THE UNIVERSITY *of* ADELAIDE

Doctoral Thesis

Efficient Deep Networks for Image Matting

Submitted by:

Yutong Dai

Supervised by:

Prof. Chunhua Shen

Dr. Qi Wu

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Faculty of Engineering, Computer and Mathematical Sciences

School of Computer Science

April 2022

Dedicated to my parents, my husband,
for their unconditional love and endless support

Contents

List of Figures

List of Tables

Declaration

Signed:

Date: 10/12/2021

THE UNIVERSITY OF ADELAIDE

Abstract

Efficient Deep Networks for Image Matting

by Yutong DAI

Image matting is a fundamental technology serving downstream image editing tasks such as composition and harmonization. Given an image, its goal is to predict an accurate alpha matte with minimum manual efforts. Since matting applications are usually on PC or mobile devices, a high standard for efficient computation and storage is set. Thus, lightweight and efficient models are in demand. However, it is non-trivial to balance the computation and the performance. We therefore investigate efficient model designs for image matting. We first look into the common encoder-decoder architecture with a lightweight backbone and explore the skipped information and downsampling-upsampling operations, from which we notice the importance of indices kept in the encoder and recovered in the decoder. Based on the observations, we design data-dependant downsampling and upsampling operators conditioned on features from the encoder, which learn to index and show significant improvement against the baseline model while promising a lightweight structure. Then, considering affinity is widely used in both traditional and deep matting methods, we propose upsampling operators conditioned on the second-order affinity information, termed affinity-aware upsampling. Instead of modeling affinity in an additional module, we include it in the unavoidable upsampling stages for a compact architecture. Through implementing the operator by a low-rank bilinear model, we achieve significantly better results with only neglectable parameter increases. Further, we explore the robustness of matting algorithms and raise a more generalizable method. It includes designing a new framework assembling multilevel context information and studying strong data augmentation strategies targeting matting. This method shows significantly higher robustness to various benchmarks, real-world images, and coarse-to-fine trimap precision compared with other methods while using less computation. Besides studying trimap-based image matting, we extend our lightweight matting architecture to portrait matting. Targeting portrait images, we propose a multi-task parameter sharing framework, where trimap generation and matting are treated as parallel tasks and help optimize each other. Compared with the conventional cascaded architecture, this design not only reduces the model capacity to a large margin but also presents more precise predictions.

Acknowledgements

First of all, I would like to thank my Ph.D. supervisor, Professor Chunhua Shen, for all his help throughout my study. I was a beginner when I first came to Adelaide and I have been so fortunate to work with him for these three years. He started me along this path and motivated me to be a good researcher. His deep insights, passion, and hardworking are invaluable treasures for me.

I would like to thank Dr. Hao Lu in HUST, who used to be a postdoc in our lab. Hao shared lots of invaluable research experience with me, including how to think, how to solve a problem, and how to present. He is also a good friend, cheering me a lot throughout my research life.

I would like to thank my mentor, Dr. Brian Price during my internship. He motivated me to look into more interesting and meaningful topics and taught me to think deeper. He is a professional collaborator and also a nice friend. I enjoyed a wonderful time working with him. I would also thank Dr. He Zhang for the inspiring discussions and help during the internship. Working with him was truly my pleasure.

I would like to thank all my lab mates, who are the most brilliant and diligent people I have met. The great lab environment would not be possible without them. I enjoyed the time working in our lab.

I would like to especially thank my friends, including but not limited to Dzung, Daqi, Dandan, Violetta, Shinfang, CK, Tong, Frances, Dong, Peng Chen, Jinan, Rafa, Andrew, Mahsa, Cuong, Yang, Xinyu, Yifan, Haokui, Wei Liu, Qingsen, Shengqin, Uni, Haiming. We had fun together and shared a lot of good memories. I am so happy to meet these nice friends, who helped me not feel lonely while staying alone in Adelaide.

I would also thank my parents. They taught me to be an independent and upright person. They respect all my decisions and always stand behind me to make me feel warm and supported.

Finally, I would like to thank my husband Yunlei. He is so autonomic and well-regulated and kept motivating and supporting me. It was he who encouraged me to apply for the Ph.D. position. I shared every happiness and sadness with him and he could always understand me. Although we had been long-distance for three years and had not seen each other for a long time because of the pandemic, I never felt we were far from each other. I appreciate him being in my life.

Publications

This thesis contains the following works that have been published:

- *Indices Matter: Learning to Index for Deep Image Matting.*
  Hao Lu, **Yutong Dai**, Chunhua Shen, Songceng Xu. IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

- *Index Networks.*
  Hao Lu, **Yutong Dai**, Chunhua Shen, Songceng Xu. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2020.

- *Learning Affinity-Aware Upsampling for Deep Image Matting.*
  Yutong Dai, Hao Lu, Chunhua Shen. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

- *Boosting Robustness of Image Matting with Context Assembling and Strong Data Augmentation.*
  Yutong Dai, Brian Price, He Zhang, Chunhua Shen. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022.

- *Towards Light-Weight Deep Portrait Matting via Parameter Sharing.*
  Yutong Dai, Hao Lu, Chunhua Shen. Computer Graphics Forum (CGF), 2021.

Chapter 1

Introduction

Object selection and composition are important operations in image editing [1]. For instance, objects in an image can be cut out and pasted onto other images in visually realistic manners after several steps of adjustment. A precise selection promises satisfying composition results. At the core of object selection, segmentation is applied, which aims to provide pixel-level category prediction of objects. An accurate pre-segmentation can speed up the editing process by satisfying the criteria: 1) distinct representation of the image; 2) soft transition of the boundaries; 3) efficient computation. Semantic segmentation meets the first criteria. As for the soft boundaries, soft segmentation is in demand, which cares about the detailed transition of boundaries such as hairs and nets. It promotes the development of image matting. Demands for efficient computation further accelerate studies on efficient matting algorithms.

1.1 Problem Formulations

Image matting is a long-standing task in computer vision. As introduced above, it is a cornerstone for many image/video editing applications, which is to extract the accurate foreground objects in an image with minimum human efforts, as shown in Fig. 1.1a and 1.1b. It can be mathematically expressed by solving:

$$I_i = \alpha_i F_i + (1 - \alpha_i)\, B_i\,, \tag{1.1}$$

where I_i, the color of pixel i, is assumed to be a linear combination of the corresponding foreground color F_i and background color B_i, and α_i is its foreground opacity. Equation (1.1) is the matting equation, where 7 unknown variables need to be solved given only 3 known equations for a RGB image, posing matting as an ill-posed problem.

In the literature, significant efforts have been made in solving the matting equation using versatile mathematical tools [57, 96, 19, 11, 56]. These conventional methods are founded on manually defined constraints. For instance, in the closed-form matting [57], the color in a small window region is assumed to be unchanged to satisfy its mathematical model. These assumptions made on low-level color cues, however, are easily violated in real scenes.

With the expansion of deep learning, deep matting methods emerge as new and strong baselines [113, 74, 89]. They apply convolutional models to extract high-level semantic cues and long-range dependencies, advancing the matting results to a new height. For example, traditional methods are easy to fail with complicated backgrounds or transparent foreground objects because the color cues are unreliable there, but deep matting methods deal with those cases successfully. It makes deep matting overwhelming in both academia and industry.

Since image matting is widely applied to image editing applications such as image post-processing and photography on PC and mobile phones, matting algorithms have to be computation and storage efficient to be approachable by these devices. Designing efficient matting models is therefore necessary. It requires to balance the computation and the accuracy.

Normally, deep matting methods need prior information inputs, such as a trimap, to specify absolute foreground, absolute background, and unknown regions. Their predictions, therefore, are focused on the unknown regions under the guidance of the information from known regions. Different from this setting, there are also some works, termed prior-free methods, trying to remove prior input [123, 12, 89], which can be achieved by focusing on specific objects, such as salient objects, humans, and portraits. They make the strong assumption that the foreground objects are strictly under the defined limitations and use the assumption as implicit prior information.

Specifically, human/portrait matting is one of the mainstream topics in deep matting. With a single RGB image input, deep human/portrait matting methods predict alpha mattes for human objects without prior inputs. It is achievable because of the specific foreground category - humans, which builds the connection between semantic segmentation and matting. [89] is the first work applying deep learning to portrait matting, where a convolutional segmentation model is connected with a differentiable closed-form [57] matting module, supporting end-to-end training.

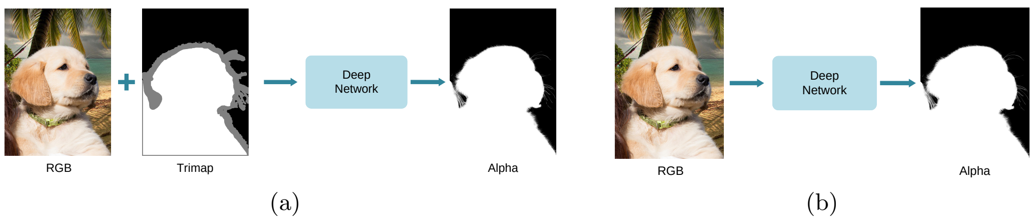


FIGURE 1.1: Examples of image matting. (a) Trimap-based image matting. (b) Prior-free image matting.

1.2 Motivation

Though deep image matting (DIM) [113] advances image matting to a large margin, it is not efficient enough in both training and inference, which leaves difficulties for real applications and thus makes more efficient methods in demand. However, it is challenging to design an efficient matting model because of the attention to tiny details. DIM proposes to formulate matting as a regression problem, and directly applies a segmentation model - SegNet [3] by changing the channel number of the output layer to be 1. Due to the gap between segmentation and matting, the model directly borrowed from segmentation is inefficient for matting. There are many specific properties of matting remaining to be explored, such as affinity, color cues, gradient, etc. Introducing these properties to matting models in a proper manner shows the potential to build more efficient models. It is one of the motivations for Chapter 4.

It is also possible to apply other deep learning skills to further boost effectiveness and efficiency. For example, modeling long-range dependencies may facilitate better recognition of foreground objects; combining low-level and high-level cues in a better way could benefit the prediction of detailed structures; using lightweight architectures should build the basis for efficient models. We take these into consideration in Chapter 3, Chapter 5 and Chapter 6.

Meanwhile, since only synthetic training data are available in current datasets and the quantity is limited, it inevitably raises gaps between the training set and real-world images. The gaps we care about more here are feature-level gaps, such as the mismatch in resolutions, illuminations, noise levels, etc. A model trained with pure synthetic data may fail on a simple real-world image with noises. On another side, even though given trimaps, different trimap precisions result in very different matting results. Thus, we need to enhance the robustness of the matting models to images from different domains and also to coarse-to-fine trimaps. It motivates us to study the robustness of matting algorithms in Chapter 5.

1.3 Contribution and Thesis Outline

Motivated by the above problems, we investigate efficient matting methods by exploring new architectures and data processing driven by this task. The main contributions of this thesis are:

1. The first lightweight matting method – IndexNet. It is motivated by the observation of the importance of indices learned in the network. This observation makes the state-of-the-art performance of a lightweight model possible. We also show effectiveness of the IndexNet on other dense prediction tasks.
2. A fast and accurate matting method – A^2U. It models the affinity information to the matting network with neglectable parameter increases, which promises an efficient architecture showing better performance against the baseline model and other top-performing methods.
3. A robust matting framework – RMat. It includes multilevel context assembling and strong data augmentation targeting matting. RMat not only achieves competitive results on the deep matting datasets but also promises higher robustness to various benchmarks, real-world images, and coarse-to-fine trimaps using less computation.
4. A lightweight prior-free deep portrait matting method – PSPM. By using a parameter-sharing mechanism, PSPM shows better results using much less computation, compared with the conventional cascaded architecture.

The outline of this thesis is organized as follows:

Chapter 2 reviews the background of image matting and analyses the progress and limitations of current methods as well as datasets. It also introduces the dynamic networks, a basic idea behind the following chapters.

Chapter 3 studies a lightweight matting network (IndexNet) with MobileNetv2 [86] backbone. At the core, an IndexNet module is investigated for recovering details in the encoder-decoder architecture. Meanwhile, we show that IndexNet can also benefit other dense prediction tasks (semantic segmentation, depth estimation, denoising, image reconstruction) with better boundary predictions. This chapter is based primarily on [72, 71].

Chapter 4 models affinity into the upsampling operation (A^2U). The motivation behind this is that affinity is a vital cue for alpha predictions in both traditional and deep matting methods, and modeling this property into upsampling stage is an efficient manner to encourage the relationship among different positions in the deep network. We build this upsampling operator using a low-rank bilinear model, which promises the implementation with neglectable parameter increases. This chapter is based primarily on [22].

Chapter 5 demonstrates to boost the robustness of image matting algorithms with a new matting framework (RMat). It is motivated by the domain gaps between synthetic training data and real-world images, as well as the unstable trimap precision. RMat achieves state-of-the-art results on several benchmarks (including both fitting and not fitting the training sets). It also presents more robust predictions on the images from various domains and coarse-to-fine trimaps. This chapter is based primarily on [24].

Chapter 6 describes a prior-free deep portrait matting method (PSPM). Based on our lightweight IndexNet model, multitask learning strategy is introduced to this method to predict the trimap and the alpha matte simultaneously. By studying parameter-sharing mechanisms, lightweight models are promised with even better matting results. This chapter is based primarily on [23].

Chapter 2

Literature Review

In this chapter, we first generally review the development of image matting, including traditional methods and recent deep matting methods. Then, we introduce image matting datasets proposed in the literature, which facilitate training the deep models. Finally, since dynamic networks are used in the following chapters to design efficient models, a brief review of dynamic networks is also included.

2.1 Image Matting

2.1.1 Traditional Image Matting

Before the application of deep learning, image matting is commonly solved as an optimization problem followed by specific assumptions. Those methods can be categorized into sampling-based matting and propagation-based matting. To ease the difficulty of alpha prediction, correlations among nearby pixels are utilized. Given a prior input, since the absolute foregrounds and absolute backgrounds are known, correlations among unknown pixels and known pixels help infer the alpha values.

Sampling-based matting. Sampling-based methods [88, 19, 35, 30] sample neighbor known foreground and background pixels, and get the foreground and background values (F_i, B_i) of pixel I_i by their correlations, then alpha value α_i can be calculated from Equation 1.1. Difficulties lie in how to find the neighbour pixels, how many pixels are needed, and how to get foreground and background values for an unknown pixel from the samples. Aiming at these difficulties, many solutions were proposed. For example, Bayesian matting [19] uses foreground and background samples to build Gaussian distributions and then models the matting problem into a Bayesian framework solved by the MAP technique; [88] designs a more comprehensive and representative sampling strategies to ensure information of true foreground and background is fully used.

Propagation-based matting. Propagation-based methods [57, 11, 2] instead rely on the affinities among different positions in the color space to propagate alpha values from known regions to unknown pixels. For instance, closed-form matting [57] assumes the local smoothness (e.g. in 3×3 windows) on the foreground and background and then models matting into a quadratic cost function solved by a linear system solver. Though pixels in nearby regions are more likely to be similar, the local color-line model is easily violated so users need to carefully label the prior input. Thus, nonlocal matting [56, 38, 11] appear. For example, KNN matting [11] applies nonlocal principles by using K nearest neighbors to contribute a more stable prediction given even sparse prior information.

Prior inputs for matting. As aforementioned, prior inputs are required by matting algorithms. Therein, trimap is a widely adopted one. Besides the trimap inputs, scribbles and click points are also supported by some methods [56, 11, 57, 39]. Compared with trimaps, scribbles and click points save users' time and costs, but they provide much less prior information. [56] discusses what is good user input and proposes to reduce human efforts by applying nonlocal principles. Since these traditional algorithms are built on low-level color cues, how efficiently to label user inputs is quite important for a high-quality alpha matte.

2.1.2 Deep Image Matting

Since traditional matting methods rely heavily on assumptions on low-level color cues, which are easily violated on complicated images, deep matting methods [17, 89, 113] emerge with the development of deep learning.

Deep image matting has experienced 'semi-deep' and 'fully-deep' stages. The 'semi-deep' means a combination of deep learning modules and traditional modules, such as [17, 89]. In [17], closed-form matting [57] and KNN matting [11] are applied to generate initial alpha mattes and a convolutional network is used to combine the alpha mattes and produce final results. [89] connects a fully-convolutional network (FCN) with a differentiable closed-form module, where the FCN is responsible for trimap segmentation and the closed-form module takes charge of matting. Though applying deep learning models, these 'semi-deep' methods are still limited by the traditional matting modules. To solve this problem, 'fully-deep' methods are investigated. As the name suggests, 'fully-deep' represents fully deep matting models without relying on traditional modules to predict alpha mattes. Deep image matting (DIM) [113] is the first fully deep method of formulating matting as a regression problem. This formulation dominates the following works on matting. The dataset proposed in [113] also promotes researchers working on this topic. As initial attempts to investigate deep matting, the above-mentioned methods are restricted by either conventional color model assumptions [17, 89] or inefficient computation [113].

Promoted by the deep matting dataset proposed in [113], several deep matting methods have been investigated since then. Our IndexNet [72] first uses a light-weight MobileNetv2 [44] backbone achieving even better results than DIM via exploring indices-guided upsampling; AdaMatting [4] uses multi-task training to implement trimap optimization and alpha prediction simultaneously, which enhances the robustness to various trimap precision; Context-Aware matting [43] adds an extra context encoder branch modeling context information to recognize the foreground object more precisely; GCA [60] proposes a guided contextual attention module to refine the features under the guidance of low-level feature information. More recently, our proposed A^{2}U [22] models the affinity information into the upsampling stage to learn affinity information compactly; SIM [97] learns a semantic trimap at the first stage of the network to distinguish different semantic parts; TIMI [68] learns to mine the input information with a 3-branch encoder; FBA [32] first proposes to predict foreground, background and alpha values simultaneously in the deep model and contributes some useful training strategies to promote matte precision; Our RMat [24] moves a step further to enhance the robustness of matting methods regarding trimap instability as well as real-world applications.

In summary, current methods commonly focus on more precise prediction and efficient models. They usually follow the UNet style [82] with an encoder-decoder architecture. Feature skip [72, 60], downsampling&upsampling stages [72, 22], contextual information [43, 24], affinity property [60, 22] and some training strategies [32, 60] are recent main focuses. Since trimap inputs are given, they pay more attention on unknown regions by enforcing loss functions specifically on those regions.

2.1.3 Prior-Free Deep Image Matting

The above deep matting methods all require trimap input. There are also some works [104, 15] study scribble/click prior input. No matter which forms of prior information is taken, it is still a necessary input. On another hand, there have been a few prior-free methods [89, 12, 123] proposed taking advantage of deep learning. They usually omit the prior input by learning semantic/salient information in the deep networks.

Some of them [89, 12, 7], focus on specific objects such as human or transparent objects. Human-targeted methods [89, 12] focus on human/portrait images, where semantic information on human bodies is learned in the network to replace the trimap input. This setting is practical because modalities of humans are limited, but existing methods combine segmentation and matting in naive ways, where cascaded architectures are applied, resulting in redundancy in the model designs. To reduce the redundancy, we propose PSPM [23]. On another side, the transparent-targeted method [7] uses the physical properties of transparent objects to waive the prior inputs. Different from the methods targeting specific objects, some works [123, 80] even target general objects. These methods [123, 80] are more like salient object detection, where obvious foreground objects are detected and cut out. They do not set limitations on the categories of the foreground objects, but have implicit requirements on the saliency of objects.

2.2 Image Matting Datasets

The first deep matting dataset proposed in [113], named Adobe Image Matting dataset, facilitates investigations on this topic. It is worth noting that, labeling alpha matting ground truths is non-trivial. It usually needs blue/green screen [93] technology to label precise values, which is time-consuming and costing, so it is difficult to build a large-scale high-quality matting dataset. The training set of the Adobe Image Matting dataset has 431 unique foreground objects and ground-truth alpha mattes. The number of foreground images is far from enough for training effective deep models, so each foreground is usually composited with hundreds of background images to generate tens of thousands of training images. In [113], the training set is generated by compositing each foreground image with fixed 100 background images selected from the MS COCO [65] dataset, contributing 43100 training images in total. We follow this setting in IndexNet and PSPM, but use different generation strategies in A^{2}U and RMat, as introduced in corresponding chapters. The test set termed Composition-1k, contains 50 unique foreground objects; each foreground is composited with 20 background images chosen from the Pascal VOC dataset [29]. Composition-1k is currently the most recognized benchmark for deep image matting. Besides the Adobe Image Matting dataset, some similar datasets are proposed in [80, 97]. Their training sets and test sets are built in the same way as [113]. The uniqueness of [97] is that data samples are categorized into tens of semantic types. All the above-mentioned datasets are synthetic ones. Targeting real-world images, a real-world benchmark [59] is presented recently, which consists of 500 test images. Another benchmark for evaluation is `alphamatting.com` [81]. It only has 27 images for training and 8 images for testing, but it is still a widely accepted benchmark for the fairness of comparison.

The first portrait matting dataset [89] (Portrait-2k) is currently the only publicly available portrait matting dataset with high-quality real-world RGB images. 1700 images and another 300 images compose its training and test set, respectively. Ground-truth alpha mattes in this dataset are generated by closed-form matting [57] and KNN matting [11] combined with manual modifications and selection. Apart from that, there are also some human matting datasets [12, 63] introduced, but some of them are not publicly available. Similar to deep matting datasets, human matting datasets use foreground images to generate synthetic data via composition.

2.3 Dynamic Networks

Dynamic networks are often implemented with adaptive modules to extend the modeling capabilities of CNNs. These networks share the following characteristics. The output is *dynamic*, conditioned on the input feature map. Since dynamic networks are learnable modules, they are *generic* in the sense that they can be used as building blocks in many network architectures. They are also *flexible* to allow modifications according to target tasks. Some representative dynamic networks are reviewed below.

Spatial Transformer Networks (STNs) [48]. STN allows explicit manipulation of spatial transformation within the network. It achieves this by regressing transformation parameters θ with a side-branch network. A spatially-transformed output is then produced by a sampler parameterized by θ. This results in a holistic transformation of the feature map. The dynamic nature of STN is reflected by the fact that, given different inputs, the inferred θ is different, allowing to learn some forms of invariance to translation, scale, rotation, etc.

Dynamic Filter Networks (DFNs) [50]. DFN implements a filter generating network to dynamically generate kernel filter parameters. Compared to conventional filter parameters that stay fixed during inference, filter parameters in DFN are dynamic and sample-specific.

Deformable Convolutional Networks (DCNs) [21]. DCNs introduce deformable transformation into convolution. The key idea is to predict offsets for convolutional kernels. With offsets, convolution can be executed on irregular sampling grids, enabling adaptive manipulation of the receptive field.

Attention Networks [77]. Attention networks are a broad family of networks that use attention mechanisms. The mechanisms introduce multiplicative interactions between the inferred attention map and the feature map. In computer vision, attention mechanisms are usually referred to spatial attention [99], channel attention [46] or both [106].

These network modules are widely applied in CNNs to force the network focusing on specific regions and therefore to refine feature maps. Essentially, attention is about feature selection.

In contrast to above dynamic networks, Chapter 3 and Chapter 4 focus on upsampling, rather than manipulating filters or refining features. They use dynamic networks to learn indices/upsampling kernels conditioned on feature maps, which share similarities with attention networks from the view of generating indices/kernels. However, a core difference is our focus on upsampling instead of refining feature maps. Detailed analyses are presented in corresponding chapters.

Vision Transformers [28]. Recently, transformers have been successfully applied to computer vision tasks. The standard transformer consists of embedding layers, transformer blocks, and task-specific heads. There are either pure transformer networks [28, 18] or hybrid networks where transformer blocks are combined with convolutional blocks [102, 25, 112, 5] to get benefits from both architectures. Founded on the self-attention mechanism, transformers learn long-range dependencies dynamically, which delivers significant improvements on various tasks. In Chapter 5, we apply the transformer to matting using a hybrid architecture. For matting, there are mainly two difficulties to overcome. The first is maintaining details, and the second is ensuring efficiency.

Title of Paper	Indices Matter: Learning to Index for Deep Image Matting
Publication Status	[✓ Published] ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	ICCV, 2019, PP. 3266-3275

Principal Author

Name of Principal Author (Candidate)	Yutong Dai
Contribution to the Paper	Experiment design and implementation, writing parts of the paper
Overall percentage (%)	50%
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.
Signature	Date 09/12/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

xiii. the candidate's stated contribution to the publication is accurate (as detailed above);

xiv. permission is granted for the candidate in include the publication in the thesis; and

xv. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Hao Lu
Contribution to the Paper	Experiment design and implementation, writing parts of the paper
Signature	Date 09/12./2021

Name of Co-Author	Chunhua Shen
Contribution to the Paper	Discussion and writing revision
Signature	09/12/2021

Name of Co-Author	Songcen Xu		
Contribution to the Paper	Discussion and writing revision		
Signature		Date	14/7/2020

Title of Paper	Index Network
Publication Status	[✓ Published] ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	TPAMI , Volume 44 , Issue : 1, Jan. 1 2022

Principal Author

Name of Principal Author (Candidate)	Yutong Dai		
Contribution to the Paper	Experiment design and implementation. writing parts of the paper		
Overall percentage (%)	80%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	09/12/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

xvi. the candidate's stated contribution to the publication is accurate (as detailed above);

xvii. permission is granted for the candidate in include the publication in the thesis; and

xviii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Hao Lu		
Contribution to the Paper	Experiment design and implementation, writing parts of the paper		
Signature		Date	09/12/2021

Name of Co-Author	Chunhua Shen	
Contribution to the Paper	Discussion and writing revision	
Signature		09/12/2021

Name of Co-Author	Songcen Xu		
Contribution to the Paper	Discussion and writing revision		
Signature		Date	14/7/2020

Chapter 3

Learning to Index for Deep Image Matting

In this chapter, we show that existing upsampling operators can be unified using the notion of the index function. This notion is inspired by an observation in the decoding process of deep image matting where indices-guided unpooling can often recover boundary details considerably better than other upsampling operators such as bilinear interpolation. By viewing the indices as a function of the feature map, we introduce the concept of 'learning to index', and present a novel index-guided encoder-decoder framework where indices are learned adaptively from data and are used to guide downsampling and upsampling stages, without extra training supervision. At the core of this framework is a new learnable module, termed Index Network (IndexNet), which dynamically generates indices conditioned on the feature map. IndexNet can be used as a plug-in applicable to almost all convolutional networks that have coupled downsampling and upsampling stages, enabling the networks to dynamically capture variations of local patterns. In particular, we instantiate, investigate five families of IndexNet, highlight their superiority in delivering spatial information over other upsampling operators with experiments on synthetic data, then demonstrate their effectiveness on deep image matting and further extend experiments to three other dense prediction tasks, including image denoising, semantic segmentation, and monocular depth estimation.

3.1 Introduction

Upsampling is an essential stage for dense prediction tasks using deep convolutional neural networks (CNNs). The frequently used upsampling operators include transposed convolution [120, 69], unpooling [3], *periodic shuffling* [90] (a.k.a. depth-to-space), and

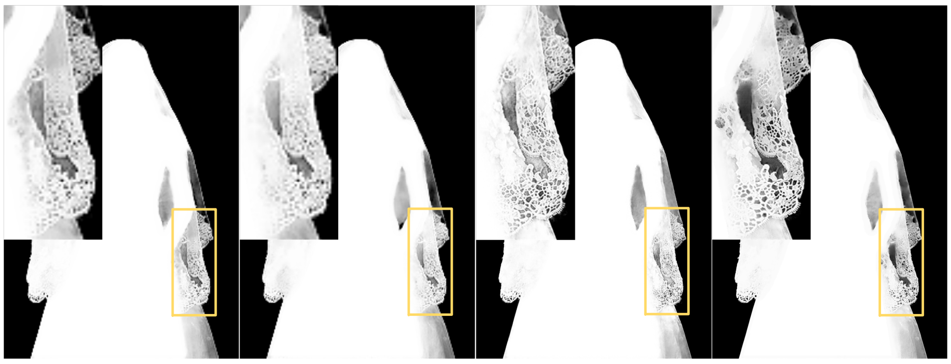


FIGURE 3.1: Alpha mattes of different models for the task of image matting. From left to right, Deeplabv3+ [9], RefineNet [62], Deep Matting [113] and IndexNet (Ours). Bilinear upsampling tends to fail to recover subtle details, while unpooling and our learned upsampling operator can produce much clear mattes with good local contrast.

naive interpolation [62, 9] followed by convolution. These operators, however, are not general-purpose designs and often exhibit different behaviors in different tasks.

The widely-adopted upsampling operator in semantic segmentation and depth estimation is bilinear interpolation, while unpooling is less popular. A reason might be that the feature map generated by max unpooling is sparse, while the bilinearly interpolated feature map has dense and consistent representations for local regions (compared to the feature map before interpolation). This is particularly true for semantic segmentation and depth estimation where pixels in a region often share the same class label or have similar depth. However, we observe that bilinear interpolation can perform significantly worse than unpooling in boundary-sensitive tasks such as image matting. A fact is that the leading deep image matting model [113] largely borrows the design from the SegNet method [3], where unpooling was first introduced. When adapting other state-of-the-art segmentation models, such as DeepLabv3+ [9] and RefineNet [62], to this task, we observe that they tend to fail to recover boundary details (Fig. 3.1). A plausible explanation is that, compared to the bilinearly upsampled feature map, unpooling uses max-pooling indices to guide upsampling. Since boundaries in the shallow layers usually have the maximum responses, indices extracted from these responses record the boundary locations. The feature map projected by the indices thus shows improved boundary delineation.

We thus believe that different upsampling operators may exhibit different characteristics, and we expect a specific behavior of the upsampling operator when dealing with specific image content for a particular vision task. A question of interest is: *Can we design a generic operator to upsample feature maps that better predict boundaries and regions simultaneously?* A key observation of this work is that unpooling, bilinear interpolation or other *upsampling operators are some forms of index functions.* For example, the nearest neighbor interpolation of a point is equivalent to allocating indices of one to its neighbor and then map the value of the point. In this sense, indices are models [54], therefore indices can be modeled and learned.

In this work, *we model indices as a function of the local feature map and learn index functions to implement upsampling within deep CNNs.* In particular, we present a novel index-guided encoder-decoder framework, which naturally generalizes models like SegNet. Instead of using max-pooling and unpooling, we introduce indexed pooling and indexed upsampling where downsampling and upsampling are guided by learned indices. The indices are generated *dynamically* conditioned on the feature map and are learned using a fully convolutional network, termed IndexNet, without extra supervision needed. IndexNet is a highly flexible module. It can be applied to almost all convolutional networks that have coupled downsampling and upsampling stages. Compared to the fixed max function or bilinear interpolation, learned index functions show potentials for simultaneous boundary and region delineation.

IndexNet is a high-level concept and represents a broad family of networks modeling the so-called index function. In this work, we instantiate and investigate five families of IndexNet. Different designs correspond to different assumptions.

We compare the behavior of IndexNet with existing upsampling operators and demonstrate its superiority in delivering spatial information. We show that IndexNet can be incorporated into many CNNs to benefit a number of visual tasks: i) *image matting*: our MobileNetv2-based [86] model with IndexNet exhibits at least 16.1% improvement against the VGG-16-based Deep Image Matting baseline [113] on the Composition-1k matting dataset; by visualizing learned indices, the indices automatically learn to capture the boundaries and textural patterns; ii) *image denoising*: a modified DnCNN model with IndexNet can achieve performance comparable to the baseline DnCNN [122] that has no downsampling stage on the BSD68 and Set12 datasets [84], thus reducing the computational cost and memory consumption significantly; iii) *semantic segmentation*: consistently improved performance is observed when SegNet [3] is equipped with IndexNet on the SUN RGB-D dataset [95]; and iv) *monocular depth estimation*: IndexNet also improves the performance of a recent light-weight FastDepth model on the NYUDv2 dataset [91], with negligible extra computation cost.

We make the following main contributions.

- We present a unified perspective of existing upsampling operators with the notion of the index function;
- We introduce Index Networks—a novel family of networks that can be included into standard CNNs to provide dynamic, adaptive downsampling and upsampling

capabilities; to the best of our knowledge, IndexNet is one of the first attempts towards the design of generic upsampling operators;

- We instantiate, and investigate five designs of IndexNet and demonstrate their effectiveness on deep image matting as well as extensions on three other vision tasks, including image denoising, semantic segmentation, and depth estimation.

3.2 Background

Upsampling in Deep Networks. Compared with other components in the design of deep networks, downsampling and upsampling of feature maps are relatively less studied. Since learning a CNN without sacrificing the spatial resolution is computationally expensive and memory intensive, and suffers from limited receptive fields, downsampling operators are common choices, such as strided convolution and max/average pooling. To recover the resolution, upsampling is thus an essential stage for almost all dense prediction tasks. This poses a fundamental question: *What is the principal approach to recover the resolution of a downsampled feature map (decoding).* A few upsampling operators are proposed. The *deconvolution* operator, a.k.a. transposed convolution, was initially used in [120] to visualize convolutional activations and introduced to semantic segmentation [69], but this operator sometimes can be harmful due to its behavior in producing checkerboard artifacts [78]. To avoid this, a suggestion is the "resize+convolution" paradigm, which has currently become the standard configuration in state-of-the-art semantic segmentation models [9, 62]. Apart from these, *perforate* [79] and *unpooling* [3] generate sparse indices to guide upsampling. The indices are able to capture and keep boundary information, but one issue is that the two operators can induce much sparsity after upsampling. Convolutional layers with large filter sizes must follow for densification. In addition, *periodic shuffling* ($\mathcal{PS}$) was introduced in [90] as a fast and memory-efficient upsampling operator for image super-resolution. $\mathcal{PS}$ recovers resolution by rearranging the feature map of size $H \times W \times Cr^2$ to $rH \times rW \times C$. It is also used in some segmentation models [116].

Our work is primarily inspired by the unpooling operator [3]. We remark that, it is important to extract spatial information before its loss during downsampling, and more importantly, to use stored information during upsampling. Unpooling shows a simple and effective use case, while we believe that there is much room to improve. Here we show that unpooling is a special form of index function, and we can learn an index function beyond unpooling.

We notice that concurrent work of [100] also pursues the idea of data-dependent upsampling and proposes an universal upsampling operator termed CARAFE. Although the idea is similar, IndexNet is different from CARAFE in several aspects. First, CARAFE does not associate upsampling with the notion of the index function. Second, the kernels used in CARAFE are generated conditioned on decoder features, while IndexNet builds upon encoder features, so the generated indices can also be used to guide downsampling. Third, CARAFE can be viewed as one of our investigated index networks—holistic index networks, but with different upsampling kernels and normalization strategies.

Relationship to Dynamic Networks. Akin to dynamic networks introduced in Chapter 2, the dynamics in IndexNet also has a physical definition—indices. Such a definition also closely relates to attention networks. Later we show that the downsampling and upsampling operators used with IndexNet can, to some extent, be viewed as attentional operators. Indeed, max-pooling indices are a form of hard attention. It is worth noting that, despite that IndexNet in its current implementation may closely relate to attention, it focuses on upsampling rather than refining feature maps. IndexNet also shares the other characteristics mentioned introduced in Chapter 2. It is implemented in a convolutional side-branch network, is trained without extra supervision and is generic and flexible. We demonstrate its effectiveness on four dense prediction tasks and five variants of IndexNet.

3.3 An Indexing Perspective of Upsampling

With the argument that upsampling operators are index functions, here we offer a unified indexing perspective of upsampling operators. The unpooling operator is straightforward. We can define its index function in a $k \times k$ local region as an indicator function

$$I_{max}(x) = \mathbb{1}(x = \max(\boldsymbol{X}))\,, x \in \boldsymbol{X}\,, \tag{3.1}$$

where $\boldsymbol{X} \in \mathbb{R}^{k \times k}$. $\mathbb{1}(\cdot)$ is the indicator function with output being a binary matrix. Similarly, if one extracts indices from average pooling, its index function takes the form

$$I_{avg}(x) = \mathbb{1}(x \in \boldsymbol{X})\,. \tag{3.2}$$

If further using $I_{avg}(x)$ during upsampling, it is equivalent to the nearest neighbor interpolation. As for the bilinear interpolation and deconvolution operators, their index functions have an identical form

$$I_{bilinear/dconv}(x) = \boldsymbol{W} \otimes \mathbb{1}(x \in \boldsymbol{X})\,, \tag{3.3}$$

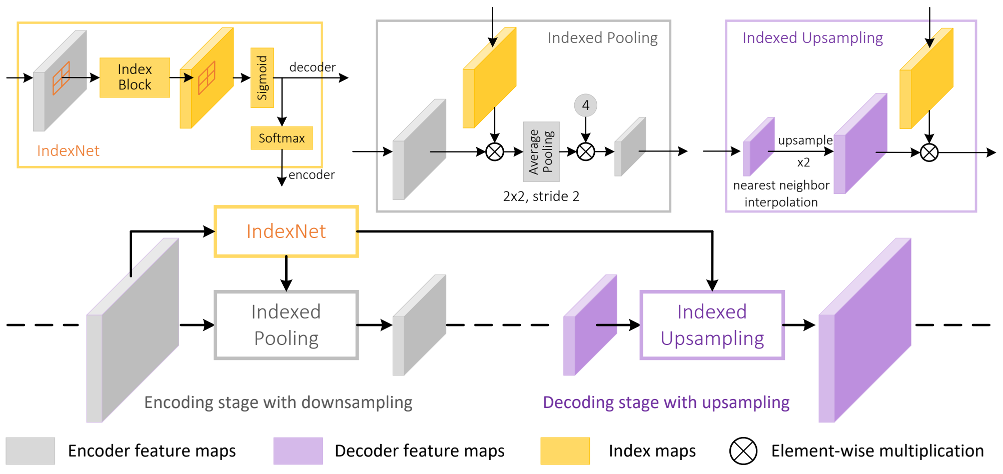


FIGURE 3.2: The index-guided encoder-decoder framework. The proposed IndexNet dynamically predicts indices for individual local regions, conditioned on the input local feature map itself. The predicted indices are further used to guide the downsampling in the encoding stage and the upsampling in the corresponding decoding stage.

where $\boldsymbol{W}$ is the weight/filter of the same size as $\boldsymbol{X}$, and $\otimes$ denotes the element-wise multiplication. The difference is that, $\boldsymbol{W}$ is learned in deconvolution but predefined in bilinear interpolation. Indeed bilinear interpolation has been shown to be a special case of deconvolution [69]. Note that, in this case, the index function generates soft indices. The sense of index for the $\mathcal{PS}$ operator [90] is also clear, because the rearrangement of the feature map is an indexing process. Considering $\mathcal{PS}$ a tensor $\mathcal{Z}$ of size $1 \times 1 \times r^2$ to a matrix $\boldsymbol{Z}$ of size $r \times r$, the index function can be expressed by the one-hot encoding

$$I_{ps}^{l}(x) = \mathbb{1}(x = \mathcal{Z}_l)\,, l = 1, ..., r^2\,, \tag{3.4}$$

such that $\boldsymbol{Z}_{m,n} = \mathcal{Z}[I_{ps}^{l}(x)]$, where $m = 1, ..., r$, $n = 1, ..., r$, and $l = (r-1)\cdot m + n$. $\mathcal{Z}_l$ denotes the l-th element of $\mathcal{Z}$. Similar notation applies to $\boldsymbol{Z}_{m,n}$.

Since upsampling operators can be unified by the notion of the index function, it is plausible to ask whether one can learn an index function to dynamically capture local spatial patterns.

3.4 Learning to Index, to Pool, and to Upsample

Before introducing the designs of IndexNet, we first present the general idea about how learned indices may be used in downsampling and upsampling with a new index-guided encoder-decoder framework. Our framework is a generalization of SegNet, as illustrated in Fig. 3.2. For ease of exposition, let us assume the downsampling and upsampling rates to be 2, and the pooling operator to use a kernel size of 2×2. The IndexNet module dynamically generates indices given the feature map. The proposed indexed pooling and indexed upsampling operators further receive generated indices to guide downsampling and upsampling, respectively. In practice, multiple such modules can be combined and used analogous to the max pooling layers for every downsampling and upsampling stage.

IndexNet models the index as a function of the feature map $\mathcal{X} \in \mathbb{R}^{H\times W\times C}$. Given $\mathcal{X}$, it generates two index maps for downsampling and upsampling, respectively. An important concept for the index is that an index can either be represented in a natural order, e.g., 1, 2, 3, ..., or be represented in a logical form, i.e., 0, 1, 0, ..., meaning that an index map can be used as a mask. This is exactly how we use the index map in downsampling/upsampling. The predicted index shares the same definition of the index in computer science, except that we generate *soft* indices for smooth optimization, i.e., for any index i, $i \in [0, 1]$.

IndexNet consists of a predefined index block and two index normalization layers. An index block can simply be a heuristically defined function, e.g., a max function, or more generally, a parameterized function such as neural network. In this work, we use a fully convolutional network to be the index block. More details are presented in Sections 3.4.2 and 3.4.3. Note that the index maps sent to the encoder and decoder are normalized differently. The decoder index map only goes through a *sigmoid* function such that for any predicted index $i \in (0, 1)$. As for the encoder index map, indices of each local region L are further normalized by a *softmax* function such that $\sum_{i\in L} i = 1$. The second normalization guarantees the magnitude consistency of the feature map after downsampling.

Indexed Pooling ($\mathcal{IP}$) performs downsampling using generated indices. Given a local region $E \in \mathbb{R}^{k\times k}$, $\mathcal{IP}$ calculates a weighted sum of activations and corresponding indices over E as $\mathcal{IP}(E) = \sum_{x\in E} I(x)x$, where $I(x)$ is the index of x. It is easy to see that max pooling and average pooling are special cases of $\mathcal{IP}$. In practice, this operator can be easily implemented with an element-wise multiplication between the feature map and the index map, an average pooling layer, and a multiplication of a constant used to compensate the effect of averaging, as instantiated in Fig. 3.2. The current implementation is equivalent to 2×2 stride-2 convolution with dynamic kernels, but is more efficient than explicit on-the-fly kernel generation.

Indexed Upsampling ($\mathcal{IU}$) is the inverse operator of $\mathcal{IP}$. $\mathcal{IU}$ upsamples $d \in \mathbb{R}^{1\times 1}$ that spatially corresponds to E taking the same indices into account. Let $I \in \mathbb{R}^{k\times k}$ be the local index map formed by $I(x)$s. $\mathcal{IU}$ upsamples d as $\mathcal{IU}(d) = I \otimes D$, where $\otimes$ denotes the element-wise multiplication, and D is of the same size as I and is upsampled from d with the nearest neighbor interpolation. $\mathcal{IU}$ also relates to deconvolution, but an important difference between $\mathcal{IU}$ and deconvolution is that, deconvolution applies a fixed kernel to

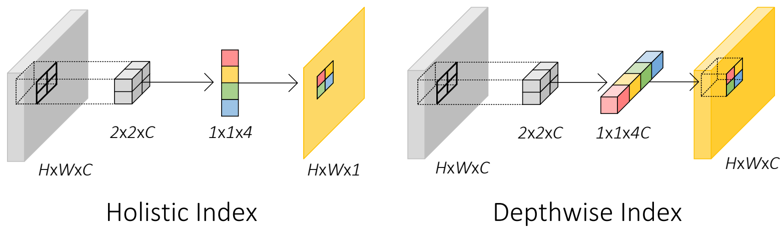


FIGURE 3.3: Conceptual differences between holistic and depthwise index.

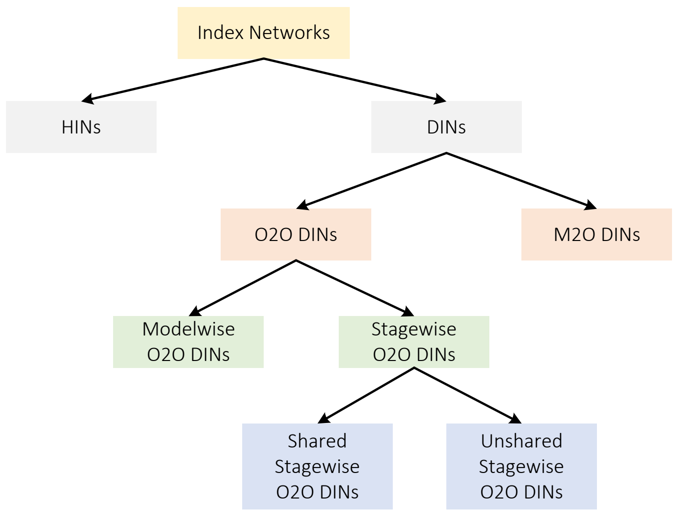


FIGURE 3.4: A taxonomy of proposed index networks.

all local regions (even if the kernel is learned), while $\mathcal{IU}$ upsamples different regions with different kernels (indices).

3.4.1 Index Networks

Here we present a taxonomy of proposed index networks. According to the shape of the output index map, index networks can be first categorized into two branches: *holistic index networks* (HINs) and *depthwise (separable) index networks* (DINs). Their conceptual differences are shown in Fig. 3.3. HINs learn an index function $I(\mathcal{X}) : \mathbb{R}^{H\times W\times C} \rightarrow \mathbb{R}^{H\times W\times 1}$. In this case, all channels of the feature map share a holistic index map. By contrast, DINs learn an index function $I(\mathcal{X}) : \mathbb{R}^{H\times W\times C} \rightarrow \mathbb{R}^{H\times W\times C}$, where the index map is of the same size as the feature map.

Since the index map generated by DINs can correspond to individual slices of the feature map, we can incorporate further assumptions into DINs to simplify the designs. If assuming that each slice of the index map only relates to its corresponding slice of the feature map, this is the One-to-One (O2O) assumption and *O2O DINs*. If each slice of the index map relates to all channels of the feature map, this leads to Many-to-One (M2O) assumption and *M2O DINs*. In O2O DINs, one can further consider sharing

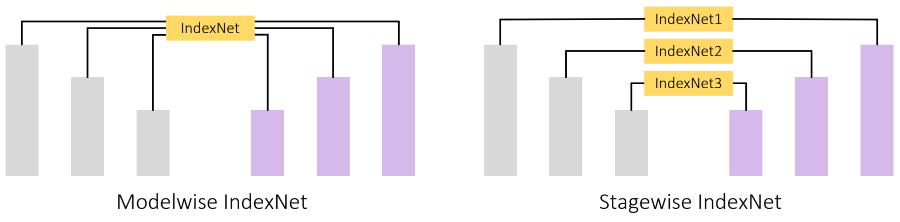


FIGURE 3.5: Modelwise IndexNet vs. stagewise IndexNet.

IndexNet. In the most simplified case, the same IndexNet can be applied to every slice of the feature map and can be shared across different downsampling/upsampling stages, like the max function. We name this IndexNet *Modelwise O2O DINs*. If IndexNet is stage-specific, i.e., only sharing indices in individual stages, we call this IndexNet *Shared Stagewise O2O DINs*. Finally, without sharing any parameter (each feature slice has its specific index function), we obtain the standard design, termed *Unshared Stagewise O2O DINs*. Fig. 3.4 shows the tree diagram of these index networks. The difference between modelwise IndexNet and stagewise IndexNet is also shown in Fig. 3.5. Notice that, HINs and M2O DINs are both stagewise.

With the taxonomy, we investigate five families of IndexNet. Each family can be designed to have either linear mappings or nonlinear mappings, as we discuss next.

3.4.2 Holistic Index Networks

Recall that HINs learn an index function $I(\mathcal{X}) : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^{H\times W\times 1}$. A naive design choice is to assume a linear mapping between the feature map and the index map.

Linear HINs. An example is shown in Fig. 3.6(a). The network is implemented in a fully convolutional network. It first applies stride-2 2×2 convolution (assuming that the downsampling rate is 2) to the feature map of size $H\times W\times C$, generating a concatenated index map of size $H/2\times W/2\times 4$. Each slice of the index map ($H/2\times W/2\times 1$) is designed to correspond to the indices of a certain position of all local regions, e.g., the top-left corner of all 2×2 regions. The network finally applies a $\mathcal{PS}$-like shuffling operator to rearrange the index map to the size of $H\times W\times 1$.

In many situations, a linear relationship is not sufficient. For example, a linear function even cannot approximate the max function. Thus, the second design choice is to introduce nonlinearity into the network.

Nonlinear HINs. Fig. 3.6(b) illustrates a nonlinear HIN where the feature map is first projected to a map of size $H/2\times W/2\times 2C$, followed by a batch normalization layer and a ReLU function for nonlinear mappings. We then use point-wise convolution to reduce

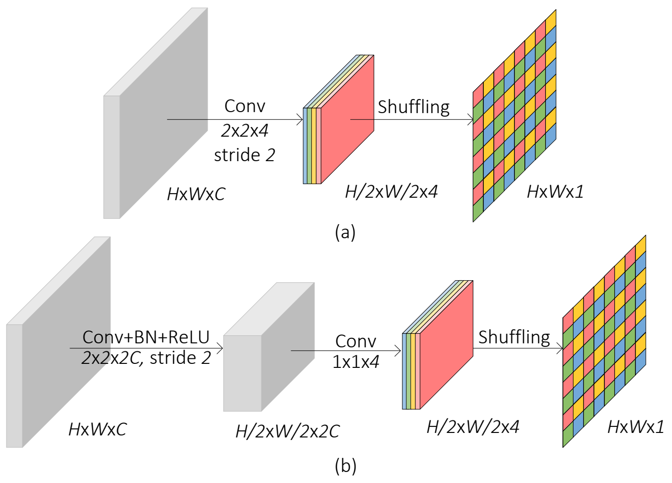


FIGURE 3.6: Holistic index networks. (a) a linear index network; (b) a nonlinear index network.

the channel dimension to an indices-compatible size. The remaining transformations follow its linear counterpart.

3.4.3 Depthwise Index Networks

In DINs, we seek $I(\mathcal{X}) : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^{H\times W\times C}$, i.e., each spatial index corresponds to each spatial activation. As aforementioned, this type of networks further has two different high-level design strategies that correspond to two different assumptions.

3.4.3.1 One-to-One Depthwise Index Networks

O2O assumption assumes that each slice of the index map only relates to its corresponding slice of the feature map. It can be denoted by a local index function $l(\mathcal{X}) : \mathbb{R}^{k\times k\times 1} \to \mathbb{R}^{k\times k\times 1}$, where k denotes the size of the local region. Since the local index function operates on individual feature slices, we can design whether different feature slices share the same local index function. Such a weight sharing strategy can be applied at a modelwise level or at a stagewise level, which leads to the following designs of O2O DINs:

1. *Modelwise O2O DINs*: the model only has a unique index function that is shared by all feature slices, even in different downsampling and upsampling stages. This is the most light-weight design;

2. *Shared Stagewise O2O DINs*: the index function is also shared by feature slices, but every stage has stage-specific IndexNet. This design is also light-weight;

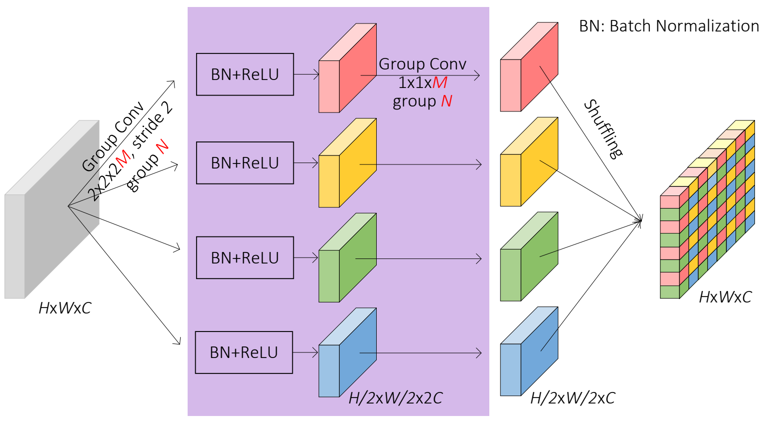


FIGURE 3.7: Depthwise index networks. $M = 1, N = 1$ for Modelwise O2O DINs and Shared Stagewise O2O DINs; $M = C, N = C$ for Unshared Stagewise O2O DINs; and $M = C, N = 1$ for the M2O DINs. The masked modules are invisible to linear networks.

3. *Unshared Stagewise O2O DINs*: even in the same stage, different feature slices have distinct index functions.

Similar to HINs, DINs can also be designed to have linear/nonlinear modeling ability. Fig. 3.7 shows an example when $k = 2$. Note that, in contrast to HINs, DINs follow a multi-column architecture. Each column is responsible for predicting indices specific to a certain spatial location of all local regions. We implement DINs with group convolutions.

Linear O2O DINs. According to Fig. 3.7, the feature map first goes through four parallel convolutional layers with the same kernel size. Modelwise O2O DINs and Shared Stagewise O2O DINs only use a kernel size of $2 \times 2 \times 1$, a stride of 2, and 1 group, while Unshared Stagewise O2O DINs has a kernel size of $2 \times 2 \times C$, a stride of 2, and C groups. One can simply reshape the feature map, i.e., reshaping $H \times W \times C$ to be $C \times H \times W \times 1$, to enable a $2 \times 2 \times 1$ kernel operating on each $H \times W \times 1$ feature slice, respectively. All O2O DINs lead to four downsampled feature maps of size $H/2 \times W/2 \times C$. The final index map of size $H \times W \times C$ is composed from the four feature maps by shuffling and rearrangement. Note that the parameters of four columns are not shared.

Nonlinear O2O DINs. Nonlinear DINs can be easily modified from linear DINs by inserting four extra convolutional layers. Each of them is followed by a batch normalization (BN) layer and a ReLU unit, as shown in Fig. 3.7. The rest remains the same as the linear DINs.

3.4.3.2 Many-to-One Depthwise Index Networks

M2O assumption assumes that all feature slices have contributions to each index slice. The local index function is defined by $l(\mathcal{X}) : \mathbb{R}^{k \times k \times C} \rightarrow \mathbb{R}^{k \times k \times 1}$. Compared to O2O

DINs, the only difference in implementation is the use of standard convolution instead of group convolution, i.e., $M = C, N = 1$ in Fig. 3.7.

3.4.4 Property and Model Complexity

Both HINs and DINs have merits and drawbacks. Here we discuss some important properties of IndexNet. We also present an analysis of computational complexity.

Remark 3.1. Index maps generated by HINs and used by the $\mathcal{IP}$ and $\mathcal{IU}$ operators are related to spatial attention.

The holistic index map is shared by all feature slices, which means that the index map is required to be expanded to the size of $H \times W \times C$ when feeding into $\mathcal{IP}$ and $\mathcal{IU}$. This index map can be thought as a collection of local attention maps [77] applied to individual local spatial regions. In this case, the $\mathcal{IP}$ and $\mathcal{IU}$ operators can also be referred to as "attentional pooling" and "attentional upsampling". However, it should be noted that spatial attention has no pooling or upsampling operators like $\mathcal{IP}$ and $\mathcal{IU}$.

Remark 3.2. HINs are more flexible than DINs and more friendly for decoder design.

Since the holistic index map is expandable, the decoder feature map does not need to forcibly increase/reduce its dimensionality to fit the shape of the index map during upsampling. This gives much flexibility for decoder design, while it is not the case for DINs.

Remark 3.3. The number of parameters in Modelwise O2O DINs and Shared Stagewise O2O DINs is independent of the dimensionality of feature maps.

No matter how large the model capacity is or how wide the feature channels are, the number of parameters in Modelwise O2O DINs remains at a constant level, and that in Shared Stagewise O2O DINs is only proportional to the number of downsampling/upsampling stages. This is desirable as the number of parameters introduced by IndexNet is not significant. However, these two types of IndexNet may be limited to capture sophisticated local patterns.

Remark 3.4. M2O DINs have the most powerful modeling capability among IndexNet variants, but also introduce many extra parameters.

M2O DINs have higher capacity than HINs and O2O DINs due to the use of standard convolution.

Another desirable property of IndexNet is that they may be able to predict the indices from a large local feature map, e.g., $l(\mathcal{X}) : \mathbb{R}^{2k \times 2k \times C} \rightarrow \mathbb{R}^{k \times k \times 1}$. An intuition behind this

IndexNet	Type	# Param.
HINs	L	$K \times K \times C \times 4$
	NL	$K \times K \times C \times 2C + 2C \times 4$
	NL+C	$2K \times 2K \times C \times 2C + 2C \times 4$
Modelwise O2O DINs	L	$(K \times K) \times 4$
	NL	$(K \times K \times 2 + 2) \times 4$
	NL+C	$(2K \times 2K \times 2 + 2) \times 4$
Shared Stagewise O2O DINs	L	$(K \times K) \times 4$
	NL	$(K \times K \times 2 + 2) \times 4$
	NL+C	$(2K \times 2K \times 2 + 2) \times 4$
Unshared Stagewise O2O DINs	L	$(K \times K \times C) \times 4$
	NL	$(K \times K \times 2C + 2C \times C) \times 4$
	NL+C	$(2K \times 2K \times 2C + 2C \times C) \times 4$
M2O DINs	L	$(K \times K \times C \times C) \times 4$
	NL	$(K \times K \times C \times 2C + 2C \times C) \times 4$
	NL+C	$(2K \times 2K \times C \times 2C + 2C \times C) \times 4$

L: Linear; NL: Nonlinear; C: Context.

TABLE 3.1: A Comparison of Model Complexity of Different Index Networks

idea is that, if one identifies a local maximum point from a $k \times k$ region, its surrounding $2k \times 2k$ region can further support whether this point is a part of a boundary or only an isolated noise point. This idea can be easily implemented by enlarging the convolutional kernel size and with appropriate padding.

In Table 3.1, we summarize the model complexity of different index networks used at a single downsampling and upsampling stage. We assume the convolution kernel has a size of $K \times K$ applied on a C-channel feature map. The number of parameters in BN layers is excluded. When considering weak context, we assume the kernel size is $2K \times 2K$. Since $C \gg K$, generally we have the model complexity *M2O DINs>HINs>Unshared Stagewise O2O DINs>Shared Stagewise O2O DINs>Modelwise O2O DINs.*

3.5 Guided Upsampling or Blind Upsampling: A Reconstruction-Based Justification

Here we introduce the concept of *guided upsampling* and *blind upsampling* to summarize existing upsampling operators. Particularly, here we present a comparison between two data-dependent upsampling operators—IndexNet and CARAFE [100]. In addition, we show results of an image reconstruction task on synthetic data, highlighting the difference between guided upsampling and blind upsampling.

3.5.1 Guided Upsampling vs. Blind Upsampling

By blind upsampling, we mean that an upsampling operator that is pre-defined with fixed parameters. Guided upsampling, instead, is guided with the involvement of auxiliary information.

Thus, most widely-used upsampling operators perform blind upsampling. These operators include nearest-neighbor (NN) interpolation, bilinear interpolation, space-to-depth and deconvolution. It is worth noting that the recent data-dependent upsampling operator CARAFE is also a blind upsampling operator. By contrast, guided upsampling operators are rare in literature. Max unpooling, albeit simple, is a guided upsampling operator. The auxiliary information used in upsampling comes from the max-pooling indices. Therefore, our proposed IndexNet clearly implements guided upsampling, with inferred dynamic indices as the auxiliary information.

The main difference here is that guided upsampling is made possible to exploit extra information to better recover the spatial information during upsampling. Thus, it is important that the spatial information is properly encoded during downsampling and is transferred to upsampling.

3.5.2 IndexNet vs. CARAFE

Both IndexNet and CARAFE are one of the few attempts pursuing the idea of data-dependent upsampling. The similarities include:

i) They both are related to dynamic networks.

ii) Both are parametric upsampling operators;

iii) CARAFE and HINs both perform holistic upsampling.

iv) CARAFE also learns an index function. The index function has an identical form to Equation (3.3), but with dynamic and normalized $\boldsymbol{W}$. In this sense, CARAFE may be considered as a single-input version of $\mathcal{IU}$, where the index map is generated internally.

The differences are:

i) CARAFE is a blind upsampling operator, while IndexNet implements guided upsampling;

ii) The reassembly kernels in CARAFE are generated conditioned on the low-resolution decoder feature map. The index maps predicted by IndexNet, however, build upon the high-resolution encoder feature map, before spatial information is lost;

iii) In IndexNet, each upsampled feature point only associates with a single point in the low-resolution feature map. From low resolution to high resolution, it is a one-to-many mapping. In CARAFE, each upsampled point is a weighted sum of a local region from the low-resolution feature map. This is a many-to-one mapping;

iv) Compared with CARAFE which is presented as a single upsampling operator, IndexNet is a more general framework.

In particular, the key difference lies in the intermediate path that allows spatial information to be visible to upsampling. To further demonstrate the benefit of this intermediate path, we present an image reconstruction experiment on synthetic data, namely, the Fashion-MNIST dataset [110].

The idea is that, if an upsampling operator can recover spatial information well from downsampled feature maps, the reconstructed output should be visually closer to the input image. The quality of reconstruction results can be a good indicator how well spatial information is recovered by an upsampling operator. We build the following baselines:

i) Average Pooling–NN interpolation (AvgPool–NN);

ii) stride-2 Convolution–Bilinear interpolation(Conv$_{/2}$–Bilinear);

iii) Space-to-Depth–Depth-to-Space (S2D–D2S);

iv) stride-2 Convolution–2-stride Deconvolution (Conv$_{/2}$–Deconv$_{/2}$);

v) stride-2 Convolution–CARAFE (Conv$_{/2}$–CARAFE);

vi) Max Pooling–Max Unpooling (MaxPool–MaxUnpool);

vii) Indexed Pooling–Indexed Upsampling ($\mathcal{IP}$–$\mathcal{IU}$).

Table 3.2 reports the reconstruction results, showing the advantage of IndexNet over CARAFE. Details can be found in the Appendix.

3.6 Applications

In this section, we first display results on image matting, then we show several extensional applications of IndexNet on the tasks of image denoising, semantic segmentation, and monocular depth estimation.

	PSNR	SSIM	MAE	MSE
AvgPool–NN	25.88	0.9811	0.0259	0.0509
Conv$_{/2}$–Bilinear	24.45	0.9726	0.0320	0.0600
S2D–D2S	28.93	0.9901	0.0204	0.0358
Conv$_{/2}$–Deconv$_{/2}$	28.75	0.9903	0.0187	0.0366
Conv$_{/2}$–CARAFE	25.55	0.9798	0.0277	0.0529
MaxPool–MaxUnpool	29.33	0.9920	0.0202	0.0342
$\mathcal{IP}$–$\mathcal{IU}^*$	37.83	0.9989	0.0089	0.0128
$\mathcal{IP}$–$\mathcal{IU}^\dagger$	45.93	0.9998	0.0032	0.0051
$\mathcal{IP}$–$\mathcal{IU}^\ddagger$	**48.37**	**0.9999**	**0.0026**	**0.0038**

^{1} * denotes Modelwise O2O DIN; † indicates HIN; ‡ refers to M2O DIN. All IndexNets are with nonlinearity and weak context. The best performance is boldfaced.

TABLE 3.2: Performance of Image Reconstruction on the Fashion-MNIST Dataset

3.6.1 Image Matting

In this application, we use Deep Image Matting (DIM) [113] as our baseline. Image matting is particularly suitable for evaluating the effectiveness of IndexNet, because the quality of learned indices can be visually observed from inferred alpha mattes. We conduct experiments on the Adobe Image Matting dataset [113].

We evaluate the results using widely-used Sum of Absolute Differences (SAD), root Mean Square Error (MSE), and perceptually-motivated Gradient (Grad) and Connectivity (Conn) errors [81]. The evaluation code implemented by [113] is used. In what follows, we first describe our modified MobileNetv2-based architecture and training details. We then perform extensive ablation studies to justify choices of model design, make comparisons of different index networks, and visualize learned indices.

3.6.1.1 Network Architecture and Implementation Details

Here we describe the network architecture and training details.

Network Architecture. We build our model based on MobileNetv2 [86] with only slight modifications to the backbone. We choose MobileNetv2 for its lightweight model and fast inference. It also follows the encoder-decoder paradigm same as SegNet. We simply change all 2-stride convolution to be 1-stride and attach 2-stride 2×2 max pooling after each encoding stage for downsampling, which allows us to extract indices. If applying the IndexNet idea, max pooling and unpooling layers can be replaced with $\mathcal{IP}$ and $\mathcal{IU}$, respectively. We also investigate alternative ways for low-level feature fusion and whether encoding context (Section 3.6.1.2). Note that, the matting refinement stage [113] is not applied here.

No.	Architecture	Backbone	Fusion	Indices	Context	OS	SAD	MSE	Grad	Conn
B1	DeepLabv3+ [9]	MobileNetv2	Concat	No	ASPP	16	60.0	0.020	39.9	61.3
B2	RefineNet [62]	MobileNetv2	Skip	No	CRP	32	60.2	0.020	41.6	61.4
B3	SegNet [113]	VGG16	No	Yes	No	32	**54.6**	**0.017**	36.7	55.3
B4	SegNet	VGG16	No	No	No	32	122.4	0.100	161.2	130.1
B5	SegNet	MobileNetv2	No	Yes	No	32	60.7	0.021	40.0	61.9
B6	SegNet	MobileNetv2	No	No	No	32	78.6	0.031	101.6	82.5
B7	SegNet	MobileNetv2	No	Yes	ASPP	32	58.0	0.021	39.0	59.5
B8	SegNet	MobileNetv2	Skip	Yes	No	32	57.1	0.019	36.7	57.0
B9	SegNet	MobileNetv2	Skip	Yes	ASPP	32	56.0	**0.017**	38.9	55.9
B10	UNet	MobileNetv2	Concat	Yes	No	32	54.7	**0.017**	34.3	**54.7**
B11	UNet	MobileNetv2	Concat	Yes	ASPP	32	54.9	**0.017**	**33.8**	55.2

Fusion: fuse encoder features; Indices: max-pooling indices (where Indices is ‘No’, bilinear interpolation is used for upsampling); CRP: chained residual pooling [62]; ASPP: atrous spatial pyramid pooling [9]; OS: output stride. The lowest errors are boldfaced.

TABLE 3.3: Ablation Study of Design Choices

Training Details. To enable a direct comparison with deep matting [113], we follow the same training configurations used in [113]. The 4-channel input concatenates the RGB image and its trimap. We follow exactly the same data augmentation strategies, including 320×320 random cropping, random flipping, random scaling, and random trimap dilation. We use a combination of the alpha prediction loss and the composition loss during training as in [113]. Only losses from the unknown region of the trimap are calculated. Encoder parameters are pretrained on ImageNet [26]. The parameters of the 4-th input channel are initialized with zeros. The Adam optimizer [52] is used. We update parameters with 30 epochs (around $90,000$ iterations). The learning rate is initially set to 0.01 and reduced by $10\times$ at the 20-th and 26-th epoch respectively. We use a batch size of 16 and fix the BN layers of the backbone.

3.6.1.2 Results on the Adobe Image Matting Dataset

Ablation Study on Model Design. To establish a better baseline comparable to DIM, here we first investigate strategies for fusing low-level features (no fusion, skip fusion as in ResNet [41] or concatenation as in UNet [82]) and whether encoding context for image matting. 11 baselines are consequently built to justify model design. Results on the Composition-1k testing set are reported in Table 3.3. B3 is cited from [113]. We can make the following observations:

i) Indices are of great importance. Matting can significantly benefit from only indices (B3 vs. B4, B5 vs. B6);

ii) State-of-the-art semantic segmentation models cannot be directly applied to image matting (B1/B2 vs. B3);

iii) Fusing low-level features help, and concatenation is better than skip connection but at a cost of increased computation (B6 vs. B8 vs. B10 or B7 vs. B9 vs. B11);

iv) Modules such as ASPP may improve the results (e.g., B6 vs. B7 or B8).

Method		#Param.	GFLOPs	SAD	MSE	Grad	Conn
B3 [113]		130.55M	32.34	54.6	0.017	36.7	55.3
B11		3.75M	4.08	54.9	0.017	33.8	55.2
B11-1.4		8.86M	7.61	55.6	0.016	36.4	55.7
B11-carafe		4.06M	5.01	50.2	0.015	27.9	50.0
HMI		3.75M	4.08	56.5	0.021	33.0	56.4
NL	C	Δ					
		HINs					
		+4.99K	4.09	55.1	0.018	32.1	55.2
✓		+0.26M	4.22	50.6	0.015	27.9	49.4
✓	✓	+1.04M	4.61	49.5	0.015	**25.6**	49.2
		Modelwise O2O DINs					
		+16	4.08	57.3	0.017	37.3	57.4
✓		+56	4.08	52.4	0.016	30.1	52.2
✓	✓	+152	4.08	59.1	0.018	39.0	59.7
		Shared Stagewise O2O DINs					
		+80	4.08	48.9	0.014	26.2	48.0
✓		+280	4.08	51.1	0.016	30.2	50.7
✓	✓	+760	4.08	56.0	0.016	37.5	55.9
		Unshared Stagewise O2O DINs					
		+4.99K	4.09	50.3	0.015	33.7	50.0
✓		+17.47K	4.10	50.6	0.016	26.5	50.3
✓	✓	+47.42K	4.15	50.2	0.016	26.8	49.3
		M2O DINs					
		+0.52M	4.34	51.0	0.015	33.7	50.5
✓		+1.30M	4.73	48.9	0.015	32.1	47.9
✓	✓	+4.40M	6.30	**45.8**	**0.013**	25.9	**43.7**
DIM w. Refinement [113]				50.4	0.014	31.0	50.8

NL: Non-Linearity; C: Context. Δ indicates increased parameters compared to B11. GFLOPs are measured on a $224 \times 224 \times 4$ input. The lowest errors are boldfaced.

TABLE 3.4: Results on the Composition-1k Testing Set

v) A MobileNetv2-based matting model can work as well as a VGG-16-based one (B3 vs. B11).

For the following experiments, we now mainly use B11.

Ablation Study on Index Networks. Here we compare different index networks and justify their effectiveness. The configurations of index networks used in the experiments follow Figs. 3.6 and 3.7. We primarily investigate the 2×2 kernel with a stride of 2. Whenever the weak context is considered, we use a 4×4 kernel in the first convolutional layer of index networks. To highlight the effectiveness of HINs, we further build a baseline called *holistic max index* (HMI) where max-pooling indices are extracted from a squeezed feature map $\mathcal{X}' \in \mathbb{R}^{H \times W \times 1}$. $\mathcal{X}'$ is generated by applying the max function along the channel dimension of $\mathcal{X} \in \mathbb{R}^{H \times W \times C}$. Furthermore, since IndexNet increases extra parameters, we introduce another baseline *B11-1.4* where the width multiplier of MobilieNetV2 is adjusted to be 1.4 to increase the model capacity. In addition, to compare IndexNet against CARAFE in this task, we build an additional baseline *B11-carafe* where the unpooling operator in B11 is replaced with CARAFE. Results on the Composition-1k testing dataset are listed in Table 3.4. We observe that, most index networks reduce the errors notably, except for some low-capacity IndexNet modules (due to limited modeling capabilities). In particular, nonlinearity and the context generally have a positive effect on deep image matting, but they do not work effectively in O2O DINs.

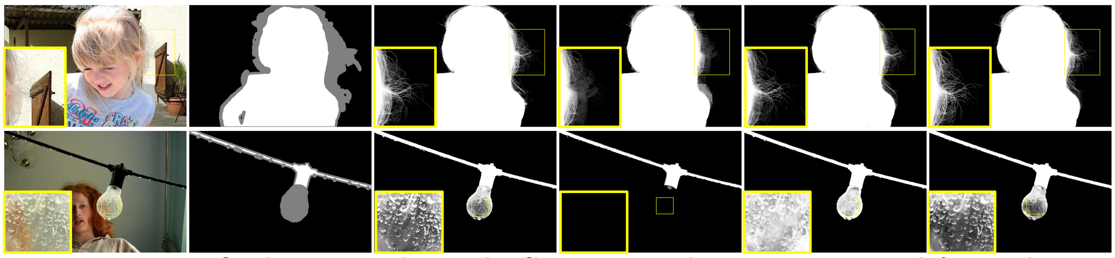


FIGURE 3.8: Qualitative results on the Composition-1k testing set. From left to right, the original image, trimap, ground-truth alpha matte, Closed-Form matting (CF) [57], DIM [57], and ours (M2O DIN with 'Nonlinearity+Context').

Encoder	Decoder	SAD	MSE	Grad	Conn
sigmoid	sigmoid	52.7	0.016	29.3	52.4
softmax	softmax	51.6	0.015	29.2	51.6
softmax+sigmoid	softmax	57.3	0.016	43.5	57.3
sigmoid+softmax	sigmoid	**45.8**	**0.013**	**25.9**	**43.7**

The lowest errors are boldfaced.

TABLE 3.5: Ablation Study of Different Normalization Choices on Index Maps

A possible reason may be that the limited dimensionality of the intermediate feature map is not sufficient to model complex patterns in matting. Compared to holistic max index, the direct baseline of HINs, the best HIN ("Nonlinearity+Context") has at least 12.3% relative improvement. Compared to B11, the baseline of DINs, M2O DIN with "Nonlinearity+Context" exhibits at least 16.5% relative improvement. Notice that, our best model outperforms the DIM approach [113] that even has the refinement stage. In addition, according to the results of B11-1.4, the performance improvement does not come from increased parameters. Moreover, CARAFE also enhances matting performance, but it falls behind M2O DIN. Some qualitative results are shown in Fig. 3.8. Our predicted mattes show improved delineation for edges and textures like hair and water drops.

Ablation Study on Index Normalization. Index normalization is important for the final performance. Here we justify this by evaluating different normalization choices. Apart from the sigmoid function used for the decoder and the sigmoid+softmax function for the encoder, we compare other three different combinations of normalization strategies listed in Table 3.5. The experiment is conducted based on M2O DIN with "Nonlinearity+Context". It is clear that keeping the magnitude consistency during downsampling matters. In fact, both max pooling and average pooling satisfy this property naturally, and our normalization design is inspired from this fact.

3.6.2 Image Denoising

The goal of image denoising is to recover a clean image $\boldsymbol{x}$ from a corrupted observation $\boldsymbol{y}$ following an image degradation model $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{v}$, where $\boldsymbol{v}$ is commonly assumed to be

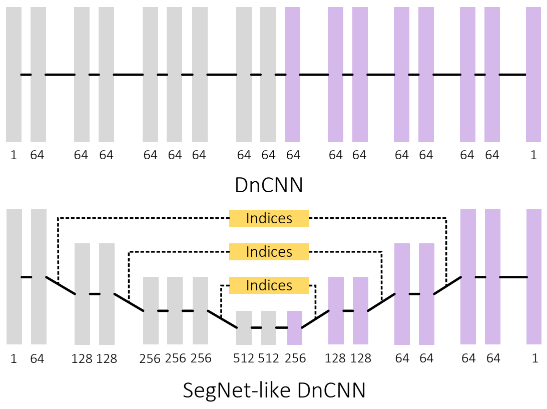


FIGURE 3.9: The DnCNN architecture and our modified SegNet-like DnCNN.

additive white Gaussian noise (AWGN) parameterized by σ. While such an assumption has been challenged in recent real-image denoising [6, 49], we still follow the AWGN paradigm in evaluation because our focus is not to improve image denoising. Most deep denoising models [76, 122] are designed with the same high-level idea—processing the feature map without decreasing its spatial resolution. It has been observed that, when the feature map is downsampled, the performance drops remarkably [76]. For such networks, although the model parameters are largely reduced, computational complexity of training and inference becomes much heavier.

We show that, by inserting IndexNet into a denoising model, it can effectively compensate the loss of spatial information, achieving performance comparable to or even better than the network without downsampling. Thus, despite the number of parameters increases, computation us much reduced. We choose DnCNN [122] as our baseline to demonstrate this on standard benchmarks. We follow the experimental setting of [14] that uses a 400-image training set. The performance is reported on a 68-image Berkeley segmentation dataset (BSD68) and the other 12-image test set (Set12). The networks are trained for Gaussian denoising, with three noise levels, i.e., $\sigma = 15, 25$ and 50. PSNR and SSIM are used as evaluation metrics.

3.6.2.1 Network Architecture and Implementation Details

Network Architecture. We use the 17-layer DnCNN model [122], implemented by PyTorch. To enable the use of IndexNet, we modify DnCNN to a SegNet-like architecture with 3 downsampling and upsampling stages (the input image size is 40×40). The number of layers remains the same to ensure a relatively fair comparison. Fig. 3.9 illustrates the original DnCNN and our modified architecture. The first 9 layers follow VGG-16 except that the first layer is a single-channel input, and the rest are 7 decoding layers formed by unpooling and convolution and the final prediction layer. All convolutional operations use 3×3 kernels. To incorporate IndexNet, it is straightforward to replace max pooling and unpooling with $\mathcal{IP}$ and $\mathcal{IU}$.

Training Details. We follow the same experimental configurations used in [122]. At each epoch, 40×40 image patches are cropped from multiple scales $(0.7, 0.8, 0.9, 1)$ with a stride of 10 and are added with Gaussian noise of a certain noise level ($\sigma = 15, 25$, or 50); image patches are further augmented with random flipping and random rotation. This results in around $240,000$ training samples. ℓ_2 loss is used. All networks are trained from scratch with a batch size of 128. Model parameters are initialized with the improved Xavier [42]. The Adam optimizer is also used. Parameters are updated with 60 epochs. The learning rate is initially set to 0.001 and reduced by $10\times$ at the 45-th and 55-th epoch, respectively.

3.6.2.2 Results on the BSD68 and Set12 Datasets

Apart from the DnCNN baseline, we also report the performance of our modified DnCNN-SegNet with max pooling and unpooling. Furthermore, to compare IndexNet against CARAFE, we build three additional baselines where CARAFE is combined with different downsampling strategies, including max pooling, average pooling, and stride-2 convolutions, denoted by DnCNN-max-carafe, DnCNN-avg-carafe, and DnCNN-conv-carafe, respectively. Results are shown in Table 3.6. It can be observed that, simply downsampling with max pooling and upsampling by unpooling as in DnCNN-SegNet lead to significant drops in both PSNR (generally $> 1dB$) and SSIM (> 0.1). This suggests that spatial information plays an important role in image denoising. Denoising is content-irrelevant (the model is unaware of regions coming from the foreground or the background). Downsampling without recording sufficient spatial information (only the boundary information is not sufficient) impedes the model from recovering the appearance and the structure in the original image. This is particularly true for baselines adopting CARAFE. Since CARAFE applies blind upsampling, no spatial information is transferred during upsampling, which may lead to inferior results. Interestingly, after IndexNet is inserted into downsampled DnCNN, the loss of PSNR and SSIM is effectively compensated. The compensation behaviors can be observed from almost all types of IndexNet, except the two cases in Modelwise O2O DINs with nonlinearity. The poor performance of Modelwise O2O DINs may attribute to the insufficient modeling ability, particularly when $\sigma = 50$. Nonlinearity and weak context generally have a positive effect on image denoising, and the effectiveness of different IndexNets is similar. Hence, Shared Stagewise O2O DINs appear to be a preferred choice due to slightly increased parameters and negligible extra computation costs.

Method		#Param.	GFLOPs	BSD68			Set12		
Noise Level				15	25	50	15	25	50
DnCNN [122]		0.56M	25.89	31.74/0.9410	29.22/0.9015	26.23/0.8269	32.87/0.9544	30.42/0.9296	27.17/0.8775
DnCNN-SegNet		7.09M	18.14	30.74/0.9278	28.27/0.8752	24.88/0.7437	31.91/0.9395	28.98/0.8881	24.99/0.7485
DnCNN-max-carafe		7.29M	19.11	25.70/0.7578	21.41/0.5670	15.40/0.2988	24.64/0.6997	20.19/0.4965	14.22/0.2489
DnCNN-avg-carafe		7.29M	19.11	25.70/0.7578	21.43/0.5673	15.56/0.2968	24.64/0.6997	20.20/0.4968	14.26/0.2460
DnCNN-conv-carafe		7.29M	15.23	25.70/0.7578	21.43/0.5672	15.50/0.2994	24.64/0.6997	20.20/0.4967	14.22/0.2496
NL	C	Δ							
				HINs					
		+7.17K	18.16	31.13/0.9357	29.02/0.8997	26.29/0.8281	32.71/0.9536	30.28/0.9285	27.20/0.8789
✓		+0.69M	19.30	31.15/0.9356	29.01/0.8999	26.29/0.8301	32.77/0.9537	30.36/0.9295	27.18/0.8799
✓	✓	+2.76M	22.75	31.20/0.9365	29.05/0.9004	26.30/0.8305	32.79/0.9541	30.37/0.9300	27.22/0.8804
				Modelwise O2O DINs					
		+16	18.14	31.22/0.9366	29.06/0.9002	25.84/0.8294	32.83/0.9545	30.42/0.9302	26.21/0.8782
✓		+56	18.14	30.64/0.9255	27.39/0.8391	24.15/0.6776	31.92/0.9386	27.97/0.8330	24.14/0.6747
✓	✓	+152	18.14	30.87/0.9296	27.70/0.8617	24.09/0.6939	32.23/0.9432	28.31/0.8677	23.85/0.6634
				Shared Stagewise O2O DINs					
		+48	18.14	31.14/0.9364	29.05/0.9002	26.32/0.8310	32.80/0.9542	30.41/0.9302	27.24/0.8807
✓		+168	18.14	31.20/0.9365	28.97/0.9000	26.18/0.8272	32.83/0.9545	30.43/0.9302	27.24/0.8801
✓	✓	+456	18.14	31.22/0.9366	29.07/0.9004	26.31/0.8311	32.82/0.9543	30.41/0.9300	27.27/0.8814
				Unshared Stagewise O2O DINs					
		+7.17K	18.16	31.17/0.9366	28.25/0.8944	25.02/0.8235	32.80/0.9544	30.23/0.9286	26.41/0.8675
✓		+25.1K	18.19	31.25/0.9368	29.06/0.9002	26.33/0.8306	32.77/0.9541	30.43/0.9303	27.29/0.8814
✓	✓	+68.1K	18.32	31.21/0.9364	27.68/0.8740	26.33/0.8312	32.83/0.9544	30.32/0.9288	27.24/0.8807
				M2O DINs					
		+1.38M	20.44	31.22/0.9365	29.03/0.9005	26.33/0.8316	32.82/0.9544	30.42/0.9302	27.28/0.8812
✓		+3.45M	23.88	31.23/0.9368	29.07/0.9002	26.26/0.8278	32.84/0.9546	30.44/0.9304	27.28/0.8808
✓	✓	+11.7M	37.67	31.23/0.9365	29.06/0.8996	26.34/0.8315	32.82/0.9545	30.43/0.9301	27.29/0.8803

NL: Non-Linearity; C: Context. Δ indicates increased parameters compared to the SegNet-DnCNN baseline. GFLOPs are measured on a $224 \times 224 \times 1$ input.

TABLE 3.6: Average PSNR (dB) and SSIM Results of Various Noise Levels on the BSD68 and Set12 Image Denoising Benchmarks

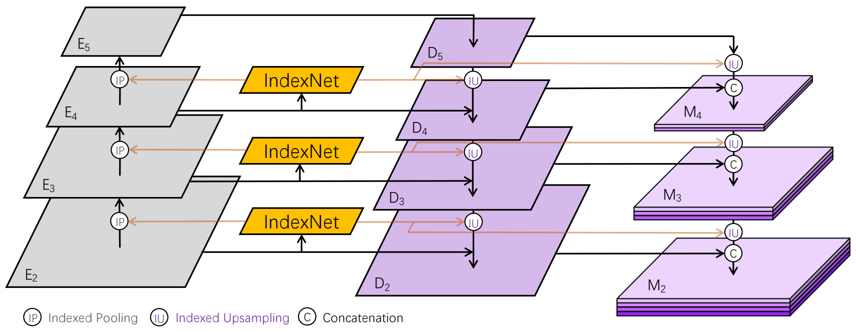


FIGURE 3.10: IndexNet-guided feature pyramid network and multi-level feature fusion.

3.6.3 Semantic Segmentation

Here we further evaluate IndexNet on semantic segmentation. Semantic segmentation aims to predict a dense labeling map for each image where each pixel is labeled into one category. Since the FCNs were introduced [69], FCN-based encoder-decoder architectures have been studied extensively [8, 62, 3, 9]. Efforts have been spent on how to encode contextual information. We use SegNet [3] as our baseline because IndexNet is primarily inspired by the unpooling operator in SegNet. We follow the experimental setting in [3] and report performance on the SUN RGB-D [95] dataset. We use RGB as the input (depth is not used). The standard mean Intersection-over-Union (mIoU) is used as the evaluation metric. we also compare against the recent UperNet [111]. We evaluate UperNet on the ADE20K dataset [127].

3.6.3.1 Network Architecture and Implementation Details

Network Architecture. The architecture of SegNet employs the first 13 layers of the VGG-16 model pretrained on ImageNet as the encoder. The decoder uses unpooling for upsampling. Each unpooling layer is followed by the same number of convolutional layers as in the corresponding encoder stage. Overall, SegNet has 5 downsampling and 5 upsampling stages. Convolutional layers in the decoding stage mainly play a role to smooth the feature maps generated by unpooling. To insert IndexNet, the only modification is to replace max pooling and unpooling layers with $\mathcal{IP}$ and $\mathcal{IU}$, respectively, which is straightforward.

UperNet builds upon the idea of Pyramid Pooling Module (PPM) [124] and Feature Pyramid Network (FPN) [64]. UperNet also implements a Multi-level Feature Fusion (MFF) module that fuses multi-resolution feature maps by concatenation. In FPN, downsampling is implemented by 2-stride convolution, and upsampling uses bilinear interpolation. It produces four feature levels $\{D_2, D_3, D_4, D_5\}$ with output strides of $\{4, 8, 16, 32\}$, conditioned on the encoder features $\{E_2, E_3, E_4, E_5\}$. MFF further fuses

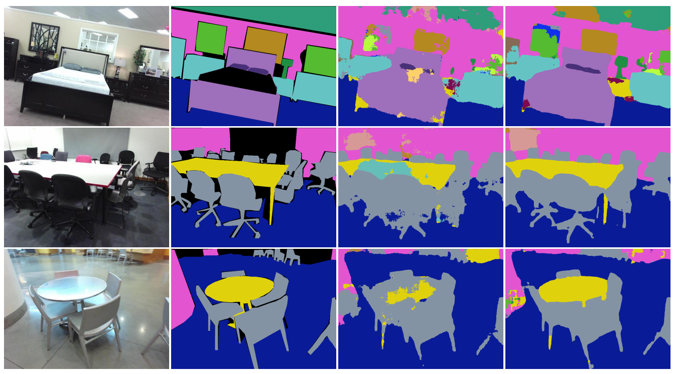


FIGURE 3.11: Scene understanding results on the SUNRGB-D dataset. From left to right, the original image, ground-truth, SegNet, and ours (Shared Stagewise O2O DIN with 'Nonlinearity').

four levels of features and generates the output M_2 with an output stride of 4. To insert IndexNet, three IndexNet blocks can be inserted into the encoder to generate index maps to guide upsampling. The same index maps can also be used in MFF in a sequential upsampling manner to fuse features, as shown in Fig. 3.10. Note that, in theory IndexNet can also be applied to PPM, because PPM itself has internal downsampling and upsampling stages. However, we discourage the use of IndexNet in PPM, because it will significantly increase parameters (due to mixed downsampling/upsampling rates). In this case blind upsampling such as NN/bilinear interpolation may be a better choice.

Training Details. On the SUN RGB-D dataset, the VGG-16 model pretrained on ImageNet with BN layers is used. We employ the standard data augmentation strategies: random scaling, random cropping 320×320 sub-images, and random horizontal flipping. We learn the model with the standard softmax loss. Encoder parameters are pretrained on ImageNet. All other parameters are initialized with the improved Xavier [42]. The SGD optimizer [52] is used with a momentum of 0.9 and a weight decay of 0.0001. We train the model with a batch size of 16 for 300 epochs (around $90,000$ iterations). The learning rate is initially set to 0.01 and reduced by $10\times$ at the 250-th and 280-th epoch, respectively. The BN layers of the encoder are fixed.

On the ADE20K benchmark, we use the MobileNetV2 pretrained on ImageNet as the encoder and UperNet the decoder. Due to limited computational resources, only this setting enables us to train a model on 4 GPUs with a batch size of 16 following the official implementation of UperNet and provided experimental settings.[1]

[1]https://github.com/CSAILVision/semantic-segmentation-pytorch

Method		#Param.	GFLOPs	mIoU
SegNet [3]		24.96M	24.76	32.47
SegNet-carafe		25.35M	25.75	**36.30**
NL	C	Δ		
		HINs		
		+23.55K	24.79	33.25
✓		+4.90M	26.40	33.11
✓	✓	+19.55M	31.28	33.31
		Modelwise O2O DINs		
		+16	24.76	33.18
✓		+56	24.76	33.70
✓	✓	+152	24.77	33.26
		Shared Stagewise O2O DINs		
		+80	24.76	33.26
✓		+280	24.76	33.97
✓	✓	+760	24.77	33.41
		Unshared Stagewise O2O DINs		
		+0.02M	24.79	33.27
✓		+0.08M	24.82	33.59
✓	✓	+0.22M	24.96	33.50
		M2O DINs		
		+9.76M	28.02	33.28
✓		+24.44M	32.90	33.51
✓	✓	+83.02M	52.42	33.48

NL: Non-Linearity; C: Context. Δ indicates increased parameters compared to the SegNet baseline. GFLOPs are measured on a $224 \times 224 \times 3$ input.

Table 3.7: Performance on the SUN RGB-D Dataset.

3.6.3.2 Results on the SUN RGB-D Dataset

We report the results in Table 3.7. All index networks show improvements over the baseline, among which Modelwise and Shared Stagewise O2O DINs improve the baseline with few extra parameters and GFLOPs. Compared with other types of IndexNet, M2O DINs and HINs (particularly under the setting of "Nonlinearity+Context") increase many parameters and GFLOPs but do not exhibit clear advantages.

We hypothesize that the improvement comes from the ability of IndexNet suppressing fractured predictions that frequently appears in the baseline SegNet. IndexNet seemingly does better in producing predictions at boundaries.

Notice that, in contrast to the behaviour in matting and denoising, CARAFE significantly enhances the performance in segmentation ($32.47 \rightarrow 36.30$), outperforming IndexNet. We observe that CARAFE tends to produce consistent region-wise predictions. A plausible explanation is that, CARAFE is designed in a way to tackle region-sensitive tasks such as semantic segmentation where region-wise matching between predictions and ground truths matters, while IndexNet prefers detail-sensitive tasks like image matting where errors come from detail-rich regions.

FPN	MFF	mIoU	Pixel Accuracy (%)
Bilinear	Bilinear	37.08	78.29
IndexNet	Bilinear	36.25	78.23
Bilinear	IndexNet	36.90	78.27
IndexNet	IndexNet	37.62	78.29
CARAFE	Bilinear	37.76	78.81
Bilinear	CARAFE	38.03	78.51
CARAFE	CARAFE	**38.31**	**78.90**

MFF: Multi-level Feature Fusion. Only HINs ('Linear') are evaluated due to varied decoder feature dimensionality. The best performance is boldfaced.

Table 3.8: Performance on the ADE-20K Dataset

3.6.3.3 Results on the ADE20K Dataset

Here we conduct ablative studies to highlight the role of upsampling in UperNet. Both IndexNet and CARAFE are considered. In addition to the full replacement of upsampling operators following Fig. 3.10, we also replace bilinear upsampling either in FPN or in MFF with IndexNet/CARAFE. Results are shown in Table 3.8. It can be observed that, IndexNet improves UperNet ($37.08 \rightarrow 37.62$) only when bilinear upsampling in FPN and MFF is simultaneously replaced. However, when only one component is modified, IndexNet even leads to negative results. This suggests that the guided information should be used consistently in the decoder. In addition, CARAFE also works better than IndexNet, showing that spatial information may not play a critical role in semantic segmentation.

3.6.4 Monocular Depth Estimation

Estimating per-pixel depth from a single image is challenging because one needs to recover 3D information from a 2D plane. With deep learning, significant progress has been witnessed [66, 109, 105]. We use the recent FastDepth [105] as our baseline. We compare the performance with/without IndexNet on the NYUv2 dataset [91] with the official train/test split. To be in consistent with [105], the following metrics are used to quantify the performance:

- root mean square error (rms): $\sqrt{\frac{1}{T}\sum_{i=1}^{T}(d_i - g_i)^2}$;
- accuracy with threshold th: percentage (%) of d_1, s.t. $\max\left(\frac{d_1}{g_1}, \frac{g_1}{d}\right) = \delta_1 < th$.

Details about the network and training can be found in the Appendix.

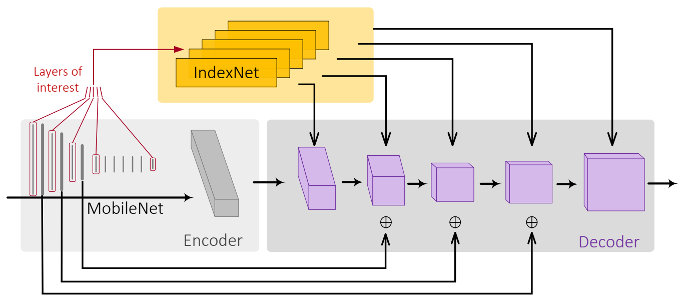


FIGURE 3.12: Our modified FastDepth [105] architecture.

3.6.4.1 Network Architecture and Implementation Details

Network Architecture. FastDepth is an encoder-decoder architecture, with MobileNet as its backbone. Here we choose the best upsampling option suggested by the authors [105] where upsampling is implemented by $\times 2$ NN interpolation and 5×5 convolution. Hence, our baseline is FastDepth-NNConv5: downsampling with 2-stride convolution and upsampling via NN interpolation. We also modify this baseline by changing the stride-2 convolution to be stride-1 followed by max-pooling, named as FastDepth-P-NNConv5. Fig. 3.12 shows how we insert IndexNet into FastDepth. Similar to the modifications applied to the matting network, stride-2 convolution layers in the encoder are changed to be stride-1, followed by $\mathcal{IP}$, and the NN interpolation in the decoder is replaced with $\mathcal{IU}$. To compare IndexNet with CARAFE, we build two additional baselines: FastDepth-carafe and FastDepth-P-carafe, where NNConv5 is modified to CARAFE in FastDepth-NNConv5 and FastDepth-P-NNConv5.

Training Details. We follow similar training settings used by FastDepth [105]. ℓ_1 loss is used. Random rotation, random scaling and random horizontal flipping are used for data augmentation. The initial learning rate is set to 0.01 and reduced by $\times 10$ every 5 epochs. The SGD optimizer is used with a momentum of 0.9 and a weight decay of 0.0001. Encoder weights are pretrained on ImageNet [26]. A batch size of 16 is used to train the network for 30 epochs in total.

3.6.4.2 Results on the NYUDv2 Dataset

We report the results in Table 3.9. We observe that almost all types of IndexNet improve the performance compared to the baselines except for the most light-weight design—linear Modelwise O2O DIN. It may be because only 16 parameters are not sufficient to model local variations of high-dimensional feature maps. Note that, Unshared Stagewise O2O DINs (with only linear mappings) shows clear improvements with only slightly increased parameters. HINs and M2O DINs increase a large amount of parameters and

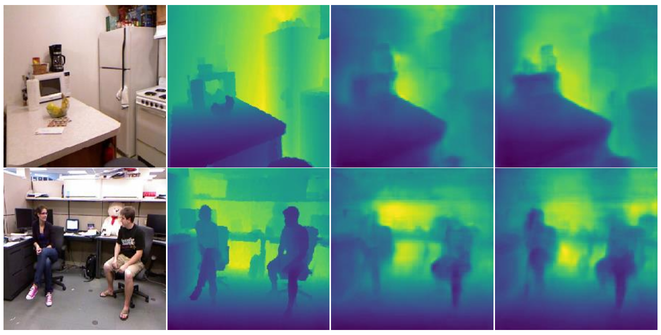


FIGURE 3.13: Qualitative results on the NYUDv2 dataset. From left to right, the original image, ground-truth, FastDepth-NNConv5, and ours (Unshared Stagewise O2O DIN with 'Linear').

floating-point calculations because of the high dimensionality of feature maps, while the improved performance is not proportional to such a high cost.

We observe that IndexNet exhibits better boundary delineation than the baseline, e.g., the edge of the desk, and the contour of the woman. Moreover, CARAFE achieves comparable performance against IndexNet in this task.

In addition, we report the results of applying IndexNet to a state-of-the-art model [47] in Table 3.10. We modify the use of IndexNet here by taking the same feature fusion strategies of Feature Pyramid Network and Multi-level Feature Fusion, as used in our semantic segmentation experiment.

Other implementation details and evaluations are kept consistent with [72]. Compared with the baseline, IndexNet shows improvement in the first four metrics.

3.6.5 Insights Towards Good Practices

As a summary of our evaluations, here we provide some guidelines for using guided/blind upsampling:

1. In detail-sensitive tasks, such as image matting, image restoration, and edge detection, the spatial information is important. Thus guided upsampling may be preferred.

2. Blind upsampling may be used in the situation when computational budget is limited because most blind upsampling operators are non-parametric and computationally efficient.

3. In image matting, the best IndexNet configuration is "M2O DINs+Nonlinearity+Context". This configuration is also true for the image reconstruction experiment and image

Method		#Param.	GFLOPs	rms	$\delta_1 < 1.25$
FastDepth-NNConv5		3.96M	0.69	0.567	0.781
FastDepth-P-NNConv5		3.96M	1.01	0.577	0.778
FastDepth-carafe		4.31M	1.63	0.558	**0.790**
FastDepth-P-carafe		4.31M	1.96	0.571	0.782
NL	C	Δ			
		HINs			
		+31.23K	1.03	0.566	0.784
✓		+11.17M	2.65	0.565	0.786
✓	✓	+44.62M	7.53	0.559	0.787
		Modelwise O2O DINs			
		+16	1.02	0.569	0.778
✓		+56	1.02	0.568	0.785
✓	✓	+152	1.02	0.564	0.786
		Shared Stagewise O2O DINs			
		+80	1.02	0.562	0.783
✓		+280	1.02	0.565	0.786
✓	✓	+760	1.02	0.567	0.783
		Unshared Stagewise O2O DINs			
		+31.23K	1.03	**0.556**	0.789
✓		+0.11M	1.06	0.564	0.786
✓	✓	+0.30M	1.16	0.562	0.788
		M2O DINs			
		+22.30M	4.27	0.563	0.783
✓		+55.78M	9.15	0.562	0.786
✓	✓	+189.57M	28.67	0.565	0.787

NL: Non-Linearity; C: Context. Δ indicates increased parameters compared to the standard FastDepth baseline. GFLOPs are measured on a $224 \times 224 \times 3$ input. The best performance is boldfaced.

TABLE 3.9: Performance of FastDepth [105] on the NYUDv2 Dataset.

	rms	rel	log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Hu *et al.* [47]	0.558	0.129	0.055	0.837	**0.968**	**0.992**
Hu *et al.* + IndexNet	**0.554**	**0.128**	**0.054**	**0.843**	**0.968**	**0.992**

Only HIN ('Nonlinear+Context') is evaluated due to varied feature dimensionality of decoder and multi-level feature fusion. *rel* and *log10* denote the average relative error and average $\log_{10}$ error [109], respectively. The best performance is boldfaced.

TABLE 3.10: Performance of Hu *et al.* [47] on the NYUDv2 dataset

denoising, where M2O DINs exhibit the best performance and the most stable behavior, respectively. Hence, the capacity of IndexNet is closely related to the complexity of local patterns. M2O DINs is preferred in a detail- or boundary-sensitive task, but one should also be aware of the increased model parameters and computation costs, especially when the feature maps are high-dimensional.

4. If one prefers a flexible decoder design, e.g., squeezing/enlarging the dimensionality of the decoder feature map, HINs are good choices, because DINs only generate index maps whose dimensionality is identical to the input feature map.

5. For real-time applications, Shared Stagewise O2O DINs are the first choices. Model parameters increased by Shared Stagewise O2O DINs are comparable to Modelwise O2O DINs, and the extra GFLOPs are also negelectable. Shared Stagewise O2O DINs, however, always work better than Modelwise O2O DINs for applications considered in this work. It implies that each upsampling stage should learn a stage-specific index function;

6. It is worth noting that, the current implementation of IndexNet has some limitations. Currently IndexNet only implements single-point upsampling—each upsampled feature point is only associated with a single point. In this sense, we may not simulate the behavior of bilinear interpolation where each upsampled point is affected by multiple points of a local region.

3.7 Conclusion

Inspired by an observation in image matting, we examine the role of indices and present a unified view of upsampling operators using the notion of index functions. We show that an index function can be learned within a proposed index-guided encoder-decoder framework. In this framework, indices are learned with a flexible network module termed IndexNet, and are used to guide downsampling and upsampling using $\mathcal{IP}$ and $\mathcal{IU}$. IndexNet itself is also a sub-framework that can be designed depending on the task at hand. We investigate five index networks, and demonstrate their effectiveness on four dense prediction tasks. We believe that IndexNet is an important step towards generic upsampling operators for deep networks.

Statement of Authorship

Title of Paper	Learning Affinity-Aware Upsampling for Deep Image Matting
Publication Status	[✓ Published] ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	CVPR, 2021, pp. 6841-6850

Principal Author

Name of Principal Author (Candidate)	Yutong Dai		
Contribution to the Paper	Experiment design and ~~implementation~~, writing draft of the paper		
Overall percentage (%)	70%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	09/12/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Hao Lu		
Contribution to the Paper	Discussion and writing revision		
Signature		Date	09/12/2021

Name of Co-Author	Chunhua Shen		
Contribution to the Paper	Discussion and writing revision		
Signature			09/12/2021

Please cut and paste additional co-author panels here as required.

Chapter 4

Learning Affinity-Aware Upsampling for Deep Image Matting

Based on Chapter 3, in this chapter, we delve deeper into the upsampling stage to investigate upsampling operators conditioned on second-order features. 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^{2}U) 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^{2}U 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^{2}U 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.

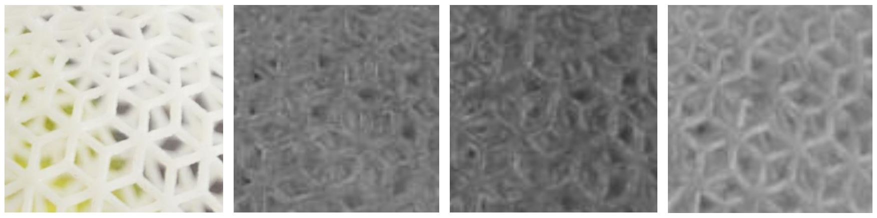


FIGURE 4.1: Visualization of upsampled feature maps with various upsampling operators. From left to right, the input RGB image, feature maps after the last upsampling using nearest neighbor interpolation, bilinear upsampling, and our proposed affinity-aware upsampling, respectively. Our method produces better details with clear connectivity.

4.1 Introduction

The similarity among positions, *a.k.a.* affinity, is commonly investigated in dense prediction tasks [67, 16, 34, 103, 60]. 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 [60] requires a feature encoding block besides the encoder-decoder architecture to learn affinity. The work in [16] needs more iterations to refine the feature maps according to their affinity at the last stage. As shown in Fig. 4.1, 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?*

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 [72, 71, 100]. 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 [100] and IndexNet [71]. 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 motivates 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 ($\mathrm{A}^2\mathrm{U}$). As we show later in Section 4.4, *A^2U is a generalization of first-order upsampling operators*: in some conditions, the first-order formulation in [100] and [72] can be viewed as special cases of our second-order one. In addition, by implementing $\mathrm{A}^2\mathrm{U}$ in a low-rank bilinear formulation, we can achieve efficient upsampling with few extra parameters.

We demonstrate the effectiveness of $\mathrm{A}^2\mathrm{U}$ 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 $\mathrm{A}^2\mathrm{U}$ and compare their similarities and differences. Compared with a strong image matting baseline on the Composition-1k matting dataset, $\mathrm{A}^2\mathrm{U}$ 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.

4.2 Background

Learning-Based Upsampling Operators. Detailed review on upsampling has been demonstrated in Chapter 3. Here we focus on learning-based upsampling operators [90, 69, 100, 71]. The Pixel Shuffle (P.S.) [90] upsamples feature maps by reshaping. The deconvolution (Deconv) [69], 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 [100] and IndexNet [72] 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.

Affinity-Based Image Matting. Affinity dominates the majority of classic image matting approaches [57, 11, 19, 38], as presented in Chapter 2. Such methods can perform well on cases with clear color contrast but more often fail in cases where the color distribution assumption is violated.

Within deep matting methods, GCA matting [60] 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.

4.3 A Mathematical View of Upsampling

The work in Chapter 3 [71] 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 upsampling, most existing upsampling operations follow

$$g(\mathbf{w}, \mathbf{z}) = \mathbf{w}^T\mathbf{z}\,. \tag{4.1}$$

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 [69]. 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

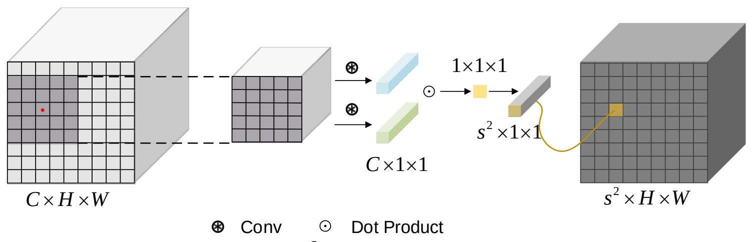


FIGURE 4.2: Kernel generation of A^{2}U. Given a feature map of size $C \times H \times W$, an $s \times s$ upsampling kernel is generated at each spatial position conditioned on the feature map. The rank d is 1 here.

Equation (4.1), 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 [3], CARAFE [100], and IndexNet [71]:

- *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.
- *CARAFE* learns an upsampling kernel $\mathbf{w} \in \mathbb{R}^{k^2\times 1}$ ($k = 5$ in [100]) via a kernel generation module given a decoder feature map ready to upsample. It also conforms to Equation (4.1), 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.
- *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 Equation (4.1), 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.*

4.4 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 Equation (4.1), 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 $sim\left(\mathbf{m}_l, \mathbf{m}_p\right)$ 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 [100, 72, 71] 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.*,

$$\begin{aligned} w &= \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}\varphi(\mathbf{x}_i)^T\phi(\mathbf{y}_j) = \sum_{k=1}^{C}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}q_k x_{ik}t_k y_{jk} \\ &= \sum_{k=1}^{C}\sum_{i=1}^{N}\sum_{j=1}^{M} a'_{ijk}x_{ik}y_{jk} = \sum_{k=1}^{C} {\mathbf{x}_k}^T\mathbf{A}_k\mathbf{y}_k\,, \end{aligned} \tag{4.2}$$

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 [51, 118], a low-rank bilinear method can be derived to reduce computational complexity of Equation (4.2). 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 $d \le \min(N, M)$. Equation (4.2) therefore can be rewritten by

$$\begin{aligned} w &= \sum_{k=1}^{C} \mathbf{x}_k{}^T\mathbf{U}_k\mathbf{V}_k{}^T\mathbf{y}_k = \sum_{k=1}^{C} \mathbb{1}^T(\mathbf{U}_k{}^T\mathbf{x}_k \circ \mathbf{V}_k{}^T\mathbf{y}_k) \\ &= \mathbb{1}^T\sum_{k=1}^{C}(\mathbf{U}_k{}^T\mathbf{x}_k \circ \mathbf{V}_k{}^T\mathbf{y}_k) \end{aligned}, \tag{4.3}$$

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 Equation (4.3) can be replaced with $\mathbf{P} \in \mathbb{R}^{d\times s^2}$. Note that, Equation (4.3) 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

$$\begin{aligned} \mathbf{w} &= \mathbf{P}^T\sum_{k=1}^{C}(\mathbf{U}_k{}^T\mathbf{x}_k \circ \mathbf{V}_k{}^T\mathbf{y}_k) \\ &= \mathbf{P}^T\underset{r=1}{\overset{d}{\mathtt{cat}}}\Big(\sum_{k=1}^{C}(\mathbf{u}_{kr}{}^T\mathbf{x}_k \circ \mathbf{v}_{kr}{}^T\mathbf{y}_k)\Big) \\ &= \mathtt{conv}\Big(\mathcal{P}, \underset{r=1}{\overset{d}{\mathtt{cat}}}\big(\mathtt{gpconv}(\mathcal{U}_r, \mathcal{X}) \odot \mathtt{gpconv}(\mathcal{V}_r, \mathcal{Y})\big)\Big) \end{aligned}, \tag{4.4}$$

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. $\mathtt{conv}(\mathcal{K}, \mathcal{M})$ represents a convolution operation on the feature map $\mathcal{M}$ with the kernel $\mathcal{K}$; $\mathtt{gpconv}(\mathcal{K}, \mathcal{M})$ defines a group convolution operation (C groups) with the same input. `cat` is the concatenate operator. This process is visualized in Fig. 4.2.

Alternative Implementations. Equation. (4.4) is a generic formulation. In practice, many design choices can be discussed in implementation:

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

- $\mathcal{U}$ 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*').
- Equation. (4.4) 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 [115], 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.
- 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 [51], 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.

Extension to Downsampling. Following [71], 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 Equation. (4.4) 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 $s_d = rs_u$ 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.

1. *Joint Bilateral Upsampling (JBU)* [53]. 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.

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 [69]	31.85	0.9942	0.0256	0.0089	27.42	0.9870	0.0426	0.0207
P.S. [90]	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 [100]	28.72	0.9885	0.0367	0.0131	25.17	0.9773	0.0552	0.0266
MaxPool-IndexNet † [72]	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 ‡ [72]	47.13	0.9997	0.0044	0.0020	44.35	0.9998	0.0061	0.0036

TABLE 4.1: 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.

2. *Guided Filter (GF)* [40]. 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) [108] 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.
3. *Attention Networks* [99, 46]. Attention networks include a wide family of networks applying the attention mechanism. As discussed in [72, 100], 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.

4.5 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 [71], we build sets of reconstruction experiments on the MNIST dataset [55] and Fashion-MNIST dataset [110]. 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 [71]. 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

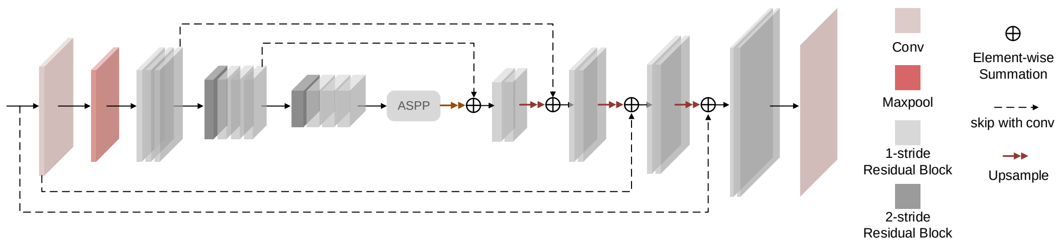


FIGURE 4.3: Overview of our matting framework. The focus of this work is on the upsampling stages.

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 4.1.

As shown in Table 4.1, 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.

4.6 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 [60], our baseline network adopts the first 11 layers of the ResNet34 [41] as the encoder. The decoder consists of residual blocks and upsampling stages. The In-Place Activated BatchNorm [83] is applied to each layer except the last one to reduce GPU memory consumption during the training stage. As shown in Fig. 4.3, the overall network follows the UNet architecture [82] 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.

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 4.2: 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.

Datasets. We mainly conduct our experiments on the Adobe Image Matting dataset [113]. Instead of compositing each foreground with fixed 100 background images chosen from MS COCO [65], we randomly choose the background images in each iteration and generate the composition images on-the-fly.

We also evaluate our method on the `alphamatting.com` benchmark [81]. 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 [80]. 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. 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 [60]. 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 [65]. After imposing random jitters to the foreground object, the RGB image is finally generated by composition.

The backbone is pretrained on ImageNet [26]. Adam optimizer [52] is used. We use the same loss function as [113, 72], including alpha prediction loss and composition loss computed from the unknown regions indicated by trimaps. We update parameters for 30

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

TABLE 4.3: Ablation study of upsampling kernel size on the Composition-1k test set.

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) [81].

4.6.1 The Adobe Image Matting Dataset

Ablation Study on Alternative Implementations. Here we verify different implementations of A^2U on the Composition-1k test set and compare them with existing upsampling operators. Quantitative results are shown in Table 4.2. 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 [100], *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 [71]. 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 [72, 71]. 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^2U 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

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

Table 4.4: Ablation study of normalization on the Composition-1k test set.

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.

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 4.3 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 [100] and [71], different normalization strategies are verified, and experiments show that normalization significantly affects the results. We thus justify the normalization choices in our A^2U 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 4.4). 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 4.5. 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 [60], 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. 4.4. Our results show improved detail delineation such as the net structure and the filament.

Method	SAD	MSE	Grad	Conn	# Params
CF [57]	168.1	0.091	126.9	167.9	-
KNN [11]	175.4	0.103	124.1	176.4	-
DIM [113]	50.4	0.014	31.0	50.8	$>$ 130.55M
IndexNet [72]	45.8	0.013	25.9	43.7	8.15M
Ada [4]	41.7	0.010	16.8	-	-
CA [43]	35.8	**0.0082**	17.3	33.2	107.5M
GCA [60]	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 4.5: Benchmark results on the Composition-1k test set. The best performance is in boldface.

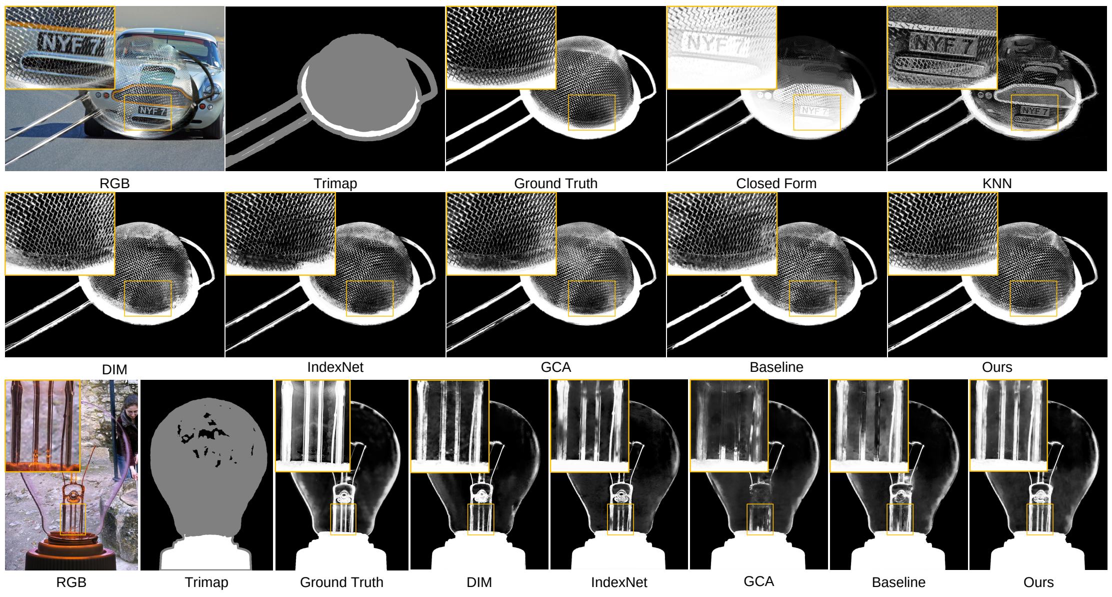


FIGURE 4.4: Qualitative results on the Composition-1k test set. The methods in comparison include Closed-Form Matting (CF) [57], KNN Matting [11], Deep Image Matting (DIM) [113], IndexNet Matting [72], GCA Matting [60], our baseline, and our method.

Gradient Error	Average Rank				Troll		Doll		Donkey		Elephant		Plant		Pineapple		Plastic bag		Net	
	Overall	S	L	U	L	U	L	U	L	U	L	U	L	U	L	U	L	U	L	U
Ours	**6.3**	5.6	**3.3**	10.1	0.2	0.2	**0.1**	0.2	**0.2**	**0.2**	0.2	0.4	**1.3**	1.9	0.7	1.7	**0.6**	0.6	**0.3**	0.4
Ada [4]	7.8	**4.5**	5.6	13.3	0.2	0.2	0.1	0.4	0.2	0.2	**0.1**	0.3	1.4	2.3	**0.6**	**0.9**	1	0.9	0.4	**0.4**
GCA [60]	8	8.4	6.6	**9.1**	**0.1**	0.2	0.1	0.3	0.2	0.2	0.2	**0.3**	1.6	1.9	0.8	1.4	0.7	**0.6**	0.4	0.4
CA [43]	9.1	10.8	9.8	6.8	0.2	**0.2**	0.2	**0.2**	0.2	0.2	0.4	0.4	1.5	**1.8**	1.3	1	1.1	0.9	0.4	0.4

TABLE 4.6: Gradient errors on the `alphamatting.com` test set. The top-4 methods are shown. The lowest errors are in boldface.

4.6.2 The `alphamatting.com` Benchmark

Here we report results on the `alphamatting.com` benchmark [81]. We train our model with all the data in the Adobe Matting dataset and test it on the benchmark. As shown in Table 4.6, our method ranks the first w.r.t. the gradient error among all published methods. We also achieve comparable overall ranking compared with AdaMatting [4]

Method	SAD	MSE	Grad	Conn
CF* [57]	105.73	0.023	91.76	114.55
KNN* [11]	116.68	0.025	103.15	121.45
DIM* [113]	47.56	0.009	43.29	55.90
Baseline-Nearest	25.03	0.0106	13.85	24.41
A^2U (hybrid-cw)	24.08	0.0104	13.53	23.59
A^2U (dynamic-cs)	24.55	0.0107	14.51	23.89
A^2U (dynamic-cs-d)	**23.20**	**0.0102**	**12.39**	**22.20**

TABLE 4.7: Benchmark results on the Distinctions-646 test set. The best performance is in boldface. * denotes results cited from [80].

under the SAD and MSE metrics, suggesting our method is one of the top performing methods on this benchmark.

4.6.3 The Distinction-646 Dataset

We also evaluate our method on the recent Distinction-646 test set. In Table 4.7, we report results of the three models performing the best on the Composition-1k dataset and also compare with other benchmarking results provided by [80]. We have two observations: 1) our models show improved performance against the baseline, which further confirms the effectiveness of our A^2U; 2) Our models outperform other reported benchmarking results by large margins, setting a new state of the art on this dataset.

4.7 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^2U can achieve better performance on both image reconstruction and image matting tasks. We also investigate different design choices of A^2U. Results on three image matting benchmarks all show that A^2U 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^2U to other dense prediction tasks.

4.8 Appendix

4.8.1 Network and Training Details of Image Reconstruction

We denote $C(k)$ to be a convolution layer with k-channel output and 3×3 filters (stride is 1 unless stated), followed by BatchNorm and ReLU, and denote D_r a downsampling operator with a ratio of r, and denote U_r an upsampling operator with a ratio of r. We build the network architecture as: $C(32)$-D_2-$C(64)$-D_2-$C(128)$-D_2-$C(256)$-$C(128)$-U_2-$C(64)$-U_2-$C(32)$-U_2-$C(1)$.

The image reconstruction experiments are implemented on the MNIST dataset [55] and Fashion-MNIST dataset [110]. They both include $60,000$ training images and $10,000$ test images. During training, the input images are resized to 32×32, and ℓ_1 loss is used. We use the SGD optimizer with an initial learning rate of 0.01. The learning rate is decreased by $\times 10$ at the 50-th, 70-th, and 85-th epoch, respectively. We update the parameters for 100 epochs in total with a batch size of 100. The evaluation metrics are Peak Signal-to-Noise Ratio (PSNR), Structural SIMilarity (SSIM), Mean Absolute Error (MAE) and root Mean Square Error (MSE).

4.8.2 Analysis of Complexity

Here we summarize the model complexity of different implementations of A^{2}U in Table 4.8. We assume that the encoding kernel size is $k \times k$, the upsampling kernel size is $s \times s$, and the channel number of feature map $\mathcal{X}$ is C. Since C is much larger than k and s, A^{2}U generally has the complexity: *dynamic cw* > *hybrid cw* > *static cw* > *dynamic cs* > *hybrid cs* > *static cs*.

Model	Type	# Params
static	cw	$4 \times s \times s + 2 \times k \times k \times C$
static	cs	$4 \times s \times s + 2 \times k \times k$
hybrid	cw	$4 \times s \times s \times C + 2 \times k \times k \times C$
hybrid	cs	$4 \times s \times s \times C + 2 \times k \times k$
dynamic	cw	$4 \times s \times s \times C + 2 \times C \times C$
dynamic	cs	$4 \times s \times s \times C + 2 \times C$

TABLE 4.8: Analysis on the complexity of A^{2}U. 'cw': channel-wise, 'cs': channel-shared

4.8.3 Qualitative Results

We show additional qualitative results on the `alphamatting.com` benchmark [81] in Fig. 4.5. 4 top-performing methods are visualized here. Since all these methods achieve

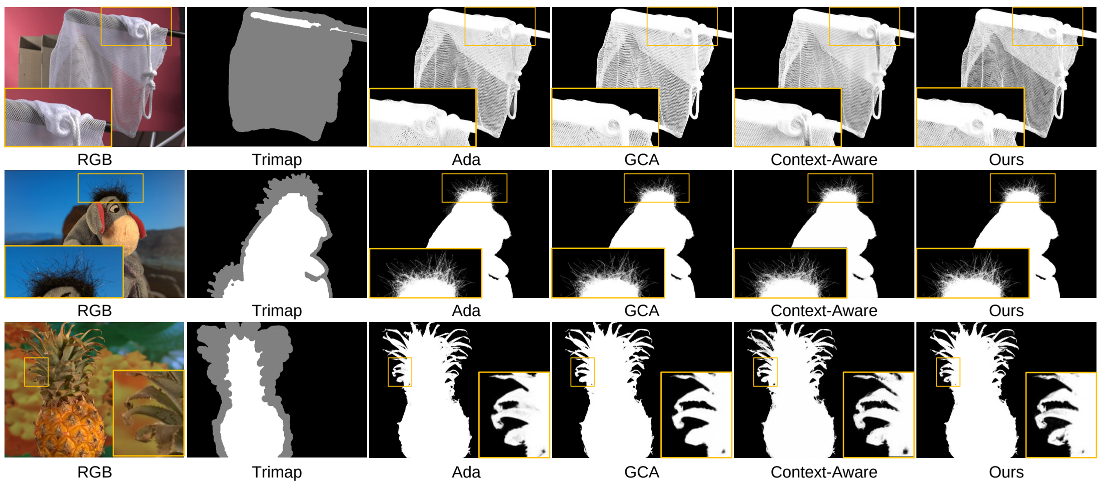


FIGURE 4.5: Qualitative results on the `alphamatting.com` test set. The methods in comparison include AdaMatting [4], GCA [60], Context-Aware [43], and our method.

good performance, and their quantitative results on the benchmark are very close, it is difficult to tell the obvious difference in Fig. 4.5. It worth noting that, however, our method produces better visual results on detailed structures, such as gridding of the net, and leaves of the pineapple.

We also show qualitative results on the Distinction-646 test set [80] in Fig. 4.6. Since no implementation of other deep methods on this benchmark is publicly available, we only present the results of our baseline and our method here to show the relative improvements. According to Fig. 4.6, our method produces clearly better predictions on highly transparent objects such as the bubbles.

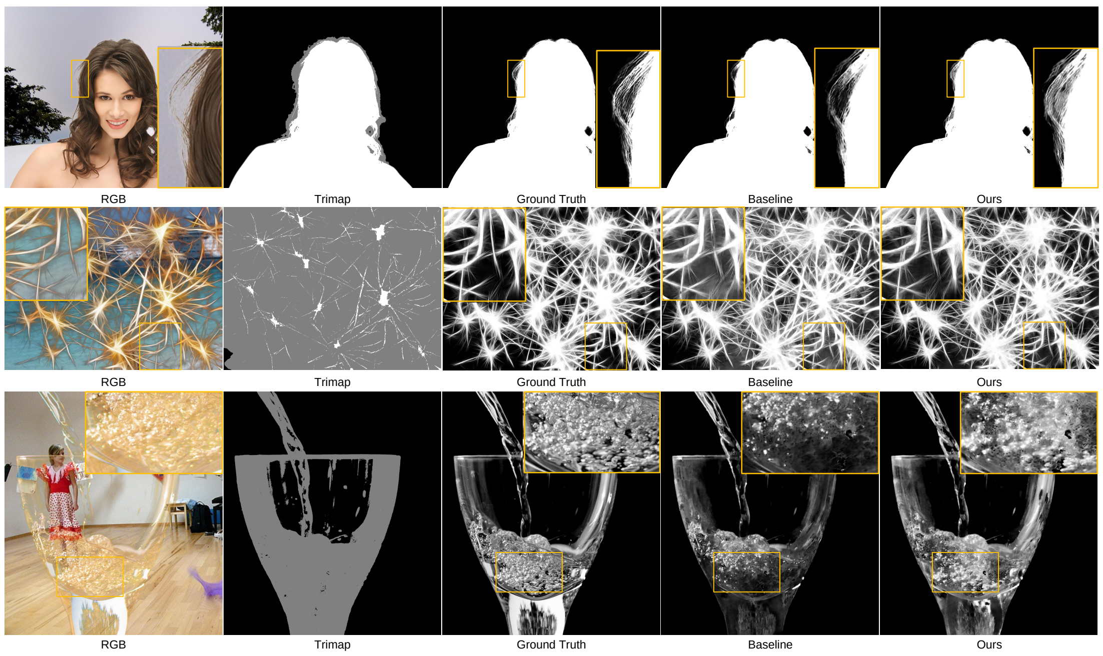


FIGURE 4.6: Qualitative results on the Distinction-646 test set. The methods in comparison include the baseline and our method.

Statement of Authorship

Title of Paper	Boosting Robustness of Image Matting with Context Assembling and Strong Data Augmentation
Publication Status	☐ Published ☐ Accepted for Publication [✓ Submitted for Publication] ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	submitted to cvpr 2022

Principal Author

Name of Principal Author (Candidate)	Yutong Dai		
Contribution to the Paper	Experiment framework design and implementation, experiment conduction, writing draft of the paper.		
Overall percentage (%)	70%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	18/11/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Brian Price		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	Nov 18, 2021

Name of Co-Author	He Zhang		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	Nov.18.2021

Name of Co-Author	Chunhua Shen		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	14/12/2021

Chapter 5

Boosting Robustness of Image Matting with Context Assembling and Strong Data Augmentation

Chapter 3 and chapter 4 consider indices and affinity properties of matting, respectively, and incorporate them into the upsampling stages. They promise efficient models with competitive results on benchmarks. Similarly, their concurrent deep image matting methods have achieved increasingly better results on benchmarks (e.g., Composition-1k/`alphamatting.com`). However, the robustness, including robustness to trimaps and generalization to images from different domains, is still under-explored. To fill this gap, in this chapter, we propose an image matting method that achieves higher robustness (RMat) via multilevel context assembling and strong data augmentation targeting matting. Specifically, we first build a strong matting framework by modeling ample global information with transformer blocks in the encoder and focusing on details in combination with convolution layers as well as a low-level feature assembling attention block in the decoder. Then, based on this strong baseline, we analyze current data augmentation and explore simple but effective strong data augmentation to boost the baseline model and contribute a more generalizable matting method. Compared with previous methods, the proposed method not only achieves state-of-the-art results on the Composition-1k benchmark (11% improvement on SAD and 27% improvement on Grad) with a smaller model size, but also shows more robust generalization results on other benchmarks, on real-world images, and also on varying coarse-to-fine trimaps with our extensive experiments.

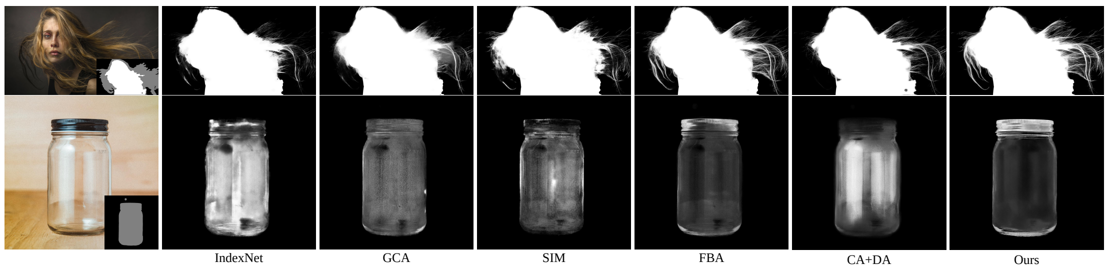


FIGURE 5.1: Matting results on real-world images. From the second column to right are results of IndexNet [72], GCA [60], SIM [97], FBA [32], CA+data augmentation ($\mathcal{DA}$) [43] and our method, respectively. Note that, all the methods are trained with the DIM [113] dataset (except SIM is trained with the SIMD [97] dataset). They are comparable on benchmark images, while presents varying results on real-world images. Our method shows better generalization ability.

5.1 Introduction

With recent success of deep learning, deep matting methods [113, 72, 43, 60, 22, 97, 32] achieve promising results on benchmarks such as Composition-1k [113] and `alphamatting.com` [81]. While increasingly higher accuracy have been promised on benchmarks, due to the limited training/test data, robustness of these methods is still under explored.

First, robustness to the trimap is important for a matting algorithm. In real applications, trimaps are labeled by users, with unpredictable precision of unknown regions. However, as shown in Fig. 5.2, existing matting methods [97, 32] are sensitive to the shape/size of the given trimap so that it requires users' more time to accurately brush the trimap. A main reason why existing methods are sensitive to the precision of trimap is they focus more on detailed cues, where robustness to trimap with varing precision, which relies more on context information, is less cared about. One possible solution is to optimize the trimap to be a more detailed one. This was proposed in [4], where an extra branch was used to generate a more precise trimap. Though multi-task learning is leveraged in this method to adapt the trimap, its context modeling is still limited, which restricts its robustness in applications. Therefore, we wonder *whether it is possible to enhance the context modeling ability (robustness) of a matting algorithm with a simpler and more effective approach.*

Meanwhile, it has been known that deep matting models trained on synthetic data undertake the risk of poor generation to real-world domains [43, 97, 119] (Fig. 5.1). However, due to the difficulty of obtaining ground-truth alpha matte annotations for real-world images, only synthetic datasets are available to train the matting algorithms, so some works attempted to narrow the domain gap. For example, [43, 119] leverage extra data augmentation to adapt the models to real-world images, while significant performance degradation on the synthetic benchmark happens at the same time. Although better prediction on real-world images is appreciated, it is desirable that the model can be generalized to broader scenes without sacrificing too much performance on images from one domain such as the benchmark data, because it is hard to confirm which domain a test image comes from, and not to mention that the real-world test images in [43, 119] can only cover a tiny part of real scenes. Therefore, *a model showing better domain generalization ability is in demand.*

Motivated by these demands, we present a more robust matting method (RMat), which achieves higher robustness to diverse trimap precision and better generalization to various domains. In detail, two steps are designed. The first step is to build a strong baseline model with multilevel context assembling. It is implemented by combining transformer blocks with convolution layers, where global context is learned via self-attention modules and local context is emphasized by convolution layers. Considering the uniqueness of matting that needs local context information and original test resolutions to capture details, we explore designs and implementations aiming at this task to build an efficient model. Further, founded on this strong baseline model, we investigate strong data augmentation for matting. We analyze the problems behind current augmentation and propose strong augmentation strategies specifically for matting. Finally, to verify robustness of the model, a series of experiments and visualizations are carried out in comparison with state-of-the-art methods.

In summary, our main contributions are: **1)** A strong matting framework with multilevel context assembling; **2)** Strong augmentation strategies targeting matting; **3)** Designs of experiments and visualizations to verify generalization capability of matting models; **4)** State-of-the-art results on benchmarks (w/ and w/o fitting the training sets), higher robustness to varying trimap precision, and better generalization to real-world images.

5.2 Background

Context in Deep Image Matting. Among recent state-of-the-art deep matting methods [43, 60, 72, 22], context and dynamic networks are two vital and correlated components. The context includes both global context and local context. Global context intuitively benefits better recognition of the foreground object. It motivates studies on extra context learning modules [43, 60, 68]. It is also one of the reasons behind using ASPP or PPM module in recent methods [72, 32, 22, 97]. Local context instead promotes detail capture by caring about correlations within a local region. The convolution operations or dynamic kernels learned from local regions [72, 22] model local context

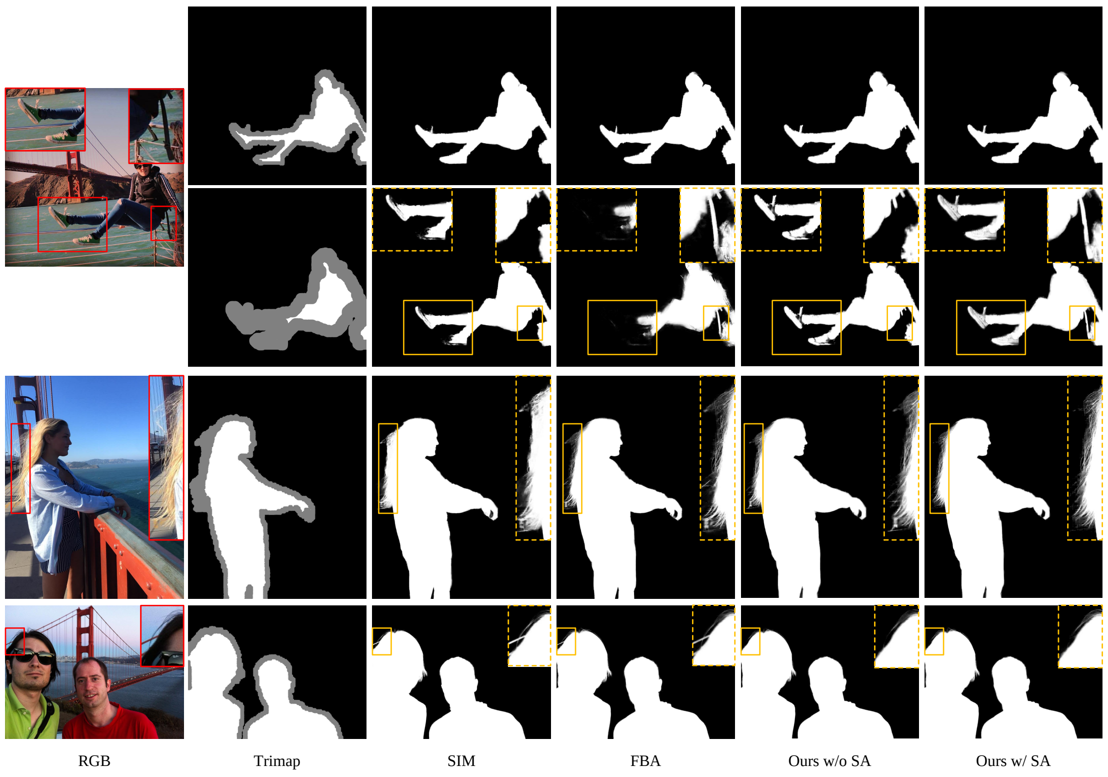


FIGURE 5.2: Visual results on real-world images showing robustness of methods. The methods in comparison are SIM [97], FBA [32], and our method w/o and w/ Strong Augmentation ($\mathcal{SA}$), respectively. The first two rows are results of the same RGB image with different trimaps. Our methods are more robust to various trimap precision. In the third row, our model using $\mathcal{SA}$ captures better details. The last row presents the benefit of modeling global context: the bridge component on the left top side is apart from the main body of the bridge, which is recognized as foreground in SIM and FBA. Our method, however, distinguishes it from the foreground human clearly thanks to the context assembling. Best viewed by zooming in.

into the network. On another side, dynamic networks were introduced to matting [72, 22, 60] to enlarge the model capacity. They also benefit the network in combination with the context assembling [72].

Since we aim for a more robust matting method, which needs multilevel context information as well as ample model capacity, the first step is taking both context and dynamic networks into consideration efficiently. We show that it is achievable by combining transformer blocks and convolution layers. We investigate various designs and also provide our insights into them. Considering the lack of training data and a relative large capacity of our model, we study strong data augmentation strategies to prevent overfitting the training data and also generalize the model better.

Domain Generalization. Domain generalization aims at learning better representations that can be transferred to unseen domains. There are many potential solutions, such as data augmentation [126, 114, 121], meta learning [31, 58], and adversarial training [33, 73]. In deep matting, since only synthetic training data is available, the trained models usually suffer from poor generalization. They may work well on specific domains, such as the synthetic ones similar to the training set, but show obvious decreasing performance when applied to another domain, as the examples in Fig. 5.1. Extra data augmentation [43, 119] has been applied to adapt models to real-world images, but they consider limited cases only, such as the resolution gap between the foreground object and background, which may bias the models to those images. We may observe it from Table 5.5.

Therefore, we move a step further by rethinking strong data augmentation for matting. We first analyze why current extra augmentation deteriorates the benchmark performance, then propose strong augmentation strategies targeting matting. Our goal is to prevent the model overfitting the synthetic training data and help them generalize better to real-world images.

5.3 A Strong Matting Framework with Context Assembling

As noted in conventional sampling-based matting [101, 88] and propagation-based matting [57, 11], both nearby and long-distance pixels contribute to alpha prediction depending on their correlations. In deep models, the correlations are related to context. Existing deep matting methods attempt to model contextual attention [60] or extra context information [43] in the network, while the global context is still under explored. This may limit their performance on complex images such as Fig. 5.2. In order to assemble multilevel context information, including global context, we build a baseline combining transformer blocks and convolution layers. Designs of the framework are detailed below:

Encoder Design. As shown in Fig. 5.3, the encoder has two branches: a transformer-based branch modeling global context and a convolution-based branch supplementing low-level information for details. Driven by recent vision transformers [28, 125, 102, 112], we use a 32-stride pyramid vision transformer backbone to obtain hierarchical features. Since matting models do inference with various original input resolutions, fixed position embedding is unsuitable for the application. We therefore take advantages of [112], where fixed position embedding is replaced with overlapped convolutions. Due to the large capacity of the transformer blocks, only 2-stride convolution layers are used in the convolution-based branch to form 8-stride. We use two small backbones in [112] (mit-b1 and mit-b2) because of the limited training data for matting. Finally, two encoder architectures with different capacities (E1, E2) are built. BiseNetV2 [117] also uses multiple branches in the encoder for segmentation. Different from our purpose on

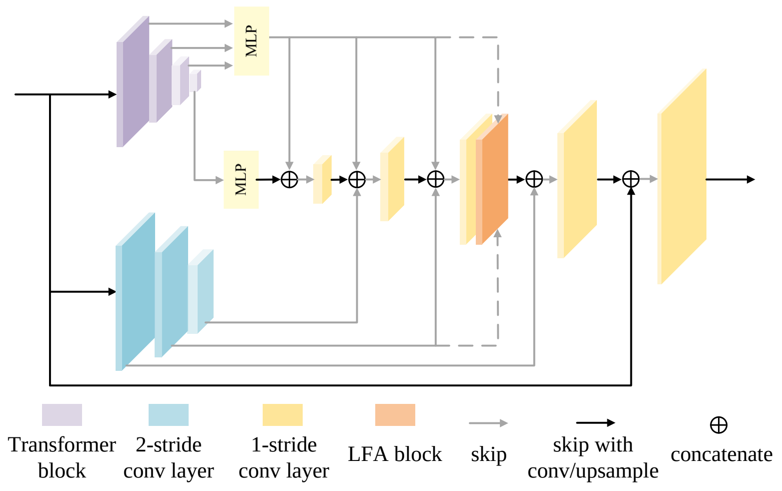


FIGURE 5.3: Model architecture of our framework.

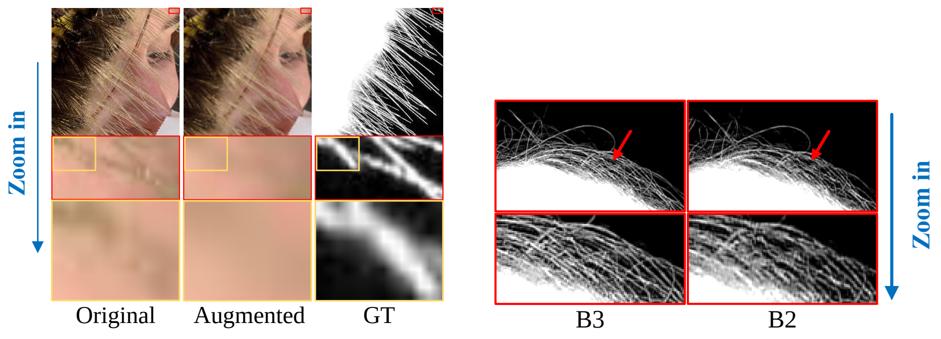
recovering missing details, [117] aims to combine high-level and low-level information in the encoding stage, to balance accuracy and efficiency.

Decoder Design. Various decoder designs [4, 23, 119, 60, 72, 22] have been studied in matting models. As the bridge to recover resolutions and capture details, the decoder matters for matting. For instance, previous methods applied feature skip [72], attention-guided refinement [60] or dynamic upsampling [22] to build functional matting decoders aiming at richer details. As the first matting method applying transformers, and considering the importance of the decoder, we investigate an efficient decoder design for our framework.

In general, options for a decoder, in order of decreasing receptive field size, include transformer layers, convolution layers, and MLP layers. Since the transformer branch in the encoder promises a large capacity and global reception field, and to reduce computation as well, we only consider using MLP layers and convolution layers in the basic decoder. These also work well to combining multilevel context information. As a result, several baseline models with different decoders are investigated as listed in Table 5.2.

Feature Skip Design. Skip information from encoder to decoder has been widely adopted in deep matting methods [72, 60, 32, 97]. We categorize the skip information into two sources: **1)** the transformer branch of the encoder (TSkip), where feature maps with different resolutions are skipped to the decoder after MLP/convolution layers. These feature maps transport abundant global information while recovering the resolution. Since the transformer branch starts from $\frac{1}{4}$ resolution, some details may be missing at the initial downsampling stage, so we use **2)** another source of skip information learned in the convolution branch (LSkip).

Low-Level Feature Assembling Attention Block (LFA) Design. Inspired by [60, 22], where low-level feature maps assist on refining decoder features, we explore efficient



(a) (b)

FIGURE 5.4: (a) A comparison between images before and after $\mathcal{DA}$ (Re-JPEG, Gaussian Blur). The augmented image loses its structure and does not match the ground truth. (b) A comparison between B3 and B2 (Table 5.2). The hairs in B2 are blurred.

low-level feature assembling using a transformer block. It can be easily extended from the transformer block in the encoder . Let $Attn(Q, K, V)$ denotes the self-attention operation in the transformer block, the feature fusion attention then can be represent by $Attn(f_{low}, f_{low}, f_d)$, where f_{low} is the skipped feature from the encoder and f_d is the feature in the decoder to be refined. Only one LFA block is added after the $\frac{1}{4}$ resolution decoder layer as noted in Fig. 5.3 in our experiments to restrict the computation. We observe introducing this block further improves the accuracy.

5.4 Domain Generalization and Data Augmentation for Image Matting

As training images for matting are created using composition, it inevitably results in generalization problems on real-world images. Also, it is noticed that large-capacity transformer-based models may encounter the overfitting problem [107, 25], especially when the dataset is small. Matting datasets [113, 80, 97], unfortunately, have limited sizes. They usually use only hundreds of foreground images to generate tens of thousands of synthetic training data, so overfitting is a potential problem. To handle this issue, we study strong augmentation ($\mathcal{SA}$) for better generalization.

Targeting the domain gap between synthetic data and real-world images, extra data augmentation ($\mathcal{DA}$) was proposed [43, 119]. It mainly includes Re-JEPG and Gaussian blur. Experiments in [43, 119] show $\mathcal{DA}$ improves results on real-world images, but deteriorates the performance on the benchmark significantly, as shown in Table 5.4 (CA vs. CA+$\mathcal{DA}$). Therefore, $\mathcal{SA}$ firstly needs to overcome the performance degradation on the benchmark.

$\mathcal{DA}$	Modified α	SAD	Grad
		32.65	18.13
✓		35.79	20.16
✓	✓	34.00	20.12

TABLE 5.1: $\mathcal{DA}$ using different ground truths. This toy experiment is trained with a batch size of 32, 45k iterations.

5.4.1 Rethinking Domain Generalization and Gaps

Why Current Extra Data Augmentation ($\mathcal{DA}$) Deteriorates Performance on Benchmarks. An example of using Re-JPEG and Gaussian blur is shown in Fig. 5.4a. As observed, some background pixels mix values with foreground pixels after augmentation. Their alpha values therefore change from 0 to a value in range $(0, 1)$. This kind of alpha value blending also exists in transparent regions and in some foreground pixels close to the background. In previous works [43, 119], however, the same alpha ground truths are used after $\mathcal{DA}$, which violates the matting equation. The image after augmentation loses the structure and does not match the alpha ground truth any more. Using these image-alpha pairs for training could mislead the network to wrong predictions. Hence, we argue it is at least one of the main reasons behind the performance deterioration. To verify this assumption, we carry out a toy-level experiment on DIM. By using Gaussian blur and Re-JPEG with the possibility 0.25 for each as $\mathcal{DA}$, two models are trained: 1) a model trained with $\mathcal{DA}$ using original alpha ground truths; 2) a model trained with $\mathcal{DA}$ using modified alpha ground truths generated by applying the same augmentation as was applied to the RGB image. To ease the difficulty, only L1 alpha prediction loss is applied and fewer training iterations are used. Other training details match the main experiments, as detailed in Section. 5.5. As present in Table 5.1, $\mathcal{DA}$ makes the errors higher, and adjusting the ground truth slightly relieves the problem. Hence, it is at least reasonable to claim that modification of ground truth matters for using $\mathcal{DA}$. The correct ground truth, however, is hard to obtain.

What are the Domain Gaps for Matting. During the data loading stage, we assume the composition process satisfies the linear Equation:

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i \,. \tag{5.1}$$

As a result, no matter how the foreground and the background have been processed individually, the image still satisfies the linear equation after the composition. Under this assumption, what are the main domain gaps for matting?

1) *Complexity of Surrounding Context.* For example, the third example in Fig. 5.2 is a challenging because it is rarely found in the synthetic datasets. **2)** *Source of Images.* In real-world images, foreground and background are from the same source, while this condition is not met in synthetic data. There could be many differences between them: brightness, saturation, sharpness, noise level, etc. **3)** *Manual Operations During Photography and Modifications Made to the Images.* For instance, unfocused boundaries, blurred regions, mosaic generated by image compression, etc. They rarely exist in synthetic datasets.

In this work, we rely on the network to deal with the context domain gap by assembling multilevel context information. As for remaining feature-level gaps, we investigate simple but efficient strong augmentation strategies to generalize the algorithm to real-world images better.

5.4.2 Strong Data Augmentation for Matting

Driven by above analysis, we study $\mathcal{SA}$ for matting. The augmentations are divided into three categories:

1) ***Linear Pixel-Wise Augmentation.*** By pixel-wise, we mean no interpolation happens on the image. Linear denotes the operations that can be linearly represented. It includes linear contrast, brightness adjustment, noise, etc. Only pixel-level changes happen without any information exchange among different pixels and even different channels. If we look at one channel of location i on image I, it can be formulated by:

$$\begin{aligned} I'_i &= aI_i + b \\ &= a\left[\alpha F_i + (1-\alpha) B_i\right] + \left[\alpha b + (1-\alpha) b\right], \\ &= \alpha\left(aF_i + b\right) + (1-\alpha)\left(aB_i + b\right) \end{aligned} \tag{5.2}$$

where a and b are constant parameters for the linear transformation. According to this equation, linear pixel-wise augmentation obeys Equation (5.1) no matter it happens on I_i, F_i or B_i. Augmentation on an image can also be viewed as processing the foreground and background individually. It is natural to extend this equation by:

$$I'_i = \alpha\left(aF_i + b\right) + (1-\alpha)\left(mB_i + n\right) , \tag{5.3}$$

where different linear transformations happen on the foreground and the background.

2) ***Nonlinear Pixel-Wise Augmentation.*** In the opposite to linear operations, there are also non-linear augmentations, such as gamma correction, hue/saturation adjustment, etc. Due to their nonlinear nature, Equation (5.1) is violated if the augmentations happen on I.

3) ***Region-Wise Augmentation.*** Region-Wise augmentation means operations applied using multiple pixels. For instance, blur, jpeg compression, etc. After interpolations on I, Equation (5.1) is violated, which needs alpha ground truth to be modified accordingly.

Based on this categorization, we propose strong data augmentation strategies:

i) **Augment the Foreground Alone (AF).** Motivated by the random jitter in [60], augmenting foreground alone is effective and obeys the composition equation. The ground truth does not need to be modified no matter which augmentation is taken because it happens before composition.

ii) **Augment the Foreground and the Background Individually (AFB).** It is an extended version of option i) and inspired by Equation (5.3). Through augmenting foreground and background individually before composition, the linear composition equation is still satisfied, the ground truth alpha matte hence does not need to be modified no matter which augmentation is taken.

iii) **Augment the Composited Image (AC).** This strategy can be further divided into two sub types. If linear pixel-wise augmentation is applied, the composition equation is satisfied as Equation (5.3). Using other strategies instead violates the equation, where a new ground truth is needed. Due to the expense of obtaining the real ground truth, we propose to generate pseudo label to facilitate the training. The strategy is to predict the pseudo label using the parameters from the last training iteration by rotating or channel-shuffling the input to generate a new training sample. We anticipate this operation promotes the network to learn features of the augmented images without sacrificing accuracy.

5.5 Experiments and Discussions

5.5.1 Implementation Details

Our models are trained on the Adobe Image Matting dataset [113] only. We generate the training samples using background images randomly selected from MS COCO [65], and use the same rules as [113] to produce test images using background images selected from Pascal VOC [29]. The evaluation metrics are commonly-used Sum of Absolute Differences (SAD), Mean Squared Error (MSE), Gradient (Grad) error, and Connectivity (Conn) error. Implementation of [113] is used.

Training Details. Our baseline models follow the dataloader pipeline in [60]. To be specific, the 4-channel input concatenates the RGB image and the trimap. The RGB

No.	Encoder	Decoder	TSkip	LSkip	#Params	SAD(↓)	MSE(↓)	Grad(↓)	Conn(↓)
B1	E1	MLP	MLP		13.6M	39.55	0.0102	24.22	37.35
B2	E1	Conv	MLP		15.4M	29.85	0.0063	13.04	25.61
B3	E1	Conv	MLP	✓	15.8M	28.94	0.0055	12.75	24.66
B4	E1	Conv	MLPConv	✓	18.2M	29.99	0.0064	15.98	25.86
B5	E1	MLPDW	MLP	✓	14.0M	29.79	0.0062	14.18	25.78
B6	E1	MLPDW	MLPConv	✓	16.1M	31.45	0.0065	15.59	27.65
B7	E2	Conv	MLP	✓	26.8M	26.11	0.0048	10.59	21.38
B8	E2	Conv	MLPConv	✓	29.2M	25.66	0.0045	10.40	20.90
B9	E2	MLPDW	MLP	✓	25.1M	28.42	0.0055	12.98	24.15
B10	E2	MLPDW	MLPConv	✓	27.2M	30.66	0.0064	14.26	26.88

TABLE 5.2: Ablation study on decoder, feature skip designs on the Composition-1k test set. 'MLPDW' denotes 'MLP+DepthWise Conv'.

No.	LFA	l_{lap}	l_g	l_{gp}	#Params	SAD	Grad
-					15.8M	28.94	12.75
-		✓			15.8M	27.64	10.68
-			✓		15.8M	27.71	10.23
-				✓	15.8M	27.00	9.50
N3		✓		✓	15.8M	25.86	9.69
-	✓				16.8M	27.67	12.44
M3	✓	✓		✓	16.8M	25.70	9.50
-					26.8M	26.11	10.59
M7	✓	✓		✓	27.9M	25.00	9.02

TABLE 5.3: Ablation study on the LFA module and loss functions on the Composition-1k test set. The upper part(containing **N3/M3**) and the lower part(containing **M7**) are based on **B3** and **B7**, respectively.

image is generated on-the-fly through the following basic augmentation: foreground random affine, foreground random combination, random resize, random crop, foreground random jitter, and composition. More details are explained in [60]. 512 × 512 patches are finally generated for training. We initialize the weights of the mit backbones using the pretrained weights on ImageNet-1K[26] from [112] for the transformer branch. Other parameters are initialized with Xavier. The training stage is optimized by AdamW [70] optimizer using initial learning rate 6×10^{-4} with cosine decay. The warm up stage takes 1000 iterations. Without specially clarifying, we update parameters for $90k$ iterations with a batch size of 32. Batch size 64 and $120k$ iterations are used for final benchmark results, as detailed in Table 5.4 and 5.5.

Loss Functions. Our baseline models only use L1 alpha prediction loss and composition loss as [72]. Since other loss functions, such as laplacian loss (l_{lap}) and gradient loss (l_g), are applied in previous pure convolution-based methods [32, 43], here we validate their effects in our framework. Besides using the usual l_{lap} and l_g, we define a new gradient loss with gradient penalty(l_{gp}) for local smoothness:

$$\begin{aligned} l_{gp} = &\|\nabla\alpha_x - \nabla\hat{\alpha}_x\|_1 + \|\nabla\alpha_y - \nabla\hat{\alpha}_y\|_1 \\ &+ \lambda\left(\|\nabla\alpha_x\|_1 + \|\nabla\alpha_y\|_1\right) \end{aligned}, \tag{5.4}$$

where λ is set as 0.01 in our experiments.

Method	SAD	MSE	Grad	Conn	# Params
CF [57]	168.1	0.091	126.9	167.9	-
KNN [11]	175.4	0.103	124.1	176.4	-
DIM [113]	50.4	0.014	31.0	50.8	$>$ 130.55M
IndexNet [72]	45.8	0.013	25.9	43.7	8.15M
CA [43]	35.8	0.0082	17.3	33.2	107.5M
CA+$\mathcal{DA}$ [43]	71.3	0.0236	38.8	72.0	107.5M
GCA [60]	35.28	0.0091	16.9	32.5	25.27M
A^{2}U [22]	32.15	0.0082	16.39	29.25	8.09M
SIM [97]	28.0	0.0058	10.8	24.8	70.16M
FBA [32]	26.4	0.0054	10.6	21.5	34.69M
FBA+TTA [32]	25.8	0.0052	10.6	20.8	34.69M
M3‡	23.98	0.0042	8.54	18.88	16.8M
M7‡	**22.87**	**0.0039**	**7.74**	**17.84**	27.9M

TABLE 5.4: Benchmark results on the Composition-1k test set. The best performance is in boldface. ‡ denotes training with a batch size of 64, 120k iterations using our $\mathcal{SA}$.

5.5.2 Results on the Adobe Image Matting Dataset

Ablation Study on Model Architecture. Based on the two-branch encoder, here we investigate designs of the decoder, the skip layer and the additional attention module.

According to the results in Table 5.2 and Table 5.3, we draw the following observations: **1)** Compared with the MLP layer and the MLPDW layer, the Conv layer suits the decoder of matting better (B1 vs. B2, B5 vs.B3, B6 vs. B4, B9 vs.B7, B10 vs. B8); **2)** Skipped information from the transformer branch to the decoder can be efficiently achieved by a simple MLP layer (B3 vs. B4, B5 vs. B6, B9 vs. B10); **3)** Low-level skip fusion is important for recovering details (B2 vs. B3, also see Fig. 5.4b); **4)** Additional low-level feature assembling attention module further improves the results (Table 5.3); **5)** The advantage of larger backbone is gradually weakened with improvement on the architecture and loss functions (B3→M3 vs. B7→M7 in Table 5.3).

Ablation Study on Loss Functions. Here we justify effectiveness of laplacian loss (l_{lap}), gradient loss (l_g) and the proposed gradient loss with gradient penalty (l_{gp}) in our framework. Results are reported in Table 5.3. Compared with using only basic loss functions, l_{lap}, l_g, and l_{gp} all reduce the errors, and our l_{gp} works better than normal l_g. Combining l_{lap} and l_{gp} together builds the best prediction. We use these two losses in the following experiments.

Comparison with State of the Art. Benchmark results on the Composition-1k are list in Table 5.4. Our models achieve significantly better results on all the metrics. Compared with a currently top-performing FBA+TTA [32] model, our method (M7‡) gains 11% improvement on SAD and 27% improvement on Grad without any augmentation on the test images. Moreover, our method is more robust to trimap precision, as shown in Fig. 5.2 and 5.5, and the detailed evaluations in the appendix.

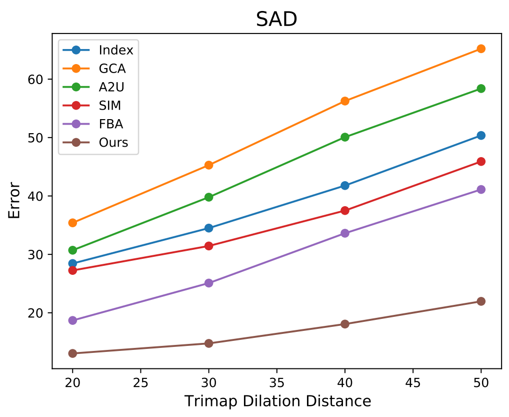


FIGURE 5.5: Robustness to trimap precision on AIM-500

5.5.3 Generalization on Various Benchmarks

To verify the generalization ability of matting methods to unseen domains, comparison experiments are carried out in Table 5.5. Specifically, we test models trained merely with the DIM dataset [113] on several different benchmarks without fitting on their training set (except that SIM [97] is trained on SIMD [97], which has 763 foregrounds, 332 more foregrounds than DIM). The test benchmarks include Distinction-646 [80], SIMD [97], and AIM-500 [59]. Distinction-646 and SIMD are synthetic benchmarks, and AIM-500 is a real-world one but with simple scenes, so none of them alone is perfect for measuring generalization ability of a model. However, since they contain images from different sources and various domains, it is at least reasonable to combine them together to see how algorithms perform quantitatively on all of them, and the overall results should reflect how well a model can adapt to diverse images to some extent. Note that, since SIMD has only provided alpha ground truths and foregrounds until the submission, we generate the test set following the rule of [113] and name it as SIMD$_{our}$. To ensure all the methods can be test on a normal modern graphic card, we restrict the maximum length of the test images in SIMD$_{our}$ by 2000.

Ablation Study on Strong Data Augmentation. Here we investigate the $\mathcal{SA}$ strategies. We either use AF, AFB alone, or combine them with AC. Specifically, the linear pixel-wise augmentations include: *linear contrast, brightness adjustment, channel inversion/shuffling, gaussian/poisson noise, random dropout, cloud, snow, multiply, salt and pepper*; the nonlinear pixel-wise augmentations include: *gamma contrast, hue and saturation add on, histogram equalization*; and region-wise augmentations consist of *gaussian blur and jpeg compression.* If AF or AFB is applied alone, we set the possibility as 0.5 and keep the ground truths unmodified; if they are combined, possibility of each is changed to 0.25; further, if AC is added on, we set its possibility as 0.1 when AF and AFB do not happen, and generate pseudo labels for the augmented samples as explained in Section 5.4 when needed. More details are in appendix. As shown in Table 5.5, both

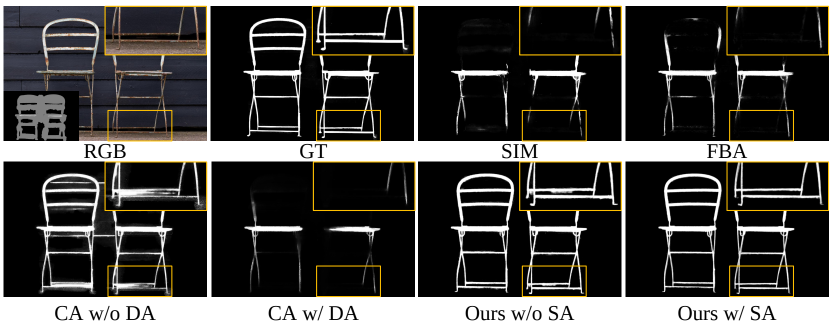


FIGURE 5.6: Visual results on the AIM-500 benchmark. The methods in comparison are SIM [97], FBA [32], CA w/o $\mathcal{DA}$ [43], CA w/ $\mathcal{DA}$ [43], Ours w/o $\mathcal{SA}$, Ours w/ $\mathcal{SA}$.

AF or AFB improve the AIM-500 results, especially AFB, but they also make errors on Distinction-646, SIMD$_{our}$, and Composition-1k (appendix) slightly higher. AF is more stable on synthetic benchmarks compared with AFB. Hence, we carry out 'AF+AFB'. It averages the effects of AF and AFB. Based on 'AF+AFB', AC further improves results on AIM-500 and keeps results on other synthetic benchmarks comparable, so we use 'AF+AFB+AC' as the final $\mathcal{SA}$.

Comparison with State of the Art. Compared with other methods, our M3‡ ans M7‡ models achieve best performance on all three benchmarks in Table 5.5, especially with MSE and Grad metrics. Note that, $\mathcal{DA}$ in [43] degrades its performance on Composition-1k (Table 5.4), SIMD$_{our}$ and AIM-500 significantly, even though AIM-500 is a real-world benchmark. Our $\mathcal{SA}$ instead promises comparable results on the synthetic benchmarks and much better results on the real-world benchmark. The advantages of $\mathcal{SA}$ against $\mathcal{DA}$ can also be noticed from visual examples in Fig. 5.1 and 5.6. Moreover, when longer training time is taken, stable improvements are observed. The examples in Fig. 5.2 and 5.6 further verify the effectiveness of our model and $\mathcal{SA}$. More visual results on real-world images and benchmarks are shown in the appendix.

5.5.4 Results on the `alphamatting.com`

We show the results of M7‡ on the `alphamatting.com` [81] online benchmark in Table. 5.6. Note that, SIM is trained with the SIMD training set, which has 736 foregrounds in the training set, while DIM only has 431 foregrounds in the training set; GCA and A^{2}U retrain their models with the whole DIM dataset (including both training set and test set). Our result is directly reported from M7‡ without using extra data or fine-tuning the model, but it still achieves top-performing ranks, especially on MSE and Grad. See the full table in the appendix.

Method				Distinction-646 [80]				SIMD$_{our}$ [97]				AIM-500 [59]			
				SAD	MSE	Grad	Conn	SAD	MSE	Grad	Conn	SAD	MSE	Grad	Conn
IndexNet* [72]				42.64	0.0256	40.17	42.76	92.45	0.0388	45.85	93.14	28.49	0.0288	18.15	27.95
CA* [43]				49.07	0.0557	114.77	48.27	79.46	0.0291	51.03	77.88	26.33	0.0266	18.89	25.05
CA+$\mathcal{DA}$* [43]				46.03	0.0356	55.45	46.18	102.97	0.0469	74.39	103.52	32.15	0.0388	30.25	31.00
GCA* [60]				31.00	0.0171	21.19	29.62	75.81	0.0271	40.57	74.45	35.10	0.0389	25.67	35.48
A^{2}U* [22]				28.74	0.0143	17.42	27.62	68.70	0.0268	39.00	66.76	30.38	0.0307	22.60	30.69
SIM* [97]				**22.68**	0.0137	20.11	21.03	**37.07**	0.0099	22.29	33.30	27.05	0.0311	23.68	27.08
FBA* [32]				30.70	0.0150	18.89	29.65	41.55	0.0109	23.21	35.07	19.05	0.0162	11.42	18.30
	AF	AFB	AC												
N3				25.86	0.0105	13.25	24.11	41.83	0.0099	22.19	34.98	16.08	0.0122	11.69	15.55
N3	✓			25.18	0.0108	13.45	23.35	44.14	0.0122	23.46	37.96	16.38	0.0112	10.75	16.00
N3		✓		27.02	0.0123	14.44	25.34	44.46	0.0113	23.93	38.30	**13.46**	0.0097	9.98	**12.47**
N3	✓	✓		26.46	0.0117	14.18	24.87	42.89	0.0110	23.37	36.05	14.63	0.0108	10.46	13.75
N3	✓	✓	✓	26.32	0.0119	14.40	24.56	43.49	0.0109	23.96	36.84	14.18	0.0093	9.36	13.61
M3‡	✓	✓	✓	23.67	0.0100	11.30	21.37	42.68	0.0121	21.20	35.84	13.68	**0.0091**	9.36	13.06
M7‡	✓	✓	✓	23.25	**0.0097**	**11.09**	**21.00**	37.31	**0.0090**	**20.00**	**30.10**	13.97	0.0094	**8.89**	13.21

TABLE 5.5: Generalization results on the Distinction-646, SIMD$_{our}$ and AIM benchmarks. The best performance is in boldface. The second is underlined. **All the models are merely trained with the DIM training set (431 foregrounds), except that SIM is trained with the SIMD training set (736 foregrounds), so we set SIM's results on SIMD as blue color to represent SIM is trained with this dataset.** * means using the officially provided model. ‡ denotes training with a batch size of 64, 120k iterations using our $\mathcal{SA}$ (AF+AFB+AC).

Method	MSE				Grad			
	overall	S	L	U	overall	S	L	U
Ours-M7‡	**6.8**	**5.9**	**5.5**	9.1	**4.7**	**4.8**	**3.8**	**5.5**
SIM [97]	7	8.1	**5.5**	**7.4**	6.9	8.5	5.9	6.5
A^{2}U [22]	15.5	13	12.6	20.8	12.3	11.3	9.4	16.1
GCA [60]	15.3	15.1	14.5	16.4	13.7	13.6	12.5	15
CA [43]	17.6	20.9	18.6	13.3	14.6	15.8	15.5	12.6
IndexNet [72]	22.9	25.3	21.5	22	18.6	17.3	17.3	21.4

TABLE 5.6: Results on the `alphamatting.com` online benchmark.

5.6 Conclusion

We propose RMat, a matting method showing higher robustness to various trimap precision and images from different domains. The efforts behind this include a new matting framework and strong augmentation strategies specifically designed for matting. We first build the strong baseline by assembling multilevel context information, then analyse the problems behind current data augmentation and design strong augmentation strategies. To verify generalization capability of the model, we not only show visual results on real-world images, but also design a series of evaluation experiments on several benchmarks without fitting their training sets. Our method achieves state-of-the-art results on all the benchmarks. We hope our work opens up more possibilities for future works on deep matting.

Limitations There are still many hard cases, such as strong light in the background, cannot be handled by our method. We show failure cases in the appendix. To handle those cases, we may need to better learn the structure of the foreground objects. We leave it as a future work.

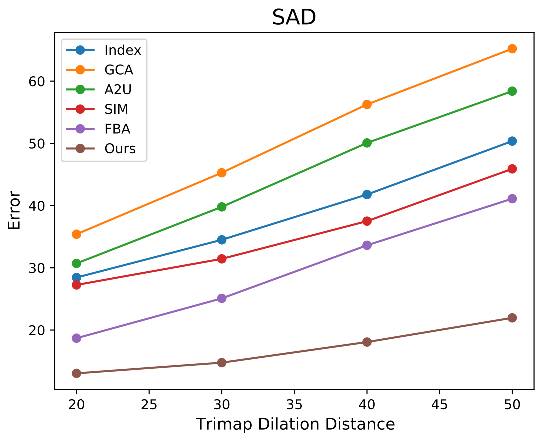
5.7 Appendix

5.7.1 Measure of GFLOPs

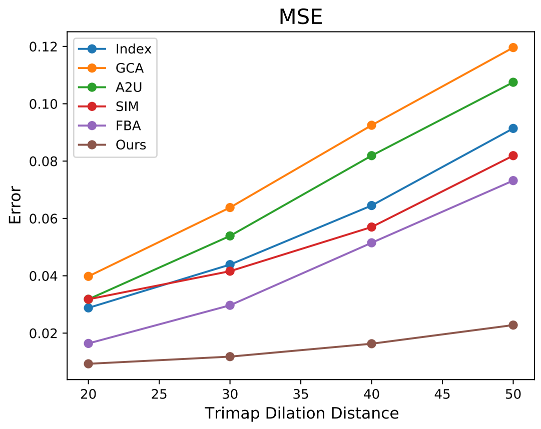
We measure GFLOPs of SIM [97], FBA [32] and our method. Results are reported in Table.5.7.

Method	GFLOPs
SIM [97]	48.30
FBA [32]	30.47
M3‡	16.62
M7‡	22.22

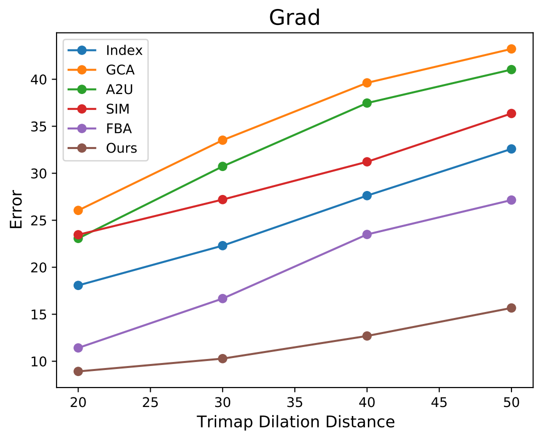
TABLE 5.7: GFLOPs measured on a 224×224 input.



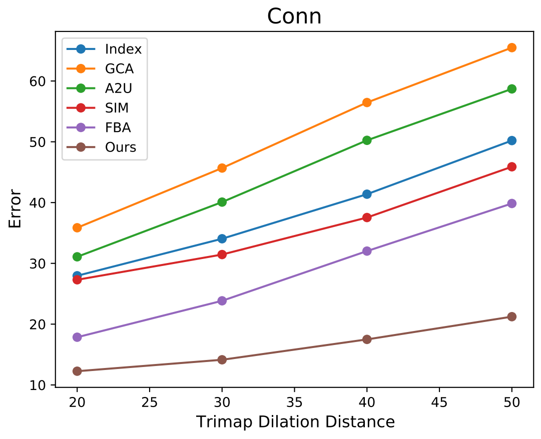
(a)



(b)



(c)



(d)

FIGURE 5.7: Robustness to trimap precision on the AIM-500.

5.7.2 Robustness to Trimap Precision

We conduct evaluations on the AIM-500 with different trimap dilation distances. Methods in comparison are IndexNet [72], GCA [60], A^{2}U [22], SIM [97], FBA [32] and our M7‡. In detail, we generate 4 sets of trimaps using random dilation distances within $[11, 20]$, $[21, 30]$, $[31, 40]$, $[41, 50]$, respectively. We denote them as 20, 30, 40, 50 accordingly in Fig. 5.7. As shown in Fig. 5.7, our method is obviously more robust to varying trimap precision on all the metrics.

5.7.3 Details on the Strong Data Augmentation Strategies

To supplement the content in the main text, we further detail the $\mathcal{SA}$ strategies in our experiments here.

Method	SAD	MSE	Grad	Conn
N3+AF	25.86	0.0045	9.82	21.27
N3+AFB	26.21	0.0048	9.91	21.43
N3+AF+AF	26.55	0.0050	10.45	21.93
N3+AF+AFB+AC	26.46	0.0049	9.98	21.72

TABLE 5.8: Generalization results on the Composition-1k.

If AF or AFB is applied alone, we set the possibility as 0.5 and keep the ground truths unmodified; if they are combined, possibility of each is changed to 0.25; further, if AC is added on, we set its possibility as 0.1 when AF and AFB do not happen.

Specifically, in AF and AFB, linear pixel-wise augmentation, nonlinear pixel-wise augmentation and region-wise augmentation happen with a probability of 0.8, 0.1 and 0.1, respectively. In AC, linear pixel-wise augmentation, nonlinear pixel-wise augmentation and region-wise augmentation happen with a probability of 0.2, 0.4 and 0.4, respectively. All the augmentations are randomly selected from the options list in the main text during each operation.

5.7.4 Ablation study and extensive verification on Strong Data Augmentation

We report ablation study results on $\mathcal{SA}$ on the Composition-1k in Table. 5.8. In consistent with results in the main text, our $\mathcal{SA}$ produces comparable results on the synthetic benchmarks.

We also show additional verification of $\mathcal{SA}$ on the A^2U [22] on AIM-500 in Table. 5.9.

Method	SAD	MSE	Grad	Conn
A^2U [22]	30.38	0.0307	22.60	30.69
A^2U+AF+AFB+AC	19.55	0.0165	15.02	19.10

TABLE 5.9: Results of $\mathcal{SA}$ on the A^2U [22] on AIM-500.

5.7.5 More Visual Results

Here we show more visual results. The visualized methods include IndexNet [72], CA [43], GCA [60], A^2U [22], SIM [97], FBA [32] and our M7‡.

Visual results on real-world images are present in Fig. 5.8 and 5.9, where Fig. 5.9 further shows results on coarse trimaps. Our method is more robust in these real-world test cases with coarse-to-fine trimaps.

Visual results on the AIM-500 [59] are exhibited in Fig. 5.10. Our method achieves better results on structures such as leaves and net.

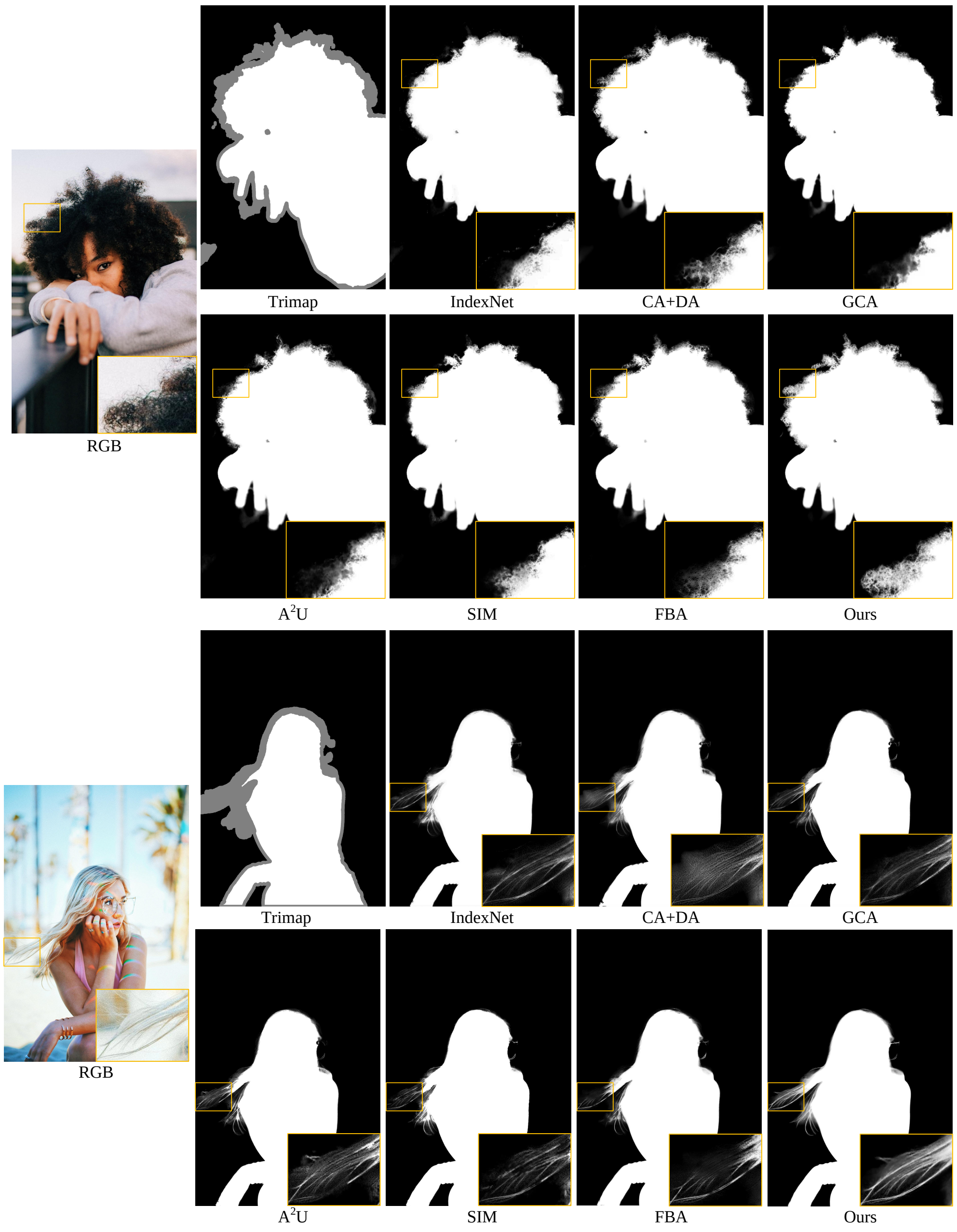


FIGURE 5.8: Visual results on real-world images. Best viewed by zooming in.

5.7.6 Results on the `alphamatting.com`

We report results of M7‡ on the `alphamatting.com` online benchmark in Table. 5.10. The methods in comparison are SIM [97], A^{2}U [22], GCA [60], CA [43], IndexNet [72]. There are only 8 test images in this online benchmark. It worth noting that, SIM is trained with the SIMD training set, which has 736 foregrounds (including 360 foregrounds from DIM) in the training set, while DIM only has 431 foregrounds in the training set; GCA and A^{2}U retrain their models with the whole DIM dataset (including both training set and test set, 481 foregrounds in total) for this benchmark. Our result is directly reported

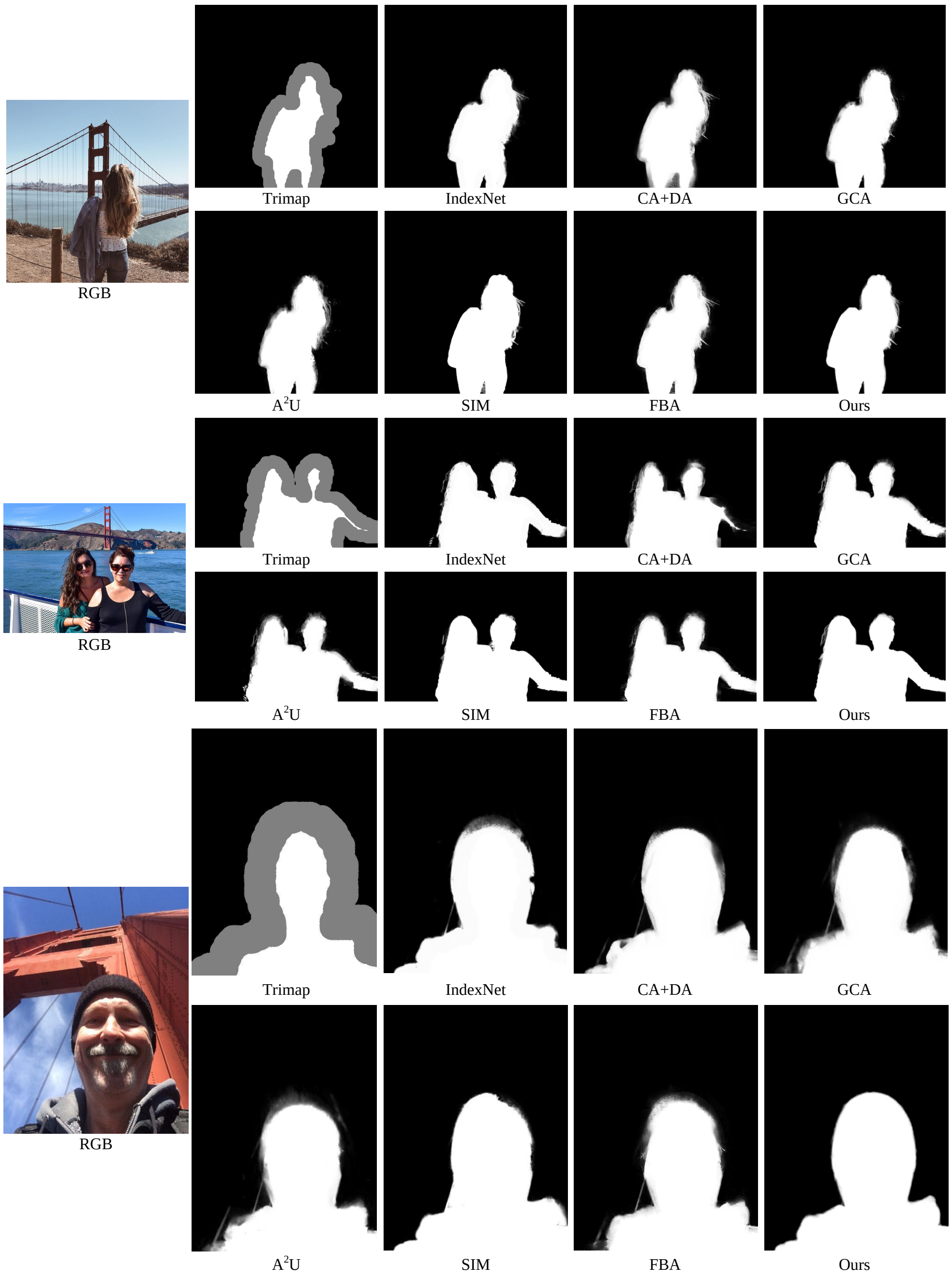


FIGURE 5.9: Visual results on real-world images with coarse trimaps.

from M7‡ trained with the DIM training set without using extra data or fine-tuning the model, but it still achieves top-performing ranks.

Method	SAD			MSE			Grad			Conn		
	overall	L	U	overall	L	U	overall	L	U	overall	L	U
Ours-M7‡	6.7	**5.8**	8.5	**6.8**	**5.5**	9.1	**4.7**	**3.8**	**5.5**	12.4	13.7	**7.6**
SIM [97]	**6.5**	**5.8**	**6.6**	7	**5.5**	**7.4**	6.9	5.9	6.5	**10**	**8.9**	11.3
A^2U [22]	13.3	10.6	17	15.5	12.6	20.8	12.3	9.4	16.1	27.3	28	24.3
GCA [60]	14.5	12.4	16	15.3	14.5	16.4	13.7	12.5	15	22.5	20.1	21.4
CA [43]	22.9	20.9	21	17.6	18.6	13.3	14.6	15.5	12.6	25.9	24.6	25
IndexNet [72]	19.4	18.1	18.6	22.9	21.5	22	18.6	17.3	21.4	25.5	26.4	26

TABLE 5.10: Results on the `alphamatting.com` online benchmark.

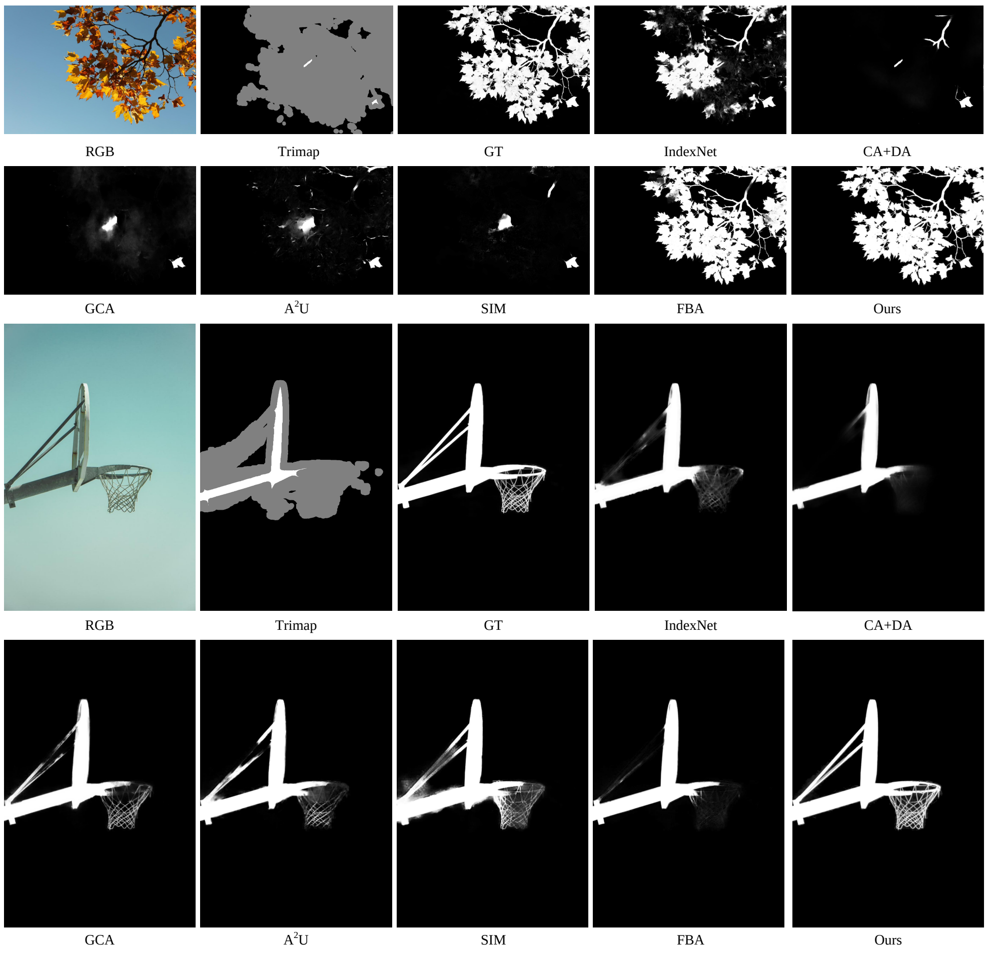


FIGURE 5.10: Visual results on the AIM-500.

5.7.7 Failure Cases

Failure examples are visualized in Fig. 5.11. Our method may fail if there is strong light in the background or there are tiny objects overlapping with the foreground object. A possible solution is to learn the structure of the foreground objects. We leave it as a future work.

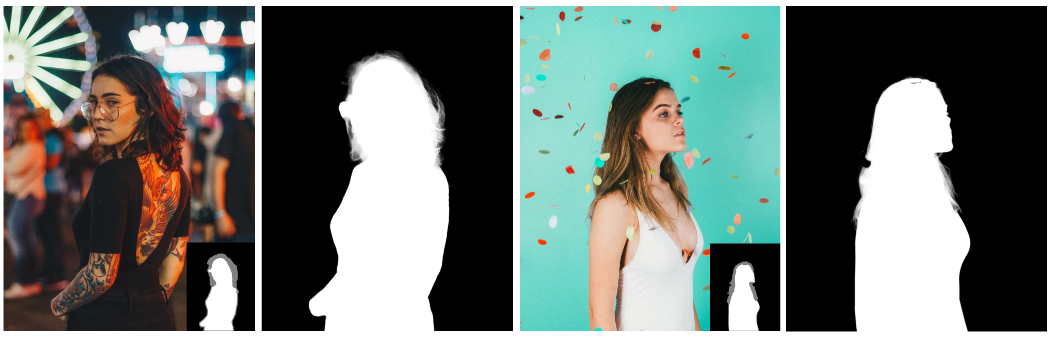


FIGURE 5.11: Failure cases.

Statement of Authorship

Title of Paper	Towards Light-Weight Deep Portrait Matting via Parameter Sharing
Publication Status	[✓ Published] ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	Computer Graphic Forum / Volume 40, Issue 1 / p. 151 – 164

Principal Author

Name of Principal Author (Candidate)	Yutong Dai		
Contribution to the Paper	Experiment design and implementation, experiment conduction, writing draft of the paper		
Overall percentage (%)	70%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	09/12/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Hao Lu		
Contribution to the Paper	Discussion and writing revision		
Signature		Date	09/12/2021

Name of Co-Author	Chunhua Shen		
Contribution to the Paper	Discussion and writing revision		
Signature			09/12/2021

Please cut and paste additional co-author panels here as required.

Chapter 6

Towards Light-Weight Portrait Matting via Parameter Sharing

Afore three chapters are about natural image matting. In this chapter, we focus on a more specific target – portrait image matting. Traditional portrait matting methods typically consist of a trimap estimation network and a matting network. Here, we propose a new light-weight portrait matting approach, termed parameter-sharing portrait matting (PSPM). Different from conventional portrait matting models where the encoder and decoder networks in two tasks are often separately designed, here a single encoder is employed for the two tasks in PSPM, while each task still has its task-specific decoder. Thus, the role of the encoder is to extract semantic features and two decoders function as a bridge between low-resolution feature maps generated by the encoder and high-resolution feature maps for pixel-wise classification/regression. In particular, three variants capable of implementing the parameter-sharing portrait matting network are proposed and investigated, respectively. As demonstrated in our experiments, model capacity and computation costs can be reduced significantly, by up to 57.8% and 40.5% respectively with PSPM, whereas the matting accuracy only slightly deteriorates. In addition, qualitative and quantitative evaluations show that sharing the encoder is an effective way to achieve portrait matting with limited computational budgets, indicating a promising direction for applications of real-time portrait matting on mobile devices.

6.1 Introduction

Selfie is taking place anywhere with the prevalence of mobile devices supporting image acquisition and post-processing software benefiting image manipulation. Therein, image matting plays an important role in the pre-processing of image manipulation, such as

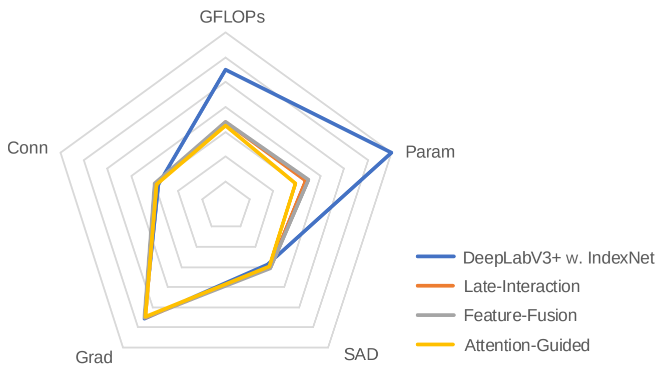


FIGURE 6.1: The model capacity (Param.), computational complexity (GFLOPs), Sum of Absolute Differences (SAD), Gradient (Grad) and Connectivity (Conn) errors of different models on the Portrait-2k test set. DeepLabV3+ [10] and IndexNet [72] are currently state-of-the-art segmentation and matting networks, respectively. 'DeepLabV3+ w. IndexNet' is the cascaded structure implementing portrait matting without trimap input (prior-free). 'Late-Interaction', 'Feature-Fusion' and 'Attention-Guided' are our proposed light-weight prior-free portrait matting networks. Compared with the cascaded structure, our propositions reduce model sizes and computational complexity by large margins with almost identical matting performance.

stylization and background editing. To improve the practicability of portrait matting, a primary obstacle is how to eliminate costly human input while still retaining high-quality matting at the same time. Recent work tackles this problem by introducing a trimap estimation network specializing in prior generation and proposing new network modules for better detail delineation of foreground. Since a high-capacity network is usually unavoidable if directly generating prior information using a network, several methods investigate end-to-end training or design light-weight models [45, 87, 128]. On the other hand, to preserve details of foreground, some methods incorporate off-the-shelf matting algorithms such as closed-form matting [57], and deep image matting [113] into their architectures or specifically collect large-scale human matting datasets [89, 12]. Despite many solutions exist, it remains a challenging topic about how to achieve light-weight and high-quality deep portrait matting in a unified framework.

Our work attempts to alleviate the above dilemma. A light-weight but high-quality portrait matting method is in demand. Since almost all existing methods generate prior information and alpha prediction in a cascaded manner, we observe that, the cascaded architecture (a trimap estimation network followed by a matting network) is effective but not efficient.We hence propose to establish a new prior-free paradigm—parameter-sharing portrait matting (PSPM), as shown in Figure 6.2. Our explorations suggest that explicitly applying semantic segmentation to generate prior input is only one alternative. Despite matting and segmentation networks have different outputs, they share similar high-level features from the encoder network. This observation inspires us to explore architectures that use the idea of parameter sharing. It is also the foundation of promoting the end-to-end network to be more efficient and light-weight. In particular, built upon a recent state-of-the-art light-weight matting network [71, 72], we investigate three alternative architectures to generate prior information from a segmentation decoder following the shared encoder.

According to the extent of sharing parameters, three architectures are respectively named as i) *late-interaction* architecture, where two decoders only communicate when calculating the loss, ii) *feature-fusion* architecture, where the output of segmentation decoder functions as the input of matting decoder and iii) *attention-guided* architecture, where features of different layers are combined more compactly before passing to the matting decoder. Since these architectures share parts of parameters of two tasks, their complexity and parameters are reduced significantly, but with almost the same matting performance (Figure 6.1). Overall, our main contributions are:

- We introduce the idea of parameter-sharing multi-task learning to deep portrait matting, significantly reducing the number of parameters and computational complexity;
- We investigate three alternative architectures for end-to-end prior-free portrait matting, which achieves parameter reduction and prior information integration.

6.2 Background

Deep Portrait Matting Deep portrait matting primarily concentrates on the prior-free paradigm and light-weight models. Shen *et al.* [89] propose a cascaded architecture, consisting of a Fully Convolutional Network (FCN) [69] for generating trimaps and a closed-form [57] matting module. Chen *et al.* [12] combine the models in [89] and [113], using high-capacity PSPNet-50 [124] and DIM-VGG16 [113, 92] as the segmentation network and the matting network, respectively. With a high-quality dataset used for training, the proposed network can delineate great details on edges such as hair. These methods divide portrait matting into segmentation and matting tasks.

Apart from this idea, some solutions are proposed to eliminate the necessity of prior input in image matting. Li *et al.* [61] generate weakly annotated masks as the guidance of deep image matting. Chen *et al.* [13] employ a boundary-aware module to refine the output of a FCN structure for high quality portrait segmentation. Prior-free architectures in these works suggest the possibility of expanding solutions for portrait matting. Generally speaking, the role of prior input is to provide some annotations to ease the task.

Parameter-Sharing Multi-Task Learning Multi-task learning is a mechanism to share parameters in a network where multiple tasks are undertaken. It has been successfully applied to various fields, from natural language processing [20, 27] to computer vision [36, 37, 123]. This mechanism is expected to enhance the universality of the network as more diverse training data may be used during training and multiple tasks can help each other. Since each task has a certain focus, the mechanism also integrates information from multiple views and then better guides each task-specific branch to extract on task-specific features. For example, if one task fails to learn some features appropriately, its counterparts are anticipated to fill the vacancy. If properly structured, multi-task learning not only shares parameters but also benefits mutually among tasks.

The mode of parameter sharing in multi-task learning can be classified into two categories [85]: *hard parameter sharing* and *soft parameter sharing*. To be specific, hard parameter sharing means sharing some hidden layers among tasks and keeping task-specific output layers, while soft parameter sharing refers to the idea that each task has its own parameters but the distances among parameters of different tasks are regularized. In this work, hard parameter sharing is used to reduce redundant parameters. We treat portrait matting as a hybrid task of segmentation and image matting for its intuitive architecture and clear task settings, then design parameter sharing and feature connection between two tasks.

6.3 Architecture Design

6.3.1 An End-to-End Matting Network Baseline

Following [89, 12], we build a baseline architecture that sequentially connects a segmentation network and a matting network. This is an intuitive combination of two tasks. Considering that, neither implementations of [89, 12] nor standard benchmark results of portrait matting have been released, and this work focuses on investigating parameter sharing mechanisms for the cascaded structure, we study different degrees of parameter-sharing manners to reduce parameters and computation complexity based on the baseline.Thus it is our focus to show the relative improvement against the baseline.

The segmentation network incorporated into our baseline can be chosen from a wide variety of architectures. Here we choose DeepLabV3+ [10] with MobileNetv2 [86] as the backbone. The matting network adopts the recently proposed IndexNet matting [72] (IndexNet is used for short in the following sections) in view of its strong ability in preserving edges and details. These two networks share the same light-weight backbone. The overall framework of our baseline is shown in Figure 6.2.

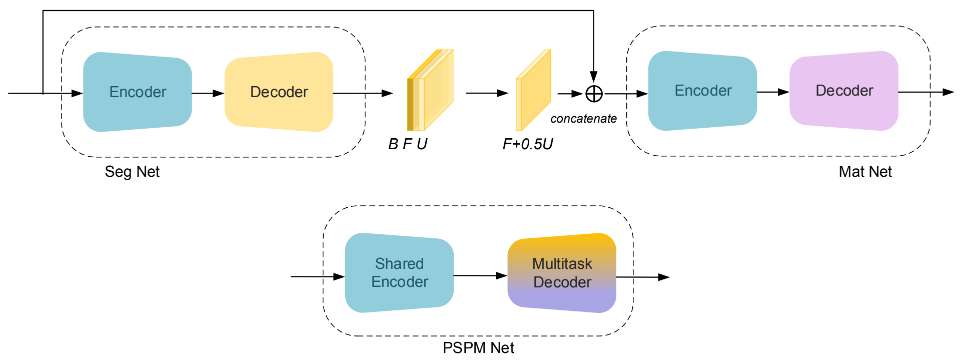


FIGURE 6.2: Upper: Our baseline framework for prior-free portrait matting. The segmentation network is DeepLabV3+ [10], and the matting network is IndexNet [72]. Bottom: the framework of our PSPM.

Since DeepLabV3+ and IndexNet are two independent pre-trained models, the cascaded network can either be directly used without updating weights of the model or be end-to-end fine-tuned. Here we initially use fixed weights to verify the effectiveness of the cascaded design. This architecture has been demonstrated in previous works [89, 12].

Based on the weight-fixed architecture, we investigate end-to-end training of this network. We make two observations. First, since 4 channels, RGB channels plus a trimap channel, are set as standard inputs of IndexNet, the 3-channel outputs of DeepLabV3+ are summed to 1-channel trimap to keep the structure of IndexNet:

$$M = F_p + \frac{U_p}{2}\,, \tag{6.1}$$

where F_p and U_p are the probabilities of the foreground and unknown categories after the softmax layer, respectively. F_p takes the form:

$$F_p = \frac{e^F}{e^F + e^B + e^U}\,, \tag{6.2}$$

and B_p and U_p can be derived similarly. Equation (6.1) follows the process how a trimap is generated in natual image matting [113]. Considering that, $F_p + B_p + U_p = 1$, Equation (6.1) contains all the information of segmentation results without modifying the structure of IndexNet. Second, we use the alpha prediction loss L_a following [113], which is calculated in the whole image rather than only in unknown regions to optimize both the segmentation network and the matting network. We also apply a Cross-Entropy loss L_c to optimize the segmentation network directly. The final loss is:

$$loss = L_a + \lambda L_c\,, \tag{6.3}$$

where λ is set to 0.1 in our experiments.

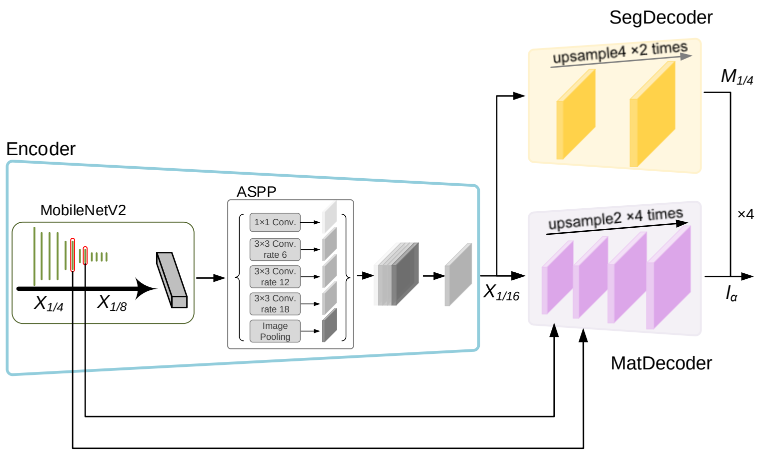


FIGURE 6.3: Late-interaction parameter-sharing network.

6.3.2 Parameter-Sharing Matting Network

Both segmentation and matting networks obtain high-level semantic information in the encoding stage. It is natural to pose the question: *Can we share one encoder between two tasks?* The encoders of two tasks are of similar architectures and perform similar duties, but the decoders focus on different tasks. We therefore investigate sharing one encoder by keeping task-specific decoder branches. Specifically, through increasing feature connections between decoder branches, three architectures are explored.

Late-Interaction Parameter Sharing To validate the above motivation, as shown in Figure 6.3, a straightforward solution is to share the encoder and parallelize the task-specific decoders, among which the loss calculation of the matting decoder is assisted by the output of the segmentation decoder. Since both encoders here use MobileNetv2 as the backbone, they can be shared with minimal modifications. The main differences between encoders of DeepLabV3+ and IndexNet are: 1) IndexNet generates the output with a downsampling factor of 32, while the downsampling factor in DeepLabV3+ is 16, and 2) DeepLabV3+ implements downsampling with stride-2 convolution, while downsampling in IndexNet is operated by max-pooling or indices-guided pooling. To unify these two encoders, 1) we set the output downsampling factor of the encoder to be 16 considering it is relatively simple to segment a portrait image into a three-category trimap, and 2) we replace stride-2 convolution with stride-1 convolutions followed by max-pooling to preserve details for matting. In terms of max-pooling, SegNet [3] uses it in the segmentation task, and deep image matting [113] applies it to the matting task. Unifying the encoder with max-pooling hence is appropriate for both tasks. The decoder of segmentation, termed SegDecoder, is identical to the one used in DeepLabV3+. The decoder of matting, termed MatDecoder, follows IndexNet with the use of max-unpooling. Learning indices can often improve the performance on the matting task.

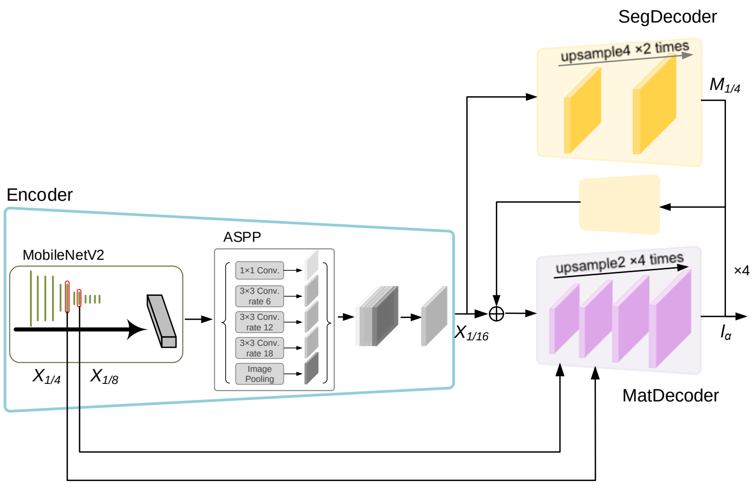


FIGURE 6.4: Feature-fusion parameter-sharing network.

However, using simple max-pooling and max-unpooling also allows us to explore whether the architecture works. Since we apply max-pooling to the encoder, 5×5 convolution is kept in MatDecoder because single-layer 3×3 convolution cannot cover all the pixels in a 2×2 region of feature maps before unpooling.

Feature-Fusion Parameter Sharing Trimaps are usually used to be part of inputs and guide loss calculations [113, 89, 74]. Two use cases can be categorized as prior information and posterior information, respectively. The late-interaction network uses posterior information only. To further introduce prior information to MatDecoder, we investigate feature connections between two decoders. Specifically, as shown in Figure 6.4, we send the output of SegDecoder to MatDecoder as the prior information following many matting methods in the literature.

In the late-interaction architecture, the output of SegDecoder is of the same resolution as input. To reduce downsampling here, the feature map before the last upsampling in SegDecoder is chosen to produce prior information for the MatDecoder. We achieve this using Equation (6.1) and name the feature map as $M_{\frac{1}{4}}$ according to its resolution. $M_{\frac{1}{4}}$ is first passed through several convolutional layers and downsampled to $\frac{1}{16}$ resolution, then is concatenated with the output of the encoder ($X_{\frac{1}{16}}$) to form the input of MatDecoder. The output of SegDecoder also assists computing the alpha prediction loss. This architecture may be viewed as mimicking common image matting solutions by sending information of both input and generated prior knowledge to MatDecoder. Since both tasks affect the alpha prediction loss, they are expected to benefit each other.

Attention-Guided Parameter Sharing The feature-fusion network uses the output of SegDecoder as prior knowledge of the matting task. However, this architecture has

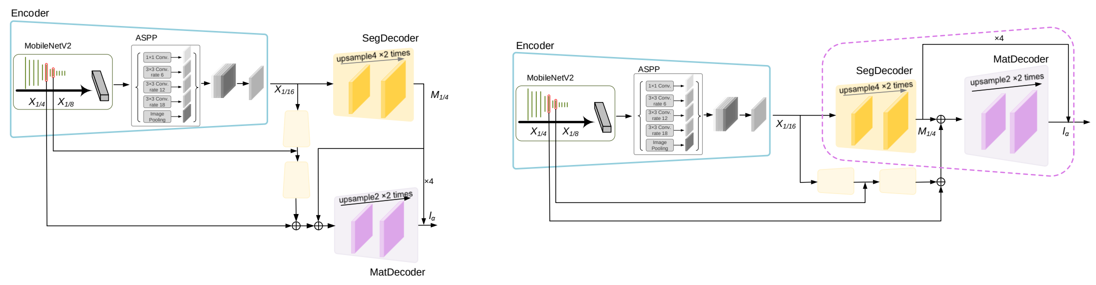


Figure 6.5: Attention-guided parameter-sharing network. The left one and the right one are the same network shown in different manners.

	SAD	MSE	Grad	Conn
GT Trimap + IndexNet(A1)	4.62	0.0278	10.97	4.49
No Trimap + IndexNet(A2)	22.21	0.2147	21.96	22.72
A2 + posterior(A2+)	6.55	0.0590	12.03	6.42

Table 6.1: Results on the Portrait-2k dataset, to verify the effectiveness of Prior.

more parameters than the late-interaction design. It motivates us to investigate a light-weight design. As shown in Figure 6.5, this can be achieved by incorporating prior information generated by SegDecoder to the matting branch more efficiently.

The use of prior knowledge in deep matting methods is closely related to the attention mechanism—pixels that belong to the same category should be treated similarly. It may introduce differences to foreground, background and unknown categories in convolutional layers. To verify this mechanism, we remove trimap from the input of the IndexNet baseline and then train the network with only trimap-guided loss (A2+). Two counterparts of A2+ are implemented. They are weight-fixed IndexNet with/without the trimap input (A1/A2), respectively. A comprehensive comparison is shown in Table 6.1. According to Table 6.1, the network with prior input achieves superior results over the one without prior input (A1 vs. A2). It suggests that prior input plays an important role in deep matting networks. However, after training A2 with posterior information guidance, A2+ overwhelms A2 although it is still slightly weaker than A1. We conclude that the posterior information may replace the prior input. Although prior and posterior information are both used in certain deep matting methods, parts of their functions are overlapped, at least in portrait matting. Considering an alternative perspective, *can posterior be removed?* The work of [89] points out posterior knowledge is an essential guidance for loss calculation. From this point of view, posterior is necessary.

In feature-fusion network, $M_{\frac{1}{4}}$ is downsampled to $\frac{1}{16}$ resolution before concatenating with feature maps from the encoder. This raises a possibility for simplification because features in SegDecoder experience first upsampling and then downsampling. As shown in Figure 6.5, we name outputs of three different layers in encoder as $X_{\frac{1}{4}}$, $X_{\frac{1}{8}}$ and $X_{\frac{1}{16}}$, representing feature maps of $\frac{1}{4}$, $\frac{1}{8}$ and $\frac{1}{16}$ the input resolution, respectively. $X_{\frac{1}{16}}$ is the output of the last layer of encoder, containing the most high-level and compact information, while $X_{\frac{1}{4}}$ and $X_{\frac{1}{8}}$ hold more details. These feature maps are fused layer by layer followed by convolution and upsampling. Feature maps of $\frac{1}{4}$ resolution are finally obtained and sent to MatDecoder. Two convolutional and upsampling layers in MatDecoder are removed in this design. It is equivalent to incorporating two convolutional and upsampling layers into the feature fusion module. The left architecture in Figure 6.5 exhibits the network in the same manner with Figures 6.3 and 6.4, where two decoders are shown in parallel. The right one is its variant, in which we can interpret SegDecoder and MatDecoder as an integrated union: $M_{\frac{1}{4}}$ output from SegDecoder is sent to MatDecoder directly without downsampling, and MatDecoder receives feature maps from the $\frac{1}{4}$ resolution directly rather than from $\frac{1}{16}$.

6.3.3 Affinity Module

In traditional matting algorithms [56, 57], graph Laplacian is universally adopted. What lays in the core of Laplacian matrices is the affinity A_{ij}, which stands for the similarity between pixel i and pixel j. A_{ij} is mathematically derived by:

$$A_{ij} = e^{\frac{-\left\|I_i - I_j\right\|^2}{\sigma^2}}. \tag{6.4}$$

A_{ij} values of all pixels compose an affinity matrice A. Specifically, affinity matrices of the input image and the predicted alpha matte are respectively measured, which are expected to share a similar color distribution. σ here is set to be the distance between the maximum and the minimum values of the matrice.

6.4 Posterior Constraints

As discussed in Section 6.3.2, posterior knowledge is essential in current deep image matting methods. No matter prior input exists or not, posterior-guided loss functions are necessary [113, 123]. Designs of these loss functions are derived from posterior constraints. In our work, we consider two constraints. They are the alpha prediction constraint in both unknown regions and the whole image as well as the gradient constraint.

6.4.1 Region-Fusion Alpha Prediction Constraint

In this work, we measure the alpha prediction loss not only in unknown regions but also in foreground/background regions. Normally, for precise regression, the loss computed between a predicted alpha matte and the corresponding ground truth only focuses on unknown regions based on the prior input. However, no direct prior information is provided in our work. We thus learn prior-like information in the segmentation module according to the alpha prediction constraint set in different regions.

We train our network in two separate phases. In the pre-training stage of MatDecoder, the alpha prediction loss constrained in unknown regions is employed (training details are provided in the sequel). In this phase, unknown regions are generated by the weight-fixed segmentation module. We focus the loss calculation on unknown regions to precisely predict alpha values of internal pixels.

In the end-to-end training stage, the alpha prediction loss in the whole image is used. We extend loss calculation to foreground and background regions because the shared encoder, SegDecoder and MatDecoder are optimized simultaneously in this stage, which is related to the whole image. Considering the unbalanced distribution of pixels in different regions, the alpha prediction loss in unknown regions should be given a large weight, however. In Section 6.5.2 we provide implementation details.

6.4.2 Gradient Constraint

To describe how values of pixels change in an image, gradient indicator is defined. Value of this indicator changes drastically when close to boundaries. Thus it is often used as a constraint in tasks caring about edges and details [13, 98]. In image matting, for each pixel, one expects a precise alpha value prediction in a range between 0 and 1. We hence take this constraint into consideration to penalize abnormal changes in predicted alpha values.

Since we apply this constraint to the whole image, different weights are assigned in different regions as: $g_a = g_u + 0.5 g_o$, where g_a, g_u and g_o represent the gradient constraint in the whole image, unknown regions and other regions, respectively. To emphasize unknown regions, a larger weight is given to g_u than g_o.

6.5 Results and Discussions

To demonstrate the effectiveness of the proposed structures, we present details and performance of our experiments, including the datasets, training details, and results of ablation studies. Our insights on the mechanism and potential improvement of deep portrait matting are discussed in the end.

6.5.1 Datasets

The training of IndexNet is based on the Adobe Image Matting dataset [113]. The Portrait-2k [89] is the main dataset we use to implement portrait matting.

6.5.2 Training Details

An End-to-End Matting Baseline: To pursue the parameter-sharing mechanism in portrait matting, based on an off-the-shelf state-of-the-art semantic segmentation model and a light-weight matting network, we build an initial end-to-end matting network as our baseline. Specifically, using MobileNetv2 [86] pre-trained on ImageNet dataset [29] as the backbone, we train the DeepLabV3+ [10] on the Portrait-2k dataset [89]. The ground truths are generated from alpha mattes through assigning pixels with alpha values between 0 and 1 as unknown regions and then manipulating them by dilation operations. We randomly crop 512×512 sub-images as input and use conventional data argumentation, including random scaling, flipping and rotation. We use SGD optimizer with a momentum of 0.9 and a weight decay of 5×10^{-4}. The training is executed for about $40k$ iterations, and a batch size of 16 is used. The learning rate is initialized as 0.01 and decayed by timing $\left(1 - \frac{iter}{max\_iter}\right)^{0.9}$ for every 20 epochs.

IndexNet matting [72] with depthwise IndexNet ("nonlinear+context") is trained on the Adobe Matting dataset [113] following the training disciplines in [72]. MobileNetv2 pre-trained on the ImageNet dataset is used as the backbone as well.

Parameter-Sharing Network: The training of parameter-sharing network is separated into two phases:

Phase 1: We first load parameters of pre-trained DeepLabV3+ to the weight-fixed encoder and the weight-fixed SegDecoder, then load parameters of corresponding layers from pre-trained IndexNet to the MatDecoder. We train the MatDecoder with SGD optimizer, following the settings in Section 6.5.2 with a learning rate of 0.01 for about $40k$ iterations. We randomly crop images into 320×320 blocks as input images and argument them with random scaling, rotation and flipping.

Phase 2: To enable end-to-end training of the whole network, we release parameters of the encoder and the SegDecoder. The same strategies are taken here except initializing the learning rate as 0.0001 and training for around $100k$ iterations. Learning rates in these two phases are all decreased by timing $\left(1 - \frac{iter}{max\_iter}\right)^{0.9}$ for every 20 epochs. A batch size of 16 is utilized.

6.5.3 Definition of Loss Functions

Following [89, 12], when training the baseline network, we use segmentation loss and alpha prediction loss. As for the parameter-sharing network, the loss function covers alpha prediction loss along with the gradient and affinity losses. It is worth noting that we do not use the compositional loss in [72] because there is no ground-truth foreground and background available in the Portrait-2k dataset.

Alpha prediction loss l_a is the average difference between a predicted alpha matte and the ground-truth alpha matte:

$$l_a = \frac{1}{|S|} \sum_{i \in S} \|\alpha_i - {\alpha^*}_i\|_2 \,, \tag{6.5}$$

where α and α^* represent predicted alpha matte and its ground-truth, respectively, and S refers to the regions where loss calculation happens. In our experiments, S indicates the unknown regions U when focusing on unknown regions (l_{a_U}) and the whole image I when targeting l_{a_I}. To calculate the gradient loss l_g, we first apply the Sobel operator [94] to generate gradient map of the alpha matte, and then measure the difference by

$$l_{gx} = |G_{\alpha x} - G_{\alpha^* x}| \ , \tag{6.6}$$

$$l_{gy} = |G_{\alpha y} - G_{\alpha^* y}| \ , \tag{6.7}$$

$$l_g = \frac{1}{S} \sum_{i \in S} l_{gxi} + \frac{1}{S} \sum_{i \in S} l_{gyi} \,. \tag{6.8}$$

Affinity loss l_A is defined in a downsampled image according to Equation (6.4) by

$$l_A = \|A_\alpha - A_I\|_2 \ . \tag{6.9}$$

The loss function of the baseline follows Equation (6.3), and the loss function of the parameter-sharing network is $l = l_{a_U}$ in *Phase 1* and $l = l_{a_U} + \lambda_1 l_{a_I} + \lambda_2 l_g + \lambda_3 l_A$ in *Phase 2*. We use $\lambda_1 = 10$, $\lambda_2 = 10$ and $\lambda_3 = 10$ in our experiments, keeping the value ranges of these losses identical.

6.5.4 Evaluation Metrics

We follow the standard and perceptual evaluation metrics [81] implemented by [113] to evaluate the matting results. The evaluation metrics are Sum of Absolute Differences (SAD), Mean Squared Error (MSE), Gradient Error (Grad) and Connectivity Error (Conn). The first three errors are per-pixel metrics while the last one is a region-level evaluation motivated by human perception. Different from existing works, we evaluate 2 sets of metrics here. Set 1, listed as SAD, MSE, Grad and Conn, are losses calculated in unknown regions, which are commonly used in standard matting benchmarks. However, since all the foreground, background and unknown regions are generated in our network, only evaluating the alpha matte in unknown regions is not sufficient to assess the performance objectively. Hence, we use Set 2, which contains SAD_f, MSE_f, Grad_f and Conn_f, to measure the errors in the full image. We also calculate the number of network parameters (Param) and the sum of Giga Floating-Point Operations Per Second (GFLOPs) on a $224 \times 224 \times 3$ input to score the capacity and complexity of models.

6.5.5 Trained Models in Experiments

Several models, including baselines and our proposed structures, are employed in the experiments. They are:

IndexNet [72] with Ground-Truth Trimap Input (A1): a model with the conventional settings. The inputs are composed by RGB channels and another trimap channel. This baseline can be viewed as the upper bound of our matting baseline.

IndexNet w/o Trimap Input (A2): based on A1, but without trimap input. We implement it by setting the 4_{th} input channel of IndexNet as a map covered by unknown regions.

Trimap-Guided Training of A2 (A2+): an A2 model that uses ground-truth trimap to calculate training loss. This model justifies the effectiveness of end-to-end training with trimap-guided loss in the matting task.

Closed-Form matting (CF) [57] with (A3) / w/o (A4) Ground-Truth Trimap Input: another pair of comparison that uses the non-deep CF method to highlight the importance of trimap input.

DeepLabV3+ + IndexNet (B): a baseline that cascades the pre-trained DeepLabV3+ network and IndexNet network, without updating weights.

	SAD	MSE	Grad	Conn	SAD_f	MSE_f	$Grad_f$	$Conn_f$
GT Trimap + IndexNet [72] (A1)	4.62	0.0278	10.97	4.49	0.0096	0.0038	0.0229	0.0094
No Trimap + IndexNet (A2)	22.21	0.2147	21.96	22.72	0.3465	0.0291	0.0572	0.0481
A2 + Loss-guided training (A2+)	6.55	0.0590	12.03	6.42	0.0136	0.0080	0.0268	0.0134
GT Trimap + CF [57] (A3)	5.37	0.0243	11.18	5.01	0.0112	0.0033	0.0234	0.0104

TABLE 6.2: Further experiments on verifying prior efficiency. CF means the Closed-Form matting algorithm [57].

End-to-End Training of B (B1): a baseline that implements end-to-end training on B. The outputs of DeepLabV3+ are summed up to one extra channel to function as the trimap input of the IndexNet.

Second End-to-End Training of B (B2): a different baseline implements end-to-end training on B. The difference from B1 is that the outputs of DeepLabV3+ are sent to the IndexNet directly without summation. Therefore, the input of IndexNet here has 6 channels.

Late-Interaction Parameter-Sharing Network (B3): our parameter-sharing design, which shares the encoder of the segmentation network and the matting network.

Feature-Fusion Parameter-Sharing Network (B4): another design sharing encoder of two tasks. This model also introduces feature connections between two decoders.

Attention-Guided Parameter-Sharing Network (B5): an improved design over B4, which investigates a more compact way on how to connect between two decoders.

6.5.6 Ablation Studies

Effectiveness of Prior Input: We first evaluate the effectiveness of the prior input. Apart from the results shown in Table 6.1, we extend the experiment by adding two counterparts and more quantitative results, as shown in Table 6.2. When using the Closed-Form (CF) matting [57] without a prior input (A4), the predicted alpha mattes are almost dark. We may conclude that the prior input is necessary in both deep matting and non-deep matting methods (A1 vs. A2 and A3 vs. A4). A slight difference is that posterior information can compensate for the absence of prior information in deep matting to a large extent, which is backed by A2 vs. A2+.

End-to-End Training of the Baseline: To justify end-to-end training of the cascaded segmentation network and matting network, we perform experiments on the baseline with (B1)/without (B) end-to-end training (Table 6.3). After end-to-end training, errors of both sets decrease dramatically. End-to-end training hence can improve performance of the baseline obviously (B vs. B1). One reason may be the fine-tuning of the IndexNet on the Portrait-2k dataset, and another reason is the optimization of the segmentation

	SAD	MSE	Grad	Conn	SAD_f	MSE_f	Grad_f	Conn_f	Param	GFLOPs
B	8.13	0.0703	13.06	8.03	0.0230	0.0151	0.0310	0.0227	13.97M	10.98
B1	5.76	0.0511	11.09	5.68	0.0176	0.0124	0.0255	0.0174	13.97M	10.98
B2	5.94	0.0543	11.23	5.94	0.0202	0.0151	0.0258	0.0202	13.97M	10.98

Table 6.3: Results of baselines. B, B1 and B2 represent the baseline without end-to-end training, baseline adding end-to-end training with 1-channel trimap input and baseline adding end-to-end training with 3-channel trimap input, respectively.

network. We also observe from Table 6.3 that the 1-channel trimap M in Figure 6.2 outperforms the 3-channel one (B1 vs. B2), which indicates that summing the outputs of segmentation to be 1-channel may benefit the training stage.

In Figure 6.7, we show some qualitative results, among which B generates full details. A possible reason is that the IndexNet pre-trained on the Adobe Matting dataset has strong ability to capture details. After end-to-end training, B1 fills the gaps of segmentation in B (e.g., the second row in Figure 6.7), but details are weakened to some extent. It is probably because ground-truth alpha mattes in the Portrait-2k dataset are not of sufficient details as the Adobe Matting dataset.

Parameter-Sharing Design: To evaluate our parameter-sharing designs, in Table 6.4, we report results of the late-interaction network (B3), the feature-fusion network (B4) and the attention-guided network (B5) in terms of their errors and complexity.

Late fusion matting (LFM) [123] in Table 6.4 is a latest trimap-free matting method. Here we use its released model directly because it is trained on large amount of human and portrait data. B1 is one of the baselines introduced in last subsection. Compared with B1, B3 reduces around 50% parameters and 40% GFLOPs, and the FPS measured on the GTX 1080 Ti is twice faster as well. This complexity reduction mainly arises from the shared encoder of two tasks. Accompanying with the significant model capacity reduction, compared with B1, quantitative results of B3 decrease only 3% $\sim$ 6% for the first set of evaluation metrics, and 7% $\sim$ 10% for the second set of metrics. Among these metrics, SAD, MSE, SAD_f and MSE_f show a certain degree of decrease. It can be because the capacity reduction decreases the learning ability of network on pixel-wise mapping, while gradient and affinity constraints make up for the Grad and Conn errors, especially on the Grad error. It is worth noting that the performance of B3 is similar to that of B4 and B5, even without an explicit prior input. We attribute this to the shared encoder, which learns necessary information for both tasks and then assists the matting task with connotative prior information.

Based on B3, B4 introduces a simple feature connection between SegDecoder and MatDecoder, with 3% parameter growth and neglectable GFLOPs increment. Such a feature connection brings a 0.3% $\sim$ 2.2% decrease in the first set of metrics but a 0.4% $\sim$ 2.9% improvement in the second set of metrics. We may conclude that directly sending the

	SAD	MSE	Grad	Conn	SAD_f	MSE_f	Grad_f	Conn_f	Param	GFLOPs	FPS
LFM [123]	8.41	0.0711	9.73	8.15	0.0440	0.0300	0.0229	0.0413	-	-	-
B1	5.76	0.0511	11.09	5.68	0.0176	0.0124	0.0255	0.0174	13.97M	10.98	21.47
B3	6.08	0.0532	11.02	5.87	0.0191	0.0136	0.0250	0.0186	6.74M	6.77	42.80
B4	6.10	0.0540	11.07	6.00	0.0186	0.0132	0.0249	0.0183	6.96M	6.81	41.98
B5	5.91	0.0518	10.96	5.82	0.0177	0.0124	0.0246	0.0175	5.90M	6.53	44.57
IndexNet [72]	-	-	-	-	-	-	-	-	8.15M	6.30	-
B5+FM	5.89	0.0502	8.86	5.80	0.0172	0.0115	0.0210	0.0169	5.96M	6.54	33.94

Table 6.4: Ablation study of parameter-sharing network design. **B1** represents Baseline; **B3** represents Late-Interaction; **B4** represents Feature-Fusion; and **B5** represents Attention-Guided. Param and GFLOPs are measured on a 224 × 224 × 3 input to compare with IndexNet. FPS is computed on a 512 × 512 × 3 input on the GTX 1080 Ti.

output of SegDecoder to MatDecoder brings more global information, which is useful for overall performance, while it lacks appropriate integration with low-level features. Downsampling the output of SegDecoder before sending it to MatDecoder may also lead to loss of detailed information.

B5 integrates the low-level feature skipping and the feature connections between SegDecoder and MatDecoder. Its parameter and GFLOPs are the least compared with all networks above and even comparable to our matting baseline (IndexNet), which has only one encoder and one decoder. Although it reduces about 15% parameters compared to B4, it performs even better than B4 across all metrics, especially on SAD and MSE. Sending the output of SegDecoder to MatDecoder without downsampling hence is likely to improve the performance because of preserving the generated prior information. Another advantage of B5 is that low-level features are combined layer by layer in a compact manner. We therefore choose B5 as the final parameter-sharing network for the following evaluations. Comparing the qualitative examples exhibited in Figure 6.7, B5 achieves competitive results in both profile and details.

To justify the parameter-sharing design from the point of segmentation, we further evaluate the trimaps generated in our networks. We obtain ground-truth trimaps from binarized ground-truth alpha mattes following the way described in Section 6.5.2. The mean Intersection over Union (mIoU) is used as the evaluation metric. As presented in Table 6.5, we measure mIoU of test results generated by the pre-trained DeepLabV3+ network and our parameter-sharing networks (B3 vs. B4 vs. B5), respectively. Considering the results in Table 6.5, our parameter-sharing designs are able to generate effective prior information, and accuracy of trimap generation is enhanced with the increase of feature connections between two decoders. We may conclude that end-to-end training of two tasks is able to produce more applicable trimaps compared with training segmentation network only, and our parameter-sharing designs are capable of implementing this function efficiently. Some visual results are shown in Figure 6.6. We observe trimaps generated by B5 fit profile of edges better than those directly produced by DeepLabV3+.

	mIoU (%)
DeepLabV3+ [10]	88.32
B3	88.36
B4	88.56
B5	88.65

TABLE 6.5: The trimap generation effect of our parameter-sharing networks.

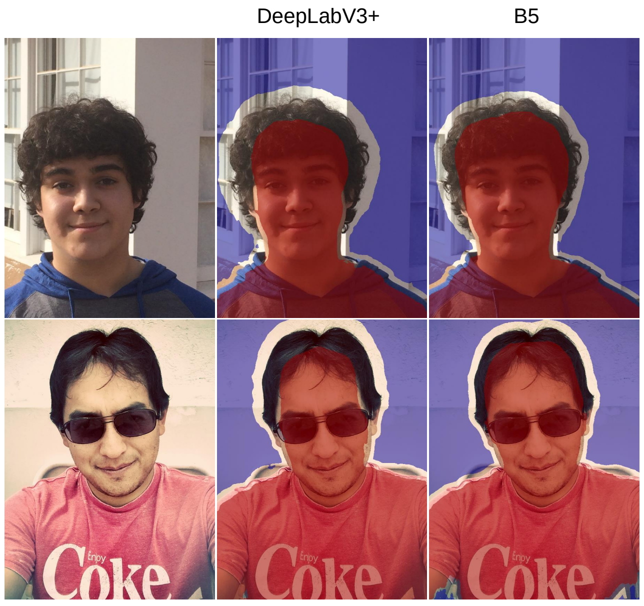


FIGURE 6.6: Visualization of trimap generation. From left to right, the original image, trimap generated by DeepLabV3+ [10] and trimap produced by our B5. For a better view, red, blue and original colors are used to indicate the foreground, background and unknown regions, respectively.

	SAD	MSE	Grad	Conn	SAD_f	MSE_f	$Grad_f$	$Conn_f$
B5	5.91	0.0518	10.96	5.82	0.0177	0.0124	0.0246	0.0175
B5 w/o affinity(B5-)	6.01	0.0533	11.12	5.94	0.0186	0.0132	0.0249	0.0184
B5- w/o gradient(B5–)	6.03	0.0551	11.55	6.01	0.0186	0.0135	0.0261	0.0185

TABLE 6.6: Ablation study of loss function design.

Loss Function: We perform an ablation study on B5 to evaluate the loss functions. As displayed in Table 6.6, we train two models, one without using the gradient loss, and another one without using both gradient and affinity losses. We can observe that the absence of any loss function harms the performance. Visual examples are presented in Figure 6.8, from which we can see an obvious effect of the gradient constraint (by comparing the last two columns). Without the gradient loss, details and edges, like hairs, are visually blurred.

Flexibility of PSPM: To show the flexibility of our parameter sharing design, we replace the segmentation baseline here (DeepLabV3+) with PSPNet [124]. MobileNetv2 is kept as the backbone. We build the end-to-end baseline (B&B1) and implement the attention-guided parameter sharing design (B5) with the same training strategies as in Section 6.5.2. Results are presented in Table 6.7. Both B1 and B5 improve the

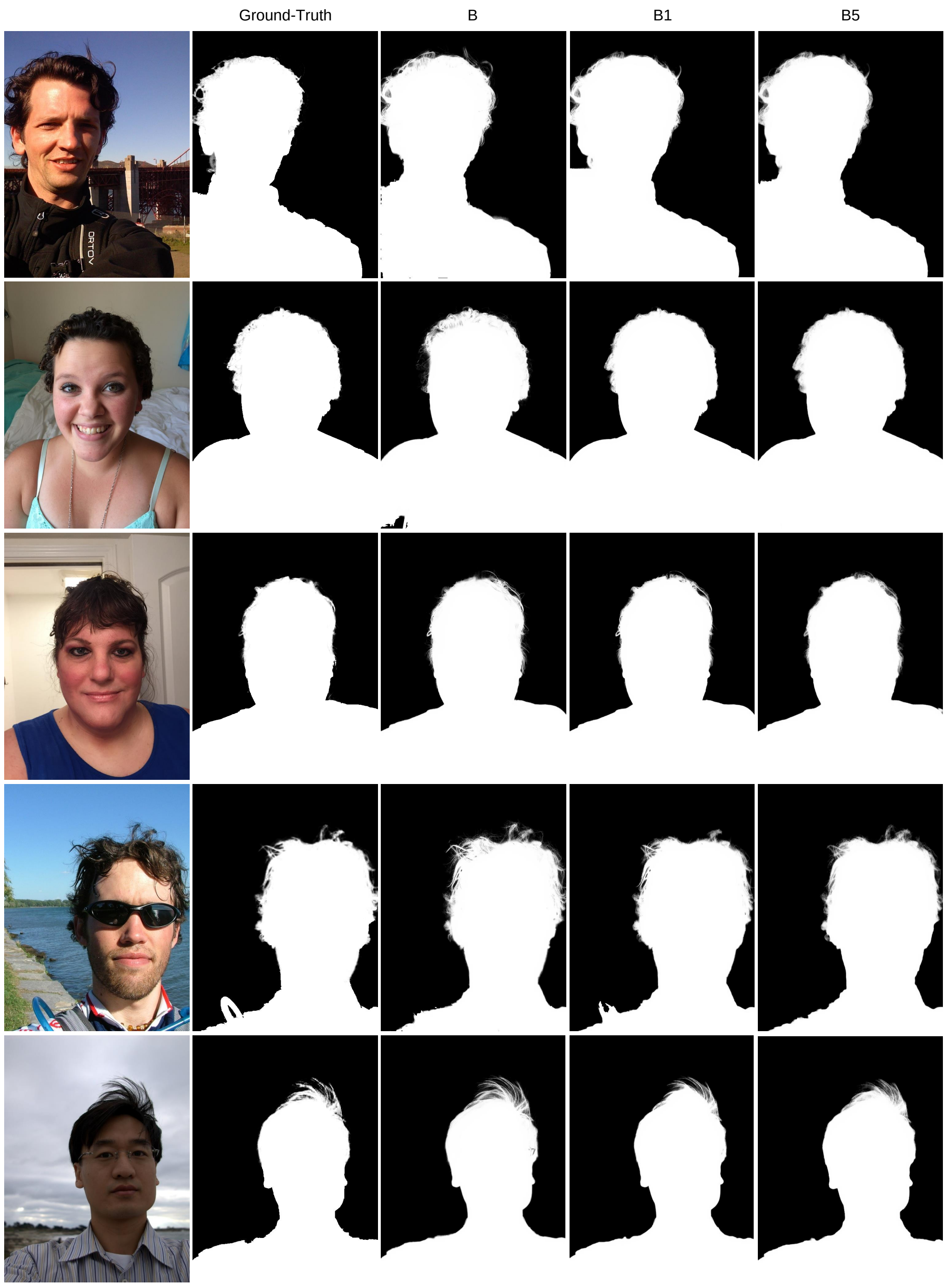


FIGURE 6.7: Qualitative results on the Portrait-2k test set. From left to right, the original image, ground-truth alpha matte, baseline B, baseline B1 and our method B5.

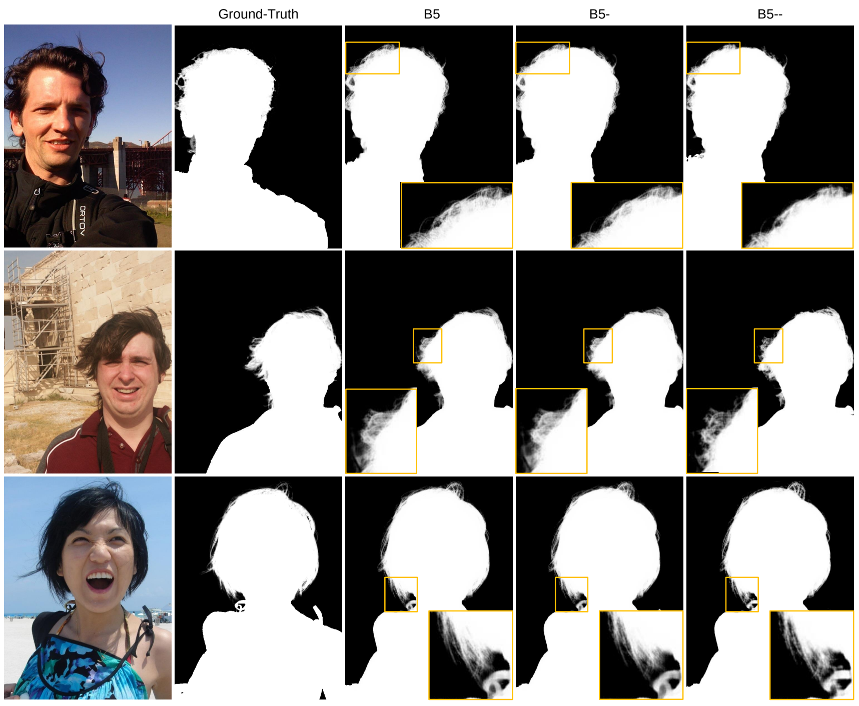


FIGURE 6.8: Qualitative results of the ablation study on loss functions. From left to right, the original image, ground-truth, B5, B5 w/o affinity loss (B5-) and B5- w/o gradient loss (B5–). Details may be viewed by zooming in.

	SAD	MSE	Grad	Conn	SAD_f	MSE_f	Grad_f	Conn_f	Param	GFLOPs	FPS
B	7.41	0.0707	15.55	7.36	0.0221	0.0155	0.0373	0.0220	13.39M	7.29	24.33
B1	6.29	0.0619	13.88	6.26	0.0183	0.0137	0.0325	0.0183	13.39M	7.29	24.33
B5	6.09	0.0513	9.06	5.83	0.0205	0.0143	0.0215	0.0198	5.33M	2.86	49.75

TABLE 6.7: Parameter sharing implemented on PSPNet [124]. Param and GFLOPs are measured on a $224 \times 224 \times 3$ input. FPS is computed on a $512 \times 512 \times 3$ input on the GTX 1080 Ti.

performance and reduce the calculation compared with B. It is worth noting that, compared with Table 6.4, this PSPNet-based architecture has significantly less parameters and GFLOPs. This difference arises from the "last convolution" layer (last-conv) in DeepLabV3+ and PSPNet. In our experiments, the last-conv of DeepLabV3+ consists of three stride-3 convolution layers, while only one stride-1 convolution layer is used in PSPNet. Qualitative results are shown in Figure 6.9.

6.5.7 Multi-task Feature Modulation

Inspired by the task-specific feature modulation [75], in this subsection, we further investigate how the encoder is shared. Different from the comprehensive experiments in previous sections, we only implement the simple idea here to verify the probability of optimizations on our networks. To force the network focusing on more relevant features when operating multiple tasks, a task-dependent gating function modeled on the

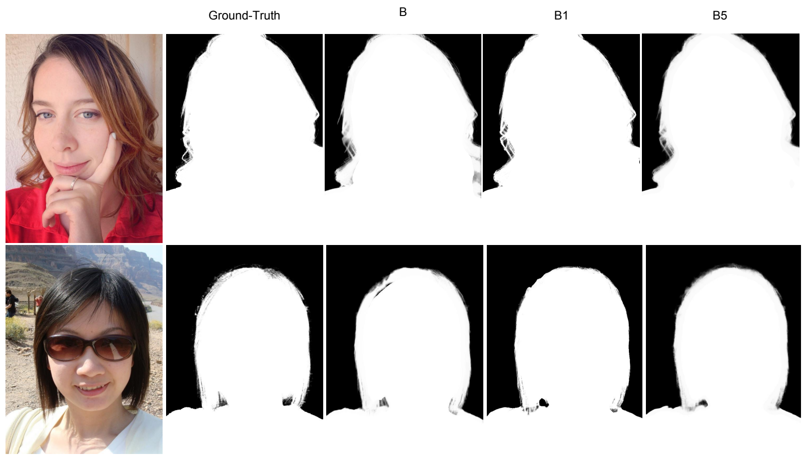


FIGURE 6.9: Qualitative results on the Portrait-2k test set of PSPNet based experiments. From left to right, the original image, ground-truth alpha matte, baseline B, baseline B1 and our method B5.

squeeze-and-excitation (SE) block [46] is used in [75]. In our case, since the two tasks use the same training data and they are operated simultaneously in both training and interference stages, soft combination of all the features rather than hard filtering is preferred.

A simple linear combination is defined by $F_c(x, y, c) = m[c] F(x, y, c)$, where $F(x, y, c)$ is a shared feature tensor, x, y are spatial coordinates and c is the channel. $m[c] \subset (0, 1)$ represents the weight of current channel, which is generated from the feature modulation module. We modify the feature modulation module from [75] by changing the output of SE module from $\mathbb{R}^{b \times c \times 1 \times 1}$ to $\mathbb{R}^{b \times 2c \times 1 \times 1}$, and then shuffling it to $\mathbb{R}^{b \times 2 \times c \times 1 \times 1}$, followed by a softmax operated on the second dimension. We take the first dimension, which in detail is $\mathbb{R}^{b \times c \times 1 \times 1}$, to be the $m[c]$ here. We term the feature modulation operation as "FM". Results of "B5+FM" are reported at the bottom of Table 6.4. With tiny calculation increase, "B5+FM" outperforms B5 on all the evaluation metrics, and even better than our baseline (B1), especially on the gradient error. Since we apply the FM module to all the layers, inference is much slower, however.

6.5.8 Discussion

Deep portrait matting, as an important branch of deep matting, has shown its practicability by previous cases. Application examples of portrait matting are shown in Figure 6.10. Currently, popular solutions of portrait matting are to generate a prior in the network automatically and then guide the following layers to implement matting [89, 12, 87, 128, 45]. These efforts divide portrait matting into two parts in the system level: prior generation and image matting. The specific application scenario and the particular

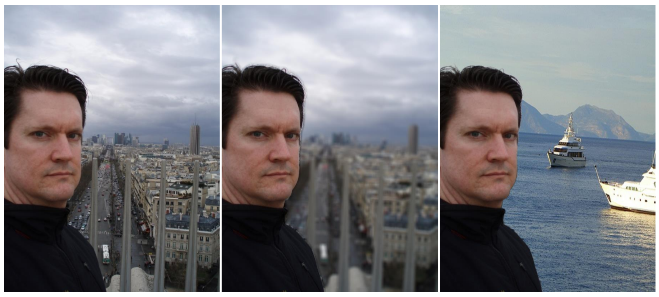


Figure 6.10: Applications of portrait matting using our proposed method. From left to right, the original image, depth-of-field, and background substitution.

pattern of human body motivate researchers to resort to segmentation and then generate unknown regions according to segmentation results or common senses.

However, those methods provide precise details at a cost of redundant structures, while those targeting mobile devices with reduced parameters output relative coarse matting results. Therefore, combining the state-of-the-art segmentation network and matting network, then further reducing parameters of the network is our motivation. We first cascade DeepLabV3+ and IndexNet to perform end-to-end training, then apply the idea of hard parameter sharing multi-task learning to share the encoder of two networks. Specifically, we explore three possible structures constructing connections between two decoders. These parameter sharing structures significantly reduce parameters and calculation costs compared with conventional cascaded structures with only slightly downgraded quantitative and acceptable qualitative performance. More important, our work poses a fundamental question about what is the best way to combine the automatically generated prior with image matting. Further simple feature modulation in the shared encoder enable our multi-task model to generate competitive results with the baseline, which promises a direction for future optimization.

6.6 Conclusion

We highlight the limitation of current automatic portrait matting that builds upon cascaded segmentation models and matting models. To address this, we propose and investigate three architectures to implement parameter-sharing networks. By sharing parameters and exploring feature connections between two tasks, we achieve a notable decrease of computation complexity, with almost the same matting performance. We also remark that trimaps indicating the foreground, background and unknown regions are not the only form of prior input. There may exist other efficient ways that can merge generated prior information to matting layers. The presented structure can provide a strong baseline for portrait matting.

Chapter 7

Conclusion

In this chapter, we summarize our key contributions and also suggest possible research directions for future works.

With the rapid development of deep learning, image matting has experienced remarkable changes in recent years. Different from traditional methods that rely heavily on low-level color cues, deep matting methods use more high-level and long-range information. It promises significant improvements against traditional methods. However, though better performance has been observed, there are still many hard cases, such as transparent objects and nets in rich details. Better performed methods are therefore needed. Besides the performance, efficiency and compactness of deep models are also vital because of the hardware restrictions in real applications. To achieve these goals, we explore efficient model designs for image matting.

Efficient deep image matting. Aiming at more precise prediction as well as efficient model designs, we propose IndexNet matting [72, 71], an indices-guided encoder-decoder architecture, which is more accurate than the baseline model with much less model capacity. We also present the extension of IndexNet on three other dense prediction tasks, including image denoising, semantic segmentation, and depth estimation. Based on a similar encoder-decoder architecture, we further propose A^{2}U matting [22], where upsampling operators are learned from second-order affinity information. It is implemented by low-rank bilinear pooling, showing more precise prediction with neglectable parameter increase. Moreover, We raise RMat [24] to enhance the robustness of deep image matting. It is achieved by a framework modeling multilevel context information and strong augmentation strategies. RMat not only presents top-performing results on the Composition-1k but also shows more robust predictions on various benchmarks, real-world images, and coarse-to-fine trimap precision.

Efficient deep portrait matting. Besides image matting, we propose a lightweight portrait matting method named PSPM [23], where the architecture with a shared encoder and task-specific decoders is firstly proposed for portrait matting. It not only facilitates a lightweight model but also encourages better performance than conventional cascaded architectures.

Future work. i) IndexNet and A^{2}U demonstrate designs for efficient matting models by exploring indices and affinity information, respectively. Their main focus is improving the accuracy in an efficient manner by investigating the task-specific properties. Apart from them, there are many unexplored properties, such as gradient differences between the foreground and the background, color varies at the edge, etc. ii) RMat moves a step further to explore the robustness of the matting model by assembling context information and applying strong data augmentation. It is an initial exploration and there could be more solutions. For example, a better data generation pipeline, generative models, and more effective training strategies are all likely to benefit the robustness of matting models. iii) Due to the attention to detail, existing matting models infer with original image resolutions, which consumes a lot of computation and limits the size of test images. Though patch-based high-resolution image matting has been studied, efficient methods operated on original inputs are still in demand. iv) RMat is one of the earliest works applying the transformer to matting. There is much space to improve current transformer-based matting networks. For instance, existing architectures treat the whole image the same, except including the trimap in the inputs. However, for trimap-based matting, precise prediction of unknown regions is the target. Using this nature may help design more efficient transformer architectures for matting. v) Further, prior-free matting, especially on general images, is a meaningful but challenging topic. How to formulate the problem effectively is an unanswered question. We may either restrict the application scenarios or decompose the problem. A formal and unified formulation would light up future research.

Bibliography

[1] Y. Aksoy, T.-H. Oh, S. Paris, M. Pollefeys, and W. Matusik. “Semantic soft segmentation”. In: *ACM Trans. Graphics* 37.4 (2018), pp. 1–13.

[2] Y. Aksoy, T. Ozan Aydin, and M. Pollefeys. “Designing effective inter-pixel information flow for natural image matting”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2017, pp. 29–37.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. “SegNet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.12 (2017), pp. 2481–2495.

[4] S. Cai, X. Zhang, H. Fan, H. Huang, J. Liu, J. Liu, J. Liu, J. Wang, and J. Sun. “Disentangled image matting”. In: *Proc. IEEE Int. Conf. Comp. Vis.* 2019, pp. 8819–8828.

[5] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. “End-to-end object detection with transformers”. In: *Proc. Eur. Conf. Comp. Vis.* Springer. 2020, pp. 213–229.

[6] C. Chen, Q. Chen, J. Xu, and V. Koltun. “Learning to see in the dark”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2018, pp. 3291–3300.

[7] G. Chen, K. Han, and K.-Y. K. Wong. “Tom-net: Learning transparent object matting from a single image”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2018, pp. 9233–9241.

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.4 (2017), pp. 834–848.

[9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *Proc. Eur. Conf. Comp. Vis.* 2018.

[10] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *Proc. Eur. Conf. Comp. Vis.* 2018, pp. 801–818.

[11] Q. Chen, D. Li, and C.-K. Tang. “KNN matting”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.9 (2013), pp. 2175–2188.

[12] Q. Chen, T. Ge, Y. Xu, Z. Zhang, X. Yang, and K. Gai. “Semantic human matting”. In: *Proc. ACM Int. Conf. Multimedia.* ACM. 2018, pp. 618–626.

[13] X. Chen, D. Qi, and J. Shen. “Boundary-Aware Network for Fast and High-Accuracy Portrait Segmentation”. In: *arXiv preprint arXiv:1901.03814* (2019).

[14] Y. Chen and T. Pock. “Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.6 (2016), pp. 1256–1272.

[15] H. Cheng, S. Xu, X. Jiang, and R. Wang. “Deep Image Matting with Flexible Guidance Input”. In: *British Machine Vis. Conf* (2021).

[16] X. Cheng, P. Wang, and R. Yang. “Depth estimation via affinity learned with convolutional spatial propagation network”. In: *Proc. Eur. Conf. Comp. Vis.* 2018, pp. 103–119.

[17] D. Cho, Y.-W. Tai, and I. Kweon. “Natural image matting using deep convolutional neural networks”. In: *Proc. Eur. Conf. Comp. Vis.* Springer. 2016, pp. 626–643.

[18] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen. “Twins: Revisiting the design of spatial attention in vision transformers”. In: *Adv. Neural Inform. Process. Syst.* 2021.

[19] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. “A bayesian approach to digital matting”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Vol. 2. IEEE. 2001, pp. II–II.

[20] R. Collobert and J. Weston. “A unified architecture for natural language processing: Deep neural networks with multitask learning”. In: *Proc. Int. Conf. Machine. Learn.* ACM. 2008, pp. 160–167.

[21] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. “Deformable convolutional networks”. In: *Proc. IEEE Int. Conf. Comp. Vis.* 2017, pp. 764–773.

[22] Y. Dai, H. Lu, and C. Shen. “Learning Affinity-Aware Upsampling for Deep Image Matting”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2021, pp. 6841–6850.

[23] Y. Dai, H. Lu, and C. Shen. "Towards Light-Weight Portrait Matting via Parameter Sharing". In: *Comput. Graph. Forum.* Vol. 40. 1. Wiley Online Library. 2021, pp. 151–164.

[24] Y. Dai, B. Price, H. Zhang, and C. Shen. "Boosting Robustness of Image Matting with Context Assembling and Strong Data Augmentation". In: *arXiv preprint arXiv:2201.06889* (2022).

[25] S. d'Ascoli, H. Touvron, M. Leavitt, A. Morcos, G. Biroli, and L. Sagun. "Convit: Improving vision transformers with soft convolutional inductive biases". In: *arXiv preprint arXiv:2103.10697* (2021).

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Ieee, 2009, pp. 248–255.

[27] D. Dong, H. Wu, W. He, D. Yu, and H. Wang. "Multi-task learning for multiple language translation". In: *Proc. the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers).* 2015, pp. 1723–1732.

[28] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *Proc. Int. Conf. Learn. Representations* (2020).

[29] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. "The pascal visual object classes (voc) challenge". In: *Int. J. Comput. Vision.* 88.2 (2010), pp. 303–338.

[30] X. Feng, X. Liang, and Z. Zhang. "A cluster sampling method for image matting via sparse coding". In: *Proc. Eur. Conf. Comp. Vis.* Springer. 2016, pp. 204–219.

[31] C. Finn, P. Abbeel, and S. Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *Int. Conf. Mach. Learn.* PMLR. 2017, pp. 1126–1135.

[32] M. Forte and F. Pitié. "F, B, Alpha Matting". In: *CoRR* (2020).

[33] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. "Domain-adversarial training of neural networks". In: *J. Mach. Learn. Research.* 17.1 (2016), pp. 2096–2030.

[34] N. Gao, Y. Shan, Y. Wang, X. Zhao, Y. Yu, M. Yang, and K. Huang. "Ssap: Single-shot instance segmentation with affinity pyramid". In: *Proc. IEEE Int. Conf. Comp. Vis.* 2019, pp. 642–651.

[35] E. S. Gastal and M. M. Oliveira. “Shared sampling for real-time alpha matting”. In: *Computer Graphics Forum*. Vol. 29. 2. 2010, pp. 575–584.

[36] R. Girshick. “Fast r-cnn”. In: *Proc. IEEE Int. Conf. Comp. Vis.* 2015, pp. 1440–1448.

[37] K. He, G. Gkioxari, P. Dollár, and R. Girshick. “Mask r-cnn”. In: *Proc. IEEE Int. Conf. Comp. Vis.* 2017, pp. 2961–2969.

[38] K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun. “A global sampling method for alpha matting”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* IEEE. 2011, pp. 2049–2056.

[39] K. He, J. Sun, and X. Tang. “Fast matting using large kernel matting laplacian matrices”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* IEEE. 2010, pp. 2165–2172.

[40] K. He, J. Sun, and X. Tang. “Guided image filtering”. In: *Proc. Eur. Conf. Comp. Vis.* 2010, pp. 1–14.

[41] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2016, pp. 770–778.

[42] K. He, X. Zhang, S. Ren, and J. Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proc. IEEE Int. Conf. Comp. Vis.* 2015, pp. 1026–1034.

[43] Q. Hou and F. Liu. “Context-aware image matting for simultaneous foreground and alpha estimation”. In: *Proc. IEEE Int. Conf. Comp. Vis.* 2019, pp. 4130–4139.

[44] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv* (2017).

[45] G. Hu and J. J. Clark. “Instance Segmentation based Semantic Matting for Compositing Applications”. In: *Conf. Comp. Rob. Vis.* (2019).

[46] J. Hu, L. Shen, and G. Sun. “Squeeze-and-excitation networks”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2018, pp. 7132–7141.

[47] J. Hu, M. Ozay, Y. Zhang, and T. Okatani. “Revisiting single image depth estimation: toward higher resolution maps with accurate object boundaries”. In: *Proc. IEEE Winter Conf. Applications of Comp. Vis.* 2019, pp. 1043–1051.

[48] M. Jaderberg, K. Simonyan, A. Zisserman, et al. “Spatial transformer networks”. In: *Adv. Neural Inform. Process. Syst.* 2015, pp. 2017–2025.

[49] R. Jaroensri, C. Biscarrat, M. Aittala, and F. Durand. "Generating Training Data for Denoising Real RGB Images via Camera Pipeline Simulation". In: *arXiv* (2019).

[50] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool. "Dynamic filter networks". In: *Adv. Neural Inform. Process. Syst.* 2016, pp. 667–675.

[51] J.-H. Kim, K.-W. On, W. Lim, J. Kim, J.-W. Ha, and B.-T. Zhang. "Hadamard product for low-rank bilinear pooling". In: *Proc. Int. Conf. Learn. Representations.* 2016.

[52] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *Proc. Int. Conf. Learn. Representations.* 2015.

[53] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. "Joint bilateral upsampling". In: *ACM Trans. Graphics* 26.3 (2007), 96–es.

[54] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. "The case for learned index structures". In: *Proc. International Conference on Management of Data.* 2018, pp. 489–504.

[55] Y. LeCun. "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/* (1998).

[56] P. Lee and Y. Wu. "Nonlocal matting". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* IEEE, 2011, pp. 2193–2200.

[57] A. Levin, D. Lischinski, and Y. Weiss. "A closed-form solution to natural image matting". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 30.2 (2007), pp. 228–242.

[58] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. "Learning to generalize: Meta-learning for domain generalization". In: *Proc. AAAI Conf. Artificial Intell.* 2018.

[59] J. Li, J. Zhang, and D. Tao. "Deep Automatic Natural Image Matting". In: *Int. Joint Conf. Artificial Intell.* (2021).

[60] Y. Li and H. Lu. "Natural image matting via guided contextual attention". In: *Proc. AAAI Conf. Artificial Intell.* Vol. 34. 2020, pp. 11450–11457.

[61] Y. Li, J. Zhang, W. Zhao, and H. Lu. "Inductive Guided Filter: Real-time Deep Image Matting with Weakly Annotated Masks on Mobile Devices". In: *arXiv preprint arXiv:1905.06747* (2019).

[62] G. Lin, A. Milan, C. Shen, and I. Reid. "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2017, pp. 1925–1934.

[63] S. Lin, A. Ryabtsev, S. Sengupta, B. L. Curless, S. M. Seitz, and I. Kemelmacher-Shlizerman. "Real-time high-resolution background matting". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2021, pp. 8762–8771.

[64] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2017, pp. 2117–2125.

[65] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft coco: Common objects in context". In: *Proc. Eur. Conf. Comp. Vis.* Springer. 2014, pp. 740–755.

[66] F. Liu, C. Shen, G. Lin, and I. Reid. "Learning depth from single monocular images using deep convolutional neural fields". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 38.10 (2015), pp. 2024–2039.

[67] S. Liu, S. De Mello, J. Gu, G. Zhong, M.-H. Yang, and J. Kautz. "Learning affinity via spatial propagation networks". In: *Adv. Neural Inform. Process. Syst.* 2017, pp. 1520–1530.

[68] Y. Liu, J. Xie, X. Shi, Y. Qiao, Y. Huang, Y. Tang, and X. Yang. "Tripartite information mining and integration for image matting". In: *Proc. IEEE Int. Conf. Comp. Vis.* 2021, pp. 7555–7564.

[69] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2015, pp. 3431–3440.

[70] I. Loshchilov and F. Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[71] H. Lu, Y. Dai, C. Shen, and S. Xu. "Index networks". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2020).

[72] H. Lu, Y. Dai, C. Shen, and S. Xu. "Indices matter: Learning to index for deep image matting". In: *Proc. IEEE Int. Conf. Comp. Vis.* 2019, pp. 3266–3275.

[73] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang. "Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2019, pp. 2507–2516.

[74] S. Lutz, K. Amplianitis, and A. Smolic. "Alphagan: Generative adversarial networks for natural image matting". In: *British Machine Vis. Conf* (2018).

[75] K.-K. Maninis, I. Radosavovic, and I. Kokkinos. "Attentive Single-Tasking of Multiple Tasks". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2019, pp. 1851–1860.

[76] X. Mao, C. Shen, and Y.-B. Yang. "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections". In: *Adv. Neural Inform. Process. Syst.* 2016, pp. 2802–2810.

[77] V. Mnih, N. Heess, A. Graves, et al. “Recurrent models of visual attention”. In: *Adv. Neural Inform. Process. Syst.* 2014, pp. 2204–2212.

[78] A. Odena, V. Dumoulin, and C. Olah. “Deconvolution and checkerboard artifacts”. In: *Distill* 1.10 (2016), e3.

[79] C. Osendorfer, H. Soyer, and P. Van Der Smagt. “Image super-resolution with fast approximate convolutional sparse coding”. In: *Proc. International Conference on Neural Information Processing (ICONIP).* 2014, pp. 250–257.

[80] Y. Qiao, Y. Liu, X. Yang, D. Zhou, M. Xu, Q. Zhang, and X. Wei. “Attention-Guided Hierarchical Structure Aggregation for Image Matting”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2020, pp. 13676–13685.

[81] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott. “A perceptually motivated online benchmark for image matting”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* IEEE. 2009, pp. 1826–1833.

[82] O. Ronneberger, P. Fischer, and T. Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *Proc. Int. Conf. Medical Image Computing and Computer-Assisted Intervention.* Springer. 2015, pp. 234–241.

[83] S. Rota Bulò, L. Porzi, and P. Kontschieder. “In-Place Activated BatchNorm for Memory-Optimized Training of DNNs”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2018.

[84] S. Roth and M. J. Black. “Fields of experts”. In: *Int. J. Comput. Vision.* 82.2 (2009), p. 205.

[85] S. Ruder. “An overview of multi-task learning in deep neural networks”. In: *arXiv preprint arXiv:1706.05098* (2017).

[86] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. “Mobilenetv2: Inverted residuals and linear bottlenecks”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2018, pp. 4510–4520.

[87] S. Seo, S. Choi, M. Kersner, B. Shin, H. Yoon, H. Byun, and S. Ha. “Towards Real-Time Automatic Portrait Matting on Mobile Devices”. In: *arXiv preprint arXiv:1904.03816* (2019).

[88] E. Shahrian, D. Rajan, B. Price, and S. Cohen. “Improving image matting using comprehensive sampling sets”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2013, pp. 636–643.

[89] X. Shen, X. Tao, H. Gao, C. Zhou, and J. Jia. “Deep automatic portrait matting”. In: *Proc. Eur. Conf. Comp. Vis.* Springer. 2016, pp. 92–107.

[90] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. 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.* 2016, pp. 1874–1883.

[91] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. “Indoor segmentation and support inference from rgbd images”. In: *Proc. Eur. Conf. Comp. Vis.* 2012, pp. 746–760.

[92] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *Int. Conf. Learn. Representations.* (2014).

[93] A. R. Smith and J. F. Blinn. “Blue screen matting”. In: *ACM SIGGRAPH.* Vol. 96. 1996, pp. 259–268.

[94] I. Sobel. *Camera models and machine perception.* Tech. rep. Computer Science Department, Technion, 1972.

[95] S. Song, S. P. Lichtenberg, and J. Xiao. “SUN RGB-D: A RGB-D scene understanding benchmark suite”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2015, pp. 567–576.

[96] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum. “Poisson matting”. In: *ACM Trans. Graphics.* Vol. 23. 3. ACM. 2004, pp. 315–321.

[97] Y. Sun, C.-K. Tang, and Y.-W. Tai. “Semantic Image Matting”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2021, pp. 11120–11129.

[98] J. Tang, Y. Aksoy, C. Oztireli, M. Gross, and T. O. Aydin. “Learning-based sampling for natural image matting”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2019, pp. 3055–3063.

[99] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. “Residual attention network for image classification”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2017, pp. 3156–3164.

[100] J. Wang, K. Chen, R. Xu, Z. Liu, C. C. Loy, and D. Lin. “Carafe: Content-aware reassembly of features”. In: *Proc. IEEE Int. Conf. Comp. Vis.* 2019, pp. 3007–3016.

[101] J. Wang and M. F. Cohen. “Optimized color sampling for robust matting”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* IEEE. 2007, pp. 1–8.

[102] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions”. In: *Proc. IEEE Int. Conf. Comp. Vis.* (2021).

[103] Y. Wang, Y. Niu, P. Duan, J. Lin, and Y. Zheng. “Deep Propagation Based Image Matting”. In: *Int. Joint Conf. Artificial Intell.* Vol. 3. 2018, pp. 999–1006.

[104] T. Wei, D. Chen, W. Zhou, J. Liao, H. Zhao, W. Zhang, and N. Yu. "Improved image matting via real-time user clicks and uncertainty estimation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2021, pp. 15374–15383.

[105] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze. "FastDepth: Fast Monocular Depth Estimation on Embedded Systems". In: *Proc. Int. Conf. Robotics and Automation.* 2019.

[106] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon. "CBAM: Convolutional block attention module". In: *Proc. Eur. Conf. Comp. Vis.* 2018, pp. 3–19.

[107] C. Wu, F. Wu, T. Qi, and Y. Huang. "Fastformer: Additive Attention Can Be All You Need". In: *arXiv preprint arXiv:2108.09084* (2021).

[108] H. Wu, S. Zheng, J. Zhang, and K. Huang. "Fast end-to-end trainable guided filter". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2018, pp. 1838–1847.

[109] K. Xian, C. Shen, Z. Cao, H. Lu, Y. Xiao, R. Li, and Z. Luo. "Monocular relative depth perception with web stereo data supervision". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2018, pp. 311–320.

[110] H. Xiao, K. Rasul, and R. Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms". In: *arXiv preprint arXiv:1708.07747* (2017).

[111] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun. "Unified perceptual parsing for scene understanding". In: *Proc. Eur. Conf. Comp. Vis.* 2018, pp. 418–434.

[112] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers". In: *Adv. Neural Inform. Process. Syst.* (2021).

[113] N. Xu, B. Price, S. Cohen, and T. Huang. "Deep image matting". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2017, pp. 2970–2979.

[114] Z. Xu, D. Liu, J. Yang, C. Raffel, and M. Niethammer. "Robust and generalizable visual representation learning via random convolutions". In: *Proc. Int. Conf. Learn. Representations* (2020).

[115] B. Yang, G. Bender, Q. V. Le, and J. Ngiam. "Condconv: Conditionally parameterized convolutions for efficient inference". In: *Adv. Neural Inform. Process. Syst.* 2019, pp. 1307–1318.

[116] T.-J. Yang, M. D. Collins, Y. Zhu, J.-J. Hwang, T. Liu, X. Zhang, V. Sze, G. Papandreou, and L.-C. Chen. "DeeperLab: Single-Shot Image Parser". In: *arXiv* (2019).

[117] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang. “Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation”. In: *Int. J. Comput. Vision.* 129.11 (2021), pp. 3051–3068.

[118] C. Yu, X. Zhao, Q. Zheng, P. Zhang, and X. You. “Hierarchical bilinear pooling for fine-grained visual recognition”. In: *Proc. Eur. Conf. Comp. Vis.* 2018, pp. 574–589.

[119] Q. Yu, J. Zhang, H. Zhang, Y. Wang, Z. Lin, N. Xu, Y. Bai, and A. Yuille. “Mask Guided Matting via Progressive Refinement Network”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2021, pp. 1154–1163.

[120] M. D. Zeiler and R. Fergus. “Visualizing and understanding convolutional networks”. In: *Proc. Eur. Conf. Comp. Vis.* 2014, pp. 818–833.

[121] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. “mixup: Beyond empirical risk minimization”. In: *Proc. Int. Conf. Learn. Representations* (2017).

[122] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. “Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising”. In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.

[123] Y. Zhang, L. Gong, L. Fan, P. Ren, Q. Huang, H. Bao, and W. Xu. “A Late Fusion CNN for Digital Matting”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2019, pp. 7469–7478.

[124] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. “Pyramid scene parsing network”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2017, pp. 2881–2890.

[125] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr, et al. “Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2021, pp. 6881–6890.

[126] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. “Random erasing data augmentation”. In: *Proc. AAAI Conf. Artificial Intell.* Vol. 34. 07. 2020, pp. 13001–13008.

[127] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. “Scene Parsing through ADE20K Dataset”. In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* 2017.

[128] B. Zhu, Y. Chen, J. Wang, S. Liu, B. Zhang, and M. Tang. “Fast deep matting for portrait animation on mobile phone”. In: *Proc. ACM Int. Conf. Multimedia.* ACM. 2017, pp. 297–305.