Generalizing Interactive Backpropagating Refinement for Dense Prediction Networks

Fanqing Lin
Brigham Young University
fanqinglin@byu.edu

Brian Price
Adobe Research
bprice@adobe.com

Tony Martinez
Brigham Young University
martinez@cs.byu.edu

Abstract

As deep neural networks become the state-of-the-art approach in the field of computer vision for dense prediction tasks, many methods have been developed for automatic estimation of the target outputs given the visual inputs. Although the estimation accuracy of the proposed automatic methods continues to improve, interactive refinement is oftentimes necessary for further correction. Recently, feature backpropagating refinement scheme [25] (f-BRS) has been proposed for the task of interactive segmentation, which enables efficient optimization of a small set of auxiliary variables inserted into the pretrained network to produce object segmentation that better aligns with user inputs. However, the proposed auxiliary variables only contain channel-wise scale and bias, limiting the optimization to global refinement only. In this work, in order to generalize backpropagating refinement for a wide range of dense prediction tasks, we introduce a set of G-BRS (Generalized Backpropagating Refinement Scheme) layers that enable both global and localized refinement for the following tasks: interactive segmentation, semantic segmentation, image matting and monocular depth estimation. Experiments on SBD, Cityscapes, Mapillary Vista, Composition-1k and NYU-Depth-V2 show that our method can successfully generalize and significantly improve performance of existing pretrained state-of-the-art models with only a few clicks.

1. Introduction

Deep learning has revolutionized the task of dense prediction, allowing a breakthrough for pixel-classification problems such as semantic segmentation [2, 17, 18, 36] and pixel-regression problems such as depth estimation [3, 4, 7, 13]. While these automatic methods are constantly improving in performance, a user has no resource to make corrections on the estimated output other than using external tools that do not leverage any learned features. To enable user interactions, dense prediction tasks such as interactive segmentation [9, 14, 16, 21, 30] and image matting [1, 5, 20, 29, 35] use user inputs in forms of distance maps and trimap respectively as network input. Although the additional information can be helpful during forward propagation, deep networks are still free to generate predictions inconsistent with the user-provided inputs.

In this work, we investigate whether a pretrained automatic dense prediction method can be effectively converted into an efficient interactive method without any additional retraining. This is a significant task as deep networks are commonly applied in interactive ways for photography [11,32,34,37,38], videography [22,23,31], special effects [6,8,28], etc. Two prior works, both focused primarily on interactive segmentation, have inspired our method. Backpropagating Refinement Scheme (BRS) [12] performs interactive segmentation using an initial forward pass given the input image and distance maps generated from a set of clicks as in [30]. To additionally refine the prediction and encourage consistency with the input clicks, it sets the input distance maps as the trainable parameters and performs backpropagation using loss computed from the prediction and the clicked labels. BRS also briefly extends this idea to a few other applications: semantic segmentation, saliency detection and medical image segmentation, showing potential use of BRS for CNNs in general. A follow-on work, f-BRS [25] later argues that due to the need for online backpropagation through the entire network, BRS has slow inference speed and is computationally expensive. To this end, instead of using the input distance maps as trainable parameters, f-BRS inserts a pair of auxiliary parameters that act as channel-wise scale and bias after an intermediate network layer, requiring backpropagation through a subpart of the network while achieving nearly equivalent performance.

Despite the improved efficiency of f-BRS, it comes with a major disadvantage: the proposed auxiliary channel-wise scale and bias are only capable of global modification. This not only neglects the need for localized refinement in many vision applications, but also makes the modified output susceptible to undesired global changes while correcting for existing clicks. To make efficient and effective refinement

generalized for dense prediction models, we propose to expand the idea of auxiliary channel-wise scale and bias to a set of G-BRS (Generalized Backpropagating Refinement Scheme) layers with more advanced layer architectures. Our approach enables both global and localized refinement using a channel-weighted bias map in various settings. In addition, we propose a novel consistency loss with an attention mechanism that stabilizes the refinement process and enables more user control. To demonstrate the generality of our approach, we implement G-BRS on four state-of-the-art models for a wide range of dense prediction tasks including interactive segmentation, semantic segmentation, image matting and depth estimation. We perform thorough evaluation on five benchmark datasets: SBD, Cityscapes, Mapillary Vista, Composition-1k and NYU-Depth-V2. Results show that our method enables existing models to achieve significant improvement with interactive clicks and opens up promising directions for equipping automatic methods with interactive features in general.

2. Method

2.1. Background

Backpropagating refinement scheme. BRS was initially proposed by Jang *et al*. [12] for interactive segmentation, which is a task to segment the foreground object and the background given the user inputs. First, the input clicks are used to generate the foreground and background interaction map using distance transform. At inference time, the input image concatenated with the interaction maps is forward propagated in a CNN to produce an output segmentation. Although information of the clicked locations are encoded in the input interaction maps, it is possible that the annotated locations are still mislabeled in the output segmentation. To address this issue, BRS proposes to use backpropagation to refine the input interaction maps to enforce consistency between the input clicks and output segmentation. An alternative method of finetuning the entire model is not ideal since it is computationally inefficient and the model would lose the pretrained knowledge needed for intelligent refinement. With the network defined as f, given a set of input clicks $\{(u_i, v_i, l_i)\}_{i=1}^{n}$ where (u, v) and $l \in \{0, 1\}$ denote the clicked location and label respectively, BRS refines the initial interaction maps x by solving for Δx in the following optimization problem:

$$E(x) = \min_{\Delta x} \left(\lambda \|\Delta x\|_2 + \sum_{i=1}^{n} (f(x + \Delta x)_{u_i, v_i} - l_i)^2 \right). \tag{1}$$

The first term represents the inertial energy used to prevent excessive modification, where λ is a scaling constant that regulates the trade-off. The second term represents the corrective energy used to enforce correct output segmentation at the clicked locations.

Feature Backpropagating Refinement Scheme. Despite the improvement in accuracy, BRS is computationally expensive as it requires gradient computation through the entire network. Consequently, Sofiiuk *et al*. [25] propose f-BRS to modify a small set of inserted auxiliary parameters instead of the input interaction maps, leading to a faster algorithm that requires gradient computation through only a small part of the network. It defines $\hat{f}(x, p)$ as the function that accepts the additionally inserted auxiliary parameter p. The optimization problem is then presented as the following:

$$E(x) = \min_{\Delta x} \left(\lambda \|\Delta p\|_2 + \sum_{i=1}^{n} (\hat{f}(x, p + \Delta p)_{u_i, v_i} - l_i)^2 \right). \tag{2}$$

To avoid minor localized refinement near the clicked locations and encourage global refinement, Sofiiuk *et al*. propose to use channel-wise scale $s \in \mathbb{R}^{\mathcal{C}}$ and bias $b \in \mathbb{R}^{\mathcal{C}}$ as the auxiliary parameters, where $\mathcal{C}$ denotes the number of channels for the corresponding intermediate feature map of the network. Let us define the inserted auxiliary parameters as a G-BRS layer. The proposed layer that utilizes channel-wise scale and bias can then be formulated as,

$$\mathcal{G}_{sb}(m) = m \dot{\times} s \dot{+} b \tag{3}$$

where $m \in \mathbb{R}^{\mathcal{H} \times \mathcal{W} \times \mathcal{C}}$ is the intermediate feature map with $\mathcal{H}, \mathcal{W}$ and $\mathcal{C}$ denoting the height, width and number of channels respectively. Channel-wise multiplication and addition are represented as $\dot{\times}$ and $\dot{+}$. Since the inserted G-BRS layer should not interfere with the initial network prediction, its initial parameters need to perform the identity operation such that $\mathcal{G}_0(m) = m$. This can be fulfilled with the initialization of $s_0 = \mathbf{1}$ and $b_0 = \mathbf{0}$. We will refer to this G-BRS layer as the G-BRS-sb layer.

2.2. Global and Localized Refinement

As the G-BRS-sb layer enables channel-wise scaling and shifting of the original feature maps, it solely focuses on global refinement since s and b are invariant to position in the selected feature. This limitation can result in unstable and undesirable effects as an attempt to fix a localized error could lead to unpredictable global changes across the image. To additionally enable positional modification of the selected feature map for precise localized refinement, we propose three novel G-BRS layer architectures with better performance on numerous applications below.

First, we introduce the G-BRS-bmsb layer that contains an additional bias map $b_m \in \mathbb{R}^{\mathcal{H} \times \mathcal{W}}$ prior to the channel-wise scale and bias. To enable all channels of the feature map to shift freely in different directions, we also introduce a channel weight variable $w_c \in \mathbb{R}^{\mathcal{C}}$ to perform channel-wise scaling for the bias map. We formulate the

G-BRS-bmsb layer as follows:

$$\mathcal{G}_{bmsb}(m) = (m + (b_m \dot{\times} w_c)) \dot{\times} s \dot{+} b \tag{4}$$

where $(b_m \dot{\times} w_c) \in \mathbb{R}^{\mathcal{H}\times\mathcal{W}\times\mathcal{C}}$. Similar to s and b, we initialize b_m as $\mathbf{0}$ and w_c as $\mathbf{1}$. Since the size of b_m depends on the resolution of the selected feature map, we apply the G-BRS insertion(s) in deeper feature space where the feature resolution is a fraction of the output resolution. This setting also prevents the aforementioned drawback that leads to trivial localized refinement.

As the channel-weighted bias map and the channel-wise scale and bias apply localized and global changes respectively, the G-BRS-bmsb layer modifies the input feature through the two variables sequentially. To explore feature fusion where the G-BRS layer merges feature maps from the global branch and the localized branch, we introduce the G-BRS-bmsb-m layer formulated as follows:

$$\begin{aligned} g_1(m) &= m \dot{\times} s \dot{+} b \\ g_2(m) &= m + (b_m \dot{\times} w_c) \\ \mathcal{G}_{bmsb-m}(m) &= w \cdot g_1(m) + (1 - w) \cdot g_2(m) \end{aligned} \tag{5}$$

where $w \in [0, 1]$ is a learnable parameter (initialized to 0.5) used to regulate the trade-off between global and localized changes in the input feature.

In addition to the channel-wise scale and bias, we explore a more powerful representation by replacing s and b with a convolutional layer, which we refer to as the G-BRS-bmconv layer. For kernel size $k = 1$, the convolutional layer essentially learns to combine features from different input channels for each output channel. With $\mathcal{C}_{in} = \mathcal{C}_{out}$, we initialize the kernel weight $w_{conv} \in \mathbb{R}^{\mathcal{C}\times\mathcal{C}\times1\times1}$ as an identity matrix and the bias $b_{conv} \in \mathbb{R}^{\mathcal{C}}$ as $\mathbf{0}$. Initially, each output channel represents exactly the corresponding input channel and $\mathcal{G}_{bmconv}(m) = m$. We formulate the G-BRS-bmconv layer as follows:

$$\mathcal{G}_{bmconv}(m) = (m + \beta(b_m \dot{\times} w_c)) \cdot w_{conv} \dot{+} b_{conv} \tag{6}$$

where the 1×1 convolutional operation is represented as matrix multiplication and channel-wise bias. $\beta = 10$ is used as a scalar for amplifying the gradient of the bias map.

2.3. Attention Mechanism

For optimization using backpropagating refinement, intelligent refinement without inaccurate excessive modification is important. Previous methods [12, 25] propose to rely on the minimization of the inertial energy $\lambda\|\Delta p\|_2$. Instead of simply enforcing a small $\|\Delta p\|_2$, we propose to punish excessive perturbation in the output estimation outside of a user-defined attention region, which becomes achievable with the proposed G-BRS layer capable of both global and localized feature map modification. In the following sections, we define each input click as (u, v, r, l) with r and l denoting the attention radius centered at (u, v) and the target label respectively. We introduce a consistency loss with the following general formulation:

$$\mathcal{L}_c = \lambda\, \mathcal{E}((\hat{f}(x, p_{prev}) - \hat{f}(x, p))\, \mathcal{M}) \tag{7}$$

where $\mathcal{E}$ is a function that computes the pixel-wise error using the current prediction $\hat{f}(x, p)$ and the initial prediction $\hat{f}(x, p_{prev})$ with p_{prev} denoting the auxiliary variables from the previous click. $\mathcal{M}$ represents a pixel-wise scaling mask generated using the newest click, which selects the region outside of the r for error computation. In all our experiments, we perform backpropagation for $I = 20$ iterations.

2.4. Generalization

In this work, we use existing pretrained state-of-the-art architectures for a wide range of dense prediction problems. The selected applications include binary-label (interactive segmentation) and multi-label (semantic segmentation) pixel-wise segmentation tasks, bounded (interactive image matting) and unbounded (depth estimation) pixel-wise regression tasks. Our goal is to demonstrate the generality of our approach in both interactive and automatic settings for dense prediction models.

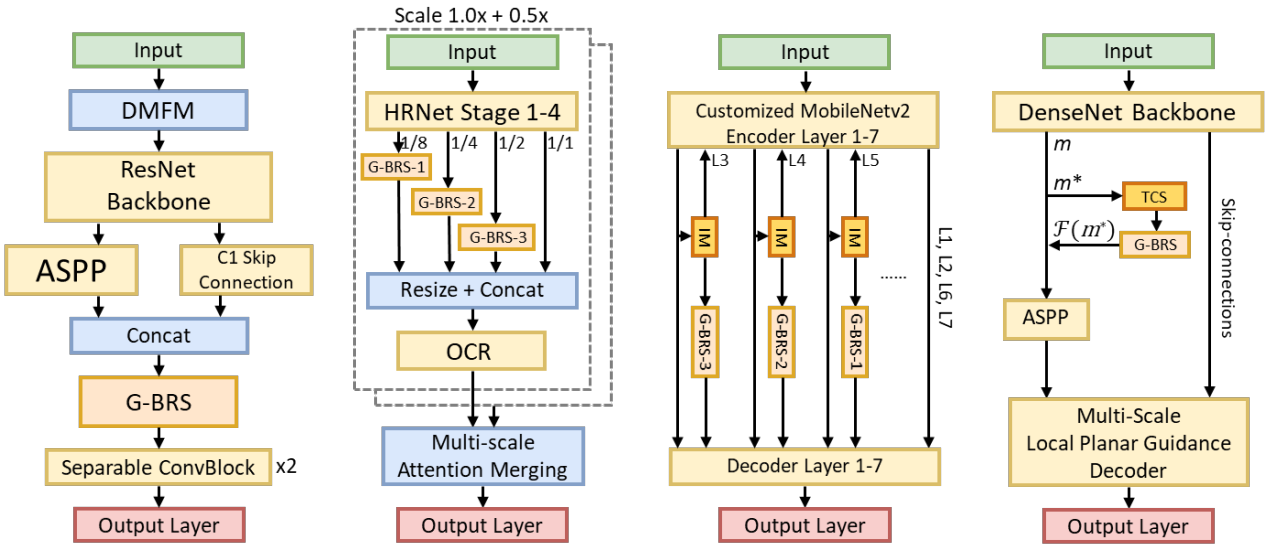
We introduce the corresponding G-BRS layer configuration for each architecture. Options for multiple G-BRS layer insertions are explored to leverage combination of feature modification at different levels. Since architectures for different tasks also drastically differ, it is worth mentioning that designing an effective G-BRS layout requires thought and experimentation to obtain optimal performance.

2.4.1 Interactive Segmentation

Interactive segmentation is a binary segmentation task that separates any target foreground object and the background using user inputs. Since prior methods [12, 25] primarily focused on this task, we make a direct comparison with the f-BRS [25] (equivalent to G-BRS-sb) layer and use the standard DeepLabV3+ with ResNet-101 and the proposed Distance Maps Fusion Module as the architecture. The G-BRS layer is also inserted at the position shown in Figure 1a, where the best performance is reported by [25]. We formulate the optimization as a minimization problem for the click refinement loss $\mathcal{L}_r$ and the consistency loss $\mathcal{L}_c$:

$$\begin{aligned} \mathcal{L}_r &= \sum_{i=1}^{n} \begin{cases} max(1 - \hat{f}(x, p)_{u_i,v_i}, 0)^2 & l_i = 1 \\ max(1 + \hat{f}(x, p)_{u_i,v_i}, 0)^2 & l_i = -1 \end{cases} \\ \mathcal{L}_c &= \lambda_{is} \frac{1}{HW} \|(\hat{f}(x, p_{prev}) - \hat{f}(x, p))\, \mathcal{M}\|_2^2 \end{aligned} \tag{8}$$

Since the selected architecture produces unbounded output values, $\mathcal{L}_r$ enables backpropagation only on positive clicks



(a) Interactive Segmentation. (b) Semantic Segmentation. (c) Image Matting. (d) Depth Estimation.

Figure 1. G-BRS configurations on four state-of-the-art architectures for various computer vision applications.

with values less than 1 and negative clicks with values larger than -1, allowing positive predictions to exceed 1 and vice versa. $\mathcal{L}_c$ uses the Mean Squared Error (MSE) and punishes excessive output deviation outside of the attention region. $\mathcal{M} \in \{0,1\}^{H\times W}$ defines a binary attention mask with the value of 0 within the circular attention region. We use $\lambda_{is} = 1 \times 10^2$ as the weight for this term.

For each click, the network makes an inference using the updated interaction maps and performs backpropagation. Note that all provided clicks are used for $\mathcal{L}_r$ while only the most recent click is used for $\mathcal{L}_c$. Using all clicks in the computation of $\mathcal{L}_r$ allows correction for the newly provided click without losing knowledge gained from previous clicks. As the threshold for binary segmentation is 0, to avoid overfitting and achieve a faster response time, the refinement does early stopping when $max(\|l_i - \hat{f}(x,p)_{u_i,v_i}\|_1 : i = 1,...,n) < 0.8$.

2.4.2 Semantic Segmentation

Semantic segmentation is a multi-label segmentation task with predefined classes. To enable interactive refinement on the output segmentation, we configure multiple G-BRS layer insertions on the architecture proposed by Tao *et al.* [27], a multi-scale attention network with HRNet-OCR [33] as the backbone. As shown in Figure 1b, we make three insertions in stage 4 of the HRNet backbone [26] for each scale branch, where the feature resolution is $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{8}$ of the input resolution. For practicality in user applications using a single GPU, we omit the branch with 2.0x scale and use two branches with 1.0x and 0.5x scale. In addition, we introduce two refinement modes: the click mode and the stroke mode. First, we formulate the optimization problem for the click mode below:

$$\mathcal{L}_r = \frac{1}{n}\sum_{i=1}^{n} log\frac{e^{\hat{f}(x,p)_{u_i,v_i,c_l}}}{\sum_{c=1}^{C} e^{\hat{f}(x,p)_{u_i,v_i,c}}}$$
$$\mathcal{L}_c = \lambda_{ss}\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} log\frac{e^{\hat{f}(x,p)_{h,w,c_p}}}{\sum_{c=1}^{C} e^{\hat{f}(x,p)_{h,w,c}}} \tag{9}$$

We compute the cross entropy loss for both $\mathcal{L}_r$ and $\mathcal{L}_c$ with $\mathcal{L}_r$ only using the clicked locations. C, c_l and c_p denote the number of classes, the clicked target class and the previously predicted class respectively. c_p is set as the ignored label within the circular attention region for the computation of $\mathcal{L}_c$.

In the stroke mode, we enable the user to draw strokes for different target classes with arbitrary radius and create a finetune mask $\mathcal{T} \in \{0,...,C\}^{H\times W}$, where the value of C is used as the ignored label for initialization. In this mode, we update $\mathcal{L}_r$ in Equation 9 as:

$$\mathcal{L}_r = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} log\frac{e^{\hat{f}(x,p)_{h,w,c_\mathcal{T}}}}{\sum_{c=1}^{C} e^{\hat{f}(x,p)_{h,w,c}}}. \tag{10}$$

For the weighting of $\mathcal{L}_c$, we use $\lambda_{ss} = 10$ for the click mode and $\lambda_{ss} = 1$ for the stroke mode.

2.4.3 Image Matting

Image matting is a task to predict dense alpha matte for the target foreground given the input image and the user-defined trimap. Although interactive refinement can be performed by modifying the trimap, such modification

does not guarantee an output matte consistent with the trimap. More importantly, the input trimap lacks the necessary precision for alpha values as it only contains three labels that denote the foreground, the background and the unsure region. To enable backpropagating refinement, IndexNet [19] with the backbone of MobileNetv2 [24] is selected as the architecture. We observe that the index maps generated by the IndexNet Module (IM) for the decoder layers contain the best features and insert the G-BRS layers as shown in Figure 1c. The optimization problem is formulated as follows:

$$\mathcal{L}_r = \frac{1}{n}\sum_{i=1}^{n}(l_i - \hat{f}(x,p)_{u_i,v_i})^2$$
$$\mathcal{L}_c = \lambda_{mt}\frac{1}{HW}\|(\hat{f}(x,p_{prev}) - \hat{f}(x,p))\ \mathcal{M}\|_2^2 \tag{11}$$

where $l_i \in [0,1]$ represents the target alpha value for click i. MSE loss is computed for $\mathcal{L}_r$ and $\mathcal{L}_c$ with $\mathcal{L}_r$ only using the clicked locations. $\mathcal{L}_c$ punishes perturbation far from the attention region using an element-wise weighting mask $\mathcal{M}$, which is defined using an inverse gaussian kernel at the newest clicked location with $\sigma = r$. $\lambda_{mt} = 1 \times 10^3$ is used as the weight for $\mathcal{L}_c$. We refer to this refinement mode as the click mode.

Since it is challenging for the user to determine the exact alpha value for the target pixels, for practicality, we introduce the push mode that allows the user to left/right click to push the alpha values up/down. We define $l \in \{0,1\}$ for the left/right click and formulate the optimization problem as below:

$$\mathcal{L}_r = \begin{cases} ((\hat{f}(x,p_{prev})_{u,v} + \epsilon) - \hat{f}(x,p)_{u,v})^2 & l = 1 \\ ((\hat{f}(x,p_{prev})_{u,v} - \epsilon) - \hat{f}(x,p)_{u,v})^2 & l = 0 \end{cases} \tag{12}$$

where $\epsilon = 0.1$ denotes the push distance. The push mode contains no memory of previous clicks and omits $\mathcal{L}_c$. Backpropagation is applied for only 1 iteration since the required modification is marginal.

2.4.4 Depth Estimation

Depth estimation is a task to produce an accurate depth map from a single image. To enable interactive refinement, we select BTSNet [15] with the backbone of DenseNet-161 [10] as the architecture. We insert the G-BRS layer after the final DenseNet Block of the encoder as shown in Figure 1d. Since feature map m at this location has a large number of channels $\mathcal{C} = 2208$, applying G-BRS on excessive number of parameters can lead to overfitting on the target clicks. Additionally, a large $\mathcal{C}$ is inefficient for the G-BRS-bmconv layer with the parameter $w_{conv} \in \mathbb{R}^{\mathcