. Inspired by CondConv [40], we attempt to generate $\mathcal{P}$ and $\mathcal{U}$, $\mathcal{V}$ dynamically conditioned on the input. We investigate three implementations: 1) *static*: none of them is input-dependent; 2) *hybrid*: only $\mathcal{P}$ is conditioned on input; and 3) *dynamic*: $\mathcal{P}$, $\mathcal{U}$, and $\mathcal{V}$ are all conditioned on input. The dynamic generation of $\mathcal{P}$, $\mathcal{U}$, or $\mathcal{V}$ is implemented using a global average pooling and a 1×1 convolution layer.

v) We implement stride-2 $\mathcal{U}$ and $\mathcal{V}$ in our experiments. They output features of size $C \times \frac{H}{2} \times \frac{W}{2}$. To generate an upsampling kernel of size $s^2 \times H \times W$, one can either use 4 sets of different weights for $\mathcal{U}$ and $\mathcal{V}$ or 4 sets of weights for $\mathcal{P}$ ($4\times s^2 \times \frac{H}{2} \times \frac{W}{2}$), followed by a shuffling operation ($s^2 \times H \times W$). We denote the former case as 'pointwise' ('*pw*'). Further, as pointed out in [14], nonlinearity, *e.g.*, `tanh` or `relu`, can be added after the encoding of $\mathcal{U}$ and $\mathcal{V}$. We verify a similar idea by adding normalization and nonlinearity in the experiments.

Method	MNIST				Fashion-MNIST			
	PSNR (↑)	SSIM (↑)	MSE (↓)	MAE (↓)	PSNR (↑)	SSIM (↑)	MSE (↓)	MAE (↓)
Conv$_{/2}$-Nearest	28.54	0.9874	0.0374	0.0148	25.58	0.9797	0.0527	0.0269
Conv$_{/2}$-Bilinear	26.12	0.9783	0.0495	0.0205	23.68	0.9675	0.0656	0.0343
Conv$_{/2}$-Deconv [23]	31.85	0.9942	0.0256	0.0089	27.42	0.9870	0.0426	0.0207
P.S. [32]	31.63	0.9939	0.0262	0.0099	27.33	0.9868	0.0431	0.0212
MaxPool-MaxUnpool	29.91	0.9916	0.0320	0.0133	28.31	0.9901	0.0385	0.0218
MaxPool-CARAFE [35]	28.72	0.9885	0.0367	0.0131	25.17	0.9773	0.0552	0.0266
MaxPool-IndexNet † [24]	45.51	0.9997	0.0053	0.0024	45.83	0.9998	0.0051	0.0033
MaxPool-A^{2}U (Ours)	47.63	0.9998	0.0042	0.0020	46.41	0.9999	0.0048	0.0031
MaxPool-IndexNet ‡ [24]	47.13	0.9997	0.0044	0.0020	44.35	0.9998	0.0061	0.0036

Table 1 – Reconstruction results on the MNIST dataset and the Fashion-MNIST dataset. † denotes holistic index network, ‡ represents depthwise index network. Both index networks here apply the setting of 'context+linear' for a fair comparison.

Extension to Downsampling. Following [25], our method can also be extended to downsampling. Downsampling is in pair with upsampling, so their kernels are generated from the same encoder feature. We use 'd' to indicate the use of paired downsampling in experiments. We share the same $\mathcal{U}$ and $\mathcal{V}$ in Eq. (4) in both downsampling and upsampling, but use different $\mathcal{P}$'s considering that they may have different kernel sizes. We denote the overall upsampling kernel by $\mathcal{W}_u \in \mathbb{R}^{{s_u}^2 \times H \times W}$ and the downsampling kernel by $\mathcal{W}_d \in \mathbb{R}^{{s_d}^2 \times H/r \times W/r}$, where r is the ratio of upsampling/downsampling. We set $s_d = r s_u$ in our experiments.

Relation to Other Works. A^{2}U learns to upsample guided by feature layers. This property shares similarity with some recent works.

i) *Joint Bilateral Upsampling (JBU)* [16]. JBU was proposed to facilitate efficient high-resolution image processing, which in detail is processing a low-resolution image first and then obtain the high-resolution result under the guidance of the corresponding high-resolution image. The upsampling weight is generated from a spatial distance and color distance on the guidance image. Our method applies the idea of guided upsampling in a more general way: inputs of the upsampling operation are feature layers, and the upsampling weights are learned from the inputs dynamically.

ii) *Guided Filter (GF)* [10]. GF upsampling was also investigated to generate high-resolution output O_h given the corresponding low-resolution one O_l and the high-resolution guidance image I_H. It models the generation of O_l as a linear model and calculates the model weights. The weights are then upsampled to be high-resolution before producing O_h by a linear transformation. Another learning-based GF (LGF) [37] further learns weights of the linear model from inputs, making the method adept to various tasks. Their nature of transforming low-resolution image processing operations to be high-resolution is different from ours, where the upsampling operations applied on feature layers are directly guided by the high-resolution maps.

iii) *Attention Networks* [34, 13]. Attention networks include a wide family of networks applying the attention mechanism. As discussed in [24, 35], attention networks exploit the relationships among different positions by feature scaling or selection. Our method instead is specially designed for the upsampling/downsampling stage rather than refining feature maps.

5. Image Reconstruction and Analysis

Here we conduct a pilot image reconstruction experiment on a toy dataset to show the effectiveness of A^{2}U. Inspired by [25], we build sets of reconstruction experiments on the MNIST dataset [18] and Fashion-MNIST dataset [38]. The motivation behind is to verify whether exploiting second-order information into upsampling benefits recovering spatial information.

The same network architecture, training strategies and evaluation metrics are used following [25]. Details are shown in the appendix. Since training patches are relatively small (32×32), upsampling kernel sizes for CARAFE and A^{2}U are both set to 1, and the encoding convolution kernels in IndexNet and A^{2}U are both set to 4. Other settings keep the default ones. We apply 'static-pw-cw' A^{2}U here because it is the same as Holistic IndexNet if letting convolution results of $\mathcal{U}$ to be all ones. We hence add a `sigmoid` function after $\mathcal{U}$ to generalize IndexNet. To avoid extra layers, we apply max-pooling to downsampling stages to obtain high-resolution layers when validating IndexNet and A^{2}U. Reconstruction results are presented in Table 1.

As shown in Table 1, upsampling operators informed by features (max-unpooling, CARAFE, IndexNet, and A^{2}U) outperform the operators guided by spatial distances (nearest, bilinear, and bicubic). Moreover, learning from high-resolution features matter for upsampling, among which, learning-based operators (IndexNet, A^{2}U) achieve the best results. Further, it is worth noting that, A^{2}U performs better than IndexNet with even fewer parameters. From these observations, we believe in upsampling: 1) high-resolution features are beneficial to extract spatial information, and 2) second-order features can help to recover more spatial details than first-order ones.

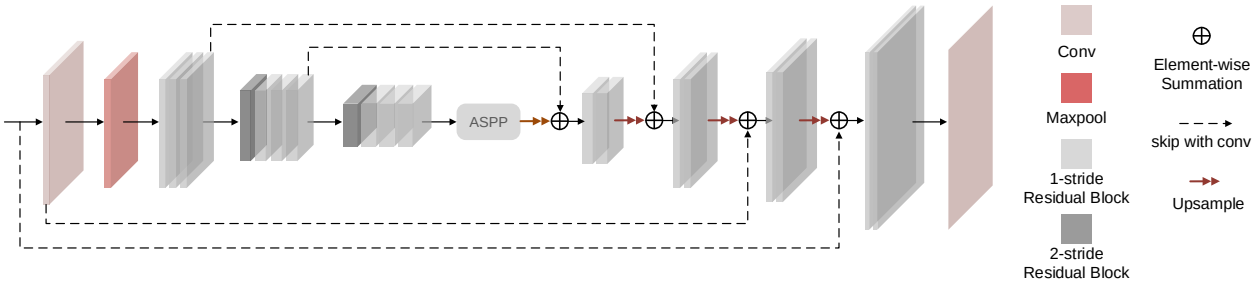

Figure 3 – Overview of our matting framework. The focus of this work is on the upsampling stages.

6. Experiments and Discussions

Here we evaluate A^{2}U on deep image matting. This task is suitable for assessing the quality of modeling pairwise relations.

Network Architecture. Similar to [20], our baseline network adopts the first 11 layers of the ResNet34 [11] as the encoder. The decoder consists of residual blocks and upsampling stages. The In-Place Activated BatchNorm [31] is applied to each layer except the last one to reduce GPU memory consumption during the training stage. As shown in Fig. 3, the overall network follows the UNet architecture [30] with 'skip' connection. To apply A^{2}U to upsampling, we replace the upsampling operations in the decoder with A^{2}U modules. Specifically, we learn upsampling kernels from the skipped features. If A^{2}U is used in both upsampling and downsampling stages, we change all 2-stride convolution layers in the encoder to be 1-stride and implement paired downsampling and upsampling operations, respectively, by learning upsampling/downsampling kernels from the modified 1-stride feature layer.

Datasets. We mainly conduct our experiments on the Adobe Image Matting dataset [39]. Its training set has 431 unique foreground objects and ground-truth alpha mattes. Instead of compositing each foreground with fixed 100 background images chosen from MS COCO [21], we randomly choose the background images in each iteration and generate the composition images on-the-fly. The test set, termed the Composition-1k, contains 50 unique foreground objects; each foreground is composited with 20 background images from the Pascal VOC dataset [7].

We also evaluate our method on the `alphamatting.com` benchmark [29]. This online benchmark has 8 unique testing images and 3 different trimaps for each image, providing 24 test cases.

Further, we report results on the recently proposed Distinctions-646 dataset [28]. It has 596 foreground objects in the training set and 50 foreground objects in the test set. We generate the training data and the test set following the same protocol as on the Adode Image Matting dataset.

Implementation Details. Our implementation is based on PyTorch [27]. Here we describe training details on the Adobe Image Matting dataset. The 4-channel input concatenates the RGB image and its trimap. We mainly follow the data augmentation of [20]. Two foreground objects are first chosen with a probability of 0.5 and are composited to generate a new foreground image and a new alpha matte. Next, they are resized to 640×640 with a probability of 0.25. Random affine transformations are then applied. Trimaps are randomly dilated from the ground truth alpha mattes with distances in the range between 1 and 29, followed by 512×512 random cropping. The background image is randomly chosen from the MS COCO dataset [21]. After imposing random jitters to the foreground object, the RGB image is finally generated by composition.

The backbone is pretrained on ImageNet [17]. Adam optimizer [15] is used. We use the same loss function as [39, 24], including alpha prediction loss and composition loss computed from the unknown regions indicated by trimaps. We update parameters for 30 epochs. Each epoch has a fixed number of 6000 iterations. A batch size of 16 is used and BN layers in the backbone are fixed. The learning rate is initialized to 0.01 and reduced by $\times 10$ at the 20-th epoch and the 26-th epoch, respectively. The training strategies on the Distinction646 dataset are the same except that we update the parameters for only 25 epochs. We evaluate our results using Sum of Absolute Differences (SAD), Mean Squared Error (MSE), Gradient (Grad), and Connectivity (Conn) [29].

Upsample	SAD	MSE	Grad	Conn	# Params
Nearest	37.51	0.0096	19.07	35.72	8.05M
Bilinear	37.31	0.0103	21.38	35.39	8.05M
CARAFE	41.01	0.0118	21.39	39.01	+0.26M
IndexNet	34.28	0.0081	**15.94**	31.91	+12.26M
A^{2}U (static-pw-cw)	36.36	0.0099	21.03	34.40	+0.10M
A^{2}U (static-cw)	35.92	0.0098	20.06	33.68	+26K
A^{2}U (hybrid-cw)	34.76	0.0088	16.39	32.29	+44K
A^{2}U (hybrid-cs)	36.43	0.0098	21.24	34.11	+19K
A^{2}U (dynamic-cw)	36.66	0.0094	18.60	34.62	+0.20M
A^{2}U (dynamic-cs)	35.86	0.0095	17.13	33.71	+20K
A^{2}U (dynamic-cs-d)	33.13	**0.0078**	17.90	30.22	+38K
A^{2}U (dynamic-cs-d)†	**32.15**	0.0082	16.39	**29.25**	+38K

Table 2 – Results of different upsampling operators on the Composition-1k test set with the same baseline model. † denotes additional normalization and nonlinearity after the encoding layers of $\mathcal{U}$ and $\mathcal{V}$. The best performance is in boldface.

Method	k_{up}	SAD	MSE	Grad	Conn
A^{2}U (hybrid-cw)	1	37.74	0.0104	22.07	35.91
A^{2}U (hybrid-cw)	3	34.76	0.0088	16.39	32.29
A^{2}U (hybrid-cw)	5	35.99	0.0093	17.96	33.90
A^{2}U (dynamic-cs)	1	36.06	0.0098	17.25	33.95
A^{2}U (dynamic-cs)	3	35.86	0.0095	17.13	33.71
A^{2}U (dynamic-cs)	5	37.40	0.0096	18.28	35.50

Table 3 – Ablation study of upsampling kernel size on the Composition-1k test set.

6.1. The Adobe Image Matting Dataset

Ablation Study on Alternative Implementations. Here we verify different implementations of A^{2}U on the Composition-1k test set and compare them with existing upsampling operators. Quantitative results are shown in Table 2. All the models are implemented by the same architecture but with different upsampling operators. The 'nearest' and 'bilinear' are our direct baselines. They achieve close performance with the same model capacity. For CARAFE, we use the default setting as in [35], *i.e.*, $k_{up} = 5$ and $k_{encoder} = 3$. We observe CARAFE has a negative effect on the performance. The idea behind CARAFE is to reassemble contextual information, which is not the focus of matting where subtle details matter. However, it is interesting that CARAFE can still be useful for matting when it follows a light-weight MobileNetV2 backbone [25]. One possible explanation is that a better backbone (ResNet34) suppresses the advantages of context reassembling. We report results of IndexNet with the best-performance setting ('depthwise+context+nonlinear') in [24, 25]. The upsampling indices are learned from the skipped feature layers. IndexNet achieves a notable improvement, especially on the Grad metric. However, IndexNet significantly increases the number of parameters.

We further investigate 6 different implementations of A^{2}U and another version with paired downsampling and upsampling. According to the results, the 'static' setting can only improve the SAD and Conn metrics. The position-wise and position-shared settings report comparable results, so we fix the position-shared setting in the following 'hybrid' and 'dynamic' experiments. We verify both channel-wise and channel-shared settings for 'hybrid' and 'dynamic' models. The 'hybrid' achieves higher performance with channel-wise design, while the 'dynamic' performs better with channel-shared design. All 'hybrid' and 'dynamic' models show improvements against baselines on all metrics, except the MSE and Grad metrics for the channel-shared 'hybrid' model. The last implementation, where channel-shared 'dynamic' downsampling is paired with upsampling, achieves the best performance (at least 14% relative improvements against the baseline) with negligible increase of parameters ($< 0.5\%$).

Hence, while the dedicated design of upsampling operators matters, paired downsampling and upsampling seems more important, at least for image matting.

Method	Norm	SAD	MSE	Grad	Conn
A^{2}U (hybrid-cw)	softmax	35.93	0.0092	17.13	33.87
A^{2}U (hybrid-cw)	sigmoid+softmax	34.76	0.0088	16.39	32.29
A^{2}U (dynamic-cs)	softmax	36.40	0.0100	17.67	34.33
A^{2}U (dynamic-cs)	sigmoid+softmax	35.86	0.0095	17.13	33.71

Table 4 – Ablation study of normalization on the Composition-1k test set.

Method	SAD	MSE	Grad	Conn	# Params
Closed-Form [19]	168.1	0.091	126.9	167.9	-
KNN Matting [3]	175.4	0.103	124.1	176.4	-
Deep Matting [39]	50.4	0.014	31.0	50.8	$>$ 130.55M
IndexNet Matting [24]	45.8	0.013	25.9	43.7	8.15M
AdaMatting [2]	41.7	0.010	16.8	-	-
Context-Aware [12]	35.8	**0.0082**	17.3	33.2	107.5M
GCA Matting [20]	35.28	0.0091	16.9	32.5	25.27M
A^{2}U (hybrid-cw)	34.76	0.0088	**16.39**	32.29	8.09M
A^{2}U (dynamic-cs)	35.86	0.0095	17.13	33.71	8.07M
A^{2}U (dynamic-cs-d)	**32.15**	**0.0082**	**16.39**	**29.25**	8.09M

Table 5 – Benchmark results on the Composition-1k test set. The best performance is in boldface.

Ablation Study on Upsampling Kernel Size. Here we investigate the performance of our models with different upsampling kernel sizes. The encoding kernel size (the kernel size of $\mathcal{U}$ or $\mathcal{V}$) is set to $k_{en} = 5$ in all matting experiments unless stated. Under this setting, results in Table 3 show that $k_{up} = 3$ performs the best. It is interesting to observe that larger upsampling kernel does not imply better performance. We believe that this is related to the encoding kernel size and the way how we generate $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{P}$. We use $k_{up} = 3$ as our default setting.

Ablation Study on Normalization. In both [35] and [25], different normalization strategies are verified, and experiments show that normalization significantly affects the results. We thus justify the normalization choices in our A^{2}U module here. We conduct the experiments on the channel-wise 'hybrid' model and the channel-shared 'dynamic' model. Two normalization choices are considered: 'softmax' and 'sigmoid+softmax'. It is clear that the latter normalization works better (Table 4). It may boil down to the nonlinearity introduced by the sigmoid function.

Comparison with State of the Art. Here we compare our models against other state-of-the-art methods on the Composition-1k test set. Results are shown in Table 5. We observe that our models outperform other methods on all the evaluation metrics with the minimum model capacity. Compared with the state-of-the-art method [20], our best model achieves 8% higher performance with only 40% model complexity. Our model is also memory-efficient, being able to infer high-resolution images on a single 1080Ti GPU without downsampling on the Composition-1k test set. Some qualitative results are shown in Fig. 4. Our results show improved detail delineation such as the net structure and the filament.

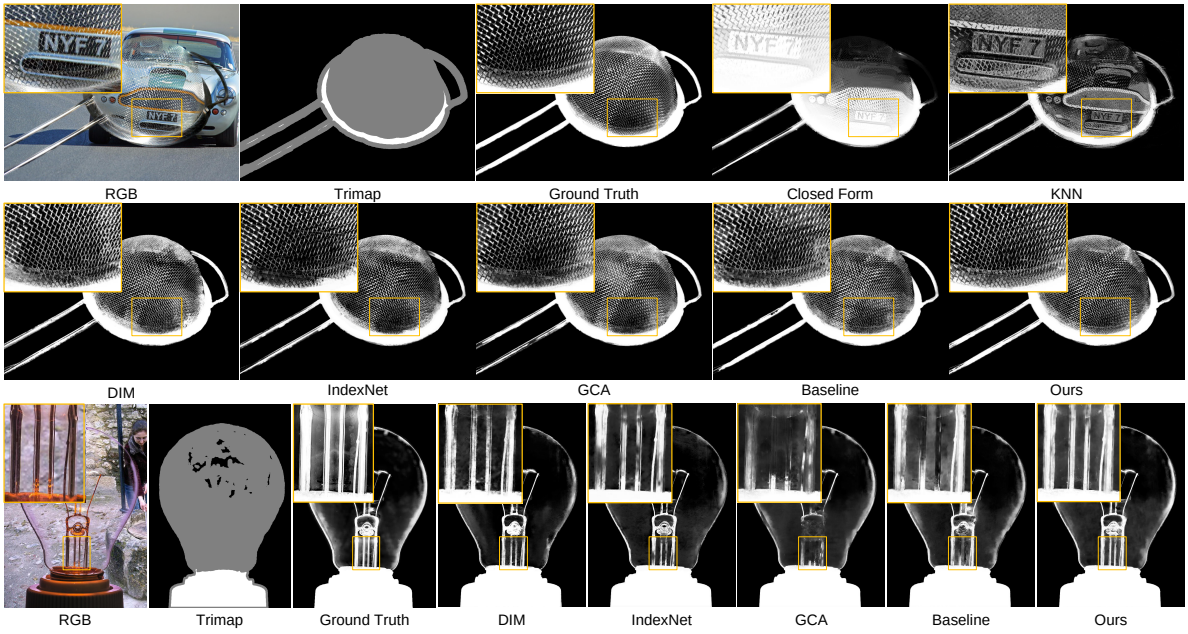

Figure 4 – Qualitative results on the Composition-1k test set. The methods in comparison include Closed-Form Matting [19], KNN Matting [3], Deep Image Matting (DIM) [39], IndexNet Matting [24], GCA Matting [20], our baseline, and our method.

Gradient Error	Average Rank				Troll			Doll			Donkey			Elephant			Plant			Pineapple			Plastic bag			Net		
	Overall	S	L	U	S	L	U	S	L	U	S	L	U	S	L	U	S	L	U	S	L	U	S	L	U	S	L	U
Ours	**6.3**	5.6	**3.3**	10.1	0.2	0.2	0.2	0.1	**0.1**	0.2	**0.1**	**0.2**	**0.2**	0.2	0.2	0.4	1.1	**1.3**	1.9	0.6	0.7	1.7	0.6	**0.6**	0.6	**0.3**	**0.3**	0.4
AdaMatting [2]	7.8	**4.5**	5.6	13.3	0.2	0.2	0.2	**0.1**	0.1	0.4	0.2	0.2	0.2	**0.1**	**0.1**	0.3	**1.1**	1.4	2.3	**0.4**	**0.6**	**0.9**	0.9	1	0.9	0.3	0.4	**0.4**
GCA Matting [20]	8	8.4	6.6	**9.1**	**0.1**	**0.1**	0.2	0.1	0.1	0.3	0.2	0.2	0.2	0.2	0.2	**0.3**	1.3	1.6	1.9	0.7	0.8	1.4	**0.6**	0.7	**0.6**	0.4	0.4	0.4
Context-aware Matting [12]	9.1	10.8	9.8	6.8	0.2	0.2	**0.2**	0.1	0.2	**0.2**	0.2	0.2	0.2	0.2	0.4	0.4	1.4	1.5	**1.8**	0.8	1.3	1	1.1	1.1	0.9	0.4	0.4	0.4

Table 6 – Gradient errors on the `alphamatting.com` test set. The top-4 methods are shown. The lowest errors are in boldface.

6.2. The `alphamatting.com` Benchmark

Here we report results on the `alphamatting.com` benchmark [29]. We train our model with all the data in the Adobe Matting dataset and test it on the benchmark. As shown in Table 6, our method ranks the first w.r.t. the gradient error among all published methods. We also achieve comparable overall ranking compared with AdaMatting [2] under the SAD and MSE metrics, suggesting our method is one of the top performing methods on this benchmark.

6.3. The Distinction-646 Dataset

We also evaluate our method on the recent Distinction-646 test set. In Table 7, we report results of the three models performing the best on the Composition-1k dataset and also compare with other benchmarking results provided by [28]. We have two observations: 1) our models show improved performance against the baseline, which further confirms the effectiveness of our A^{2}U; 2) Our models outperform other reported benchmarking results by large margins, setting a new state of the art on this dataset.

Method	SAD	MSE	Grad	Conn
Closed-Form* [19]	105.73	0.023	91.76	114.55
KNN Matting* [3]	116.68	0.025	103.15	121.45
Deep Matting* [39]	47.56	0.009	43.29	55.90
Baseline-Nearest	25.03	0.0106	13.85	24.41
A^{2}U (hybrid-cw)	24.08	0.0104	13.53	23.59
A^{2}U (dynamic-cs)	24.55	0.0107	14.51	23.89
A^{2}U (dynamic-cs-d)	**23.20**	**0.0102**	**12.39**	**22.20**

Table 7 – Benchmark results on the Distinctions-646 test set. The best performance is in boldface. * denotes results cited from [28].

7. Conclusion

Considering that affinity is widely exploited in dense prediction, we explore the feasibility to model such second-order information into upsampling for building compact models. We implement this idea with a low-rank bilinear formulation, based on a generalized mathematical view of upsampling. We show that, with negligible parameters increase, our method A^{2}U can achieve better performance on both image reconstruction and image matting tasks. We also investigate different design choices of A^{2}U. Results on three image matting benchmarks all show that A^{2}U invites a significant relative improvement and also state-of-the-art results. In particular, compared with the best performing image matting network, our model achieves 8% higher performance on the Composition-1k test set, with only 40% model capacity. For future work, we plan to extend A^{2}U to other dense prediction tasks.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(12):2481–2495, 2017. 3

[2] Shaofan Cai, Xiaoshuai Zhang, Haoqiang Fan, Haibin Huang, Jiangyu Liu, Jiaming Liu, Jiaying Liu, Jue Wang, and Jian Sun. Disentangled image matting. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 8819–8828, 2019. 2, 7, 8

[3] Qifeng Chen, Dingzeyu Li, and Chi-Keung Tang. Knn matting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(9):2175–2188, 2013. 2, 7, 8

[4] Xinjing Cheng, Peng Wang, and Ruigang Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *Proc. Eur. Conf. Comp. Vis.*, pages 103–119, 2018. 1

[5] Donghyeon Cho, Yu-Wing Tai, and Inso Kweon. Natural image matting using deep convolutional neural networks. In *Proc. Eur. Conf. Comp. Vis.*, pages 626–643. Springer, 2016. 2

[6] Yung-Yu Chuang, Brian Curless, David H Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, volume 2, pages II–II. IEEE, 2001. 2

[7] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision.*, 88(2):303–338, 2010. 6

[8] Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 642–651, 2019. 1

[9] Kaiming He, Christoph Rhemann, Carsten Rother, Xiaoou Tang, and Jian Sun. A global sampling method for alpha matting. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2049–2056. IEEE, 2011. 2

[10] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *Proc. Eur. Conf. Comp. Vis.*, pages 1–14, 2010. 5

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 770–778, 2016. 6

[12] Qiqi Hou and Feng Liu. Context-aware image matting for simultaneous foreground and alpha estimation. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 4130–4139, 2019. 2, 7, 8

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 7132–7141, 2018. 5

[14] Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Hadamard product for low-rank bilinear pooling. In *Proc. Int. Conf. Learn. Representations*, 2016. 4

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[16] Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Trans. Graphics*, 26(3):96–es, 2007. 5

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 6

[18] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 5

[19] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):228–242, 2007. 2, 7, 8

[20] Yaoyi Li and Hongtao Lu. Natural image matting via guided contextual attention. In *Proc. AAAI Conf. Artificial Intell.*, volume 34, pages 11450–11457, 2020. 1, 2, 6, 7, 8

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. Eur. Conf. Comp. Vis.*, pages 740–755. Springer, 2014. 6

[22] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, and Jan Kautz. Learning affinity via spatial propagation networks. In *Adv. Neural Inform. Process. Syst.*, pages 1520–1530, 2017. 1

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3431–3440, 2015. 2, 3, 5

[24] Hao Lu, Yutong Dai, Chunhua Shen, and Songcen Xu. Indices matter: Learning to index for deep image matting. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 3266–3275, 2019. 1, 2, 4, 5, 6, 7, 8

[25] Hao Lu, Yutong Dai, Chunhua Shen, and Songcen Xu. Index networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020. 1, 2, 3, 4, 5, 7

[26] Sebastian Lutz, Konstantinos Amplianitis, and Aljosa Smolic. Alphagan: Generative adversarial networks for natural image matting. *British Machine Vis. Conf*, 2018. 2

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Adv. Neural Inform. Process. Syst.*, pages 8026–8037, 2019. 6

[28] Yu Qiao, Yuhao Liu, Xin Yang, Dongsheng Zhou, Mingliang Xu, Qiang Zhang, and Xiaopeng Wei. Attention-guided hierarchical structure aggregation for image matting. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 13676–13685, 2020. 6, 8

[29] Christoph Rhemann, Carsten Rother, Jue Wang, Margrit Gelautz, Pushmeet Kohli, and Pamela Rott. A perceptually motivated online benchmark for image matting. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1826–1833. IEEE, 2009. 6, 8

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. Int. Conf. Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. 6

[31] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 6

[32] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1874–1883, 2016. 2, 5

[33] Jingwei Tang, Yagiz Aksoy, Cengiz Oztireli, Markus Gross, and Tunc Ozan Aydin. Learning-based sampling for natural image matting. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3055–3063, 2019. 2

[34] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3156–3164, 2017. 5

[35] Jiaqi Wang, Kai Chen, Rui Xu, Ziwei Liu, Chen Change Loy, and Dahua Lin. Carafe: Content-aware reassembly of features. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 3007–3016, 2019. 1, 2, 3, 4, 5, 7

[36] Yu Wang, Yi Niu, Peiyong Duan, Jianwei Lin, and Yuanjie Zheng. Deep propagation based image matting. In *Int. Joint Conf. Artificial Intell.*, volume 3, pages 999–1006, 2018. 1, 2

[37] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided filter. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1838–1847, 2018. 5

[38] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 5

[39] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2970–2979, 2017. 2, 6, 7, 8

[40] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *Adv. Neural Inform. Process. Syst.*, pages 1307–1318, 2019. 4

[41] Chaojian Yu, Xinyi Zhao, Qi Zheng, Peng Zhang, and Xinge You. Hierarchical bilinear pooling for fine-grained visual recognition. In *Proc. Eur. Conf. Comp. Vis.*, pages 574–589, 2018. 4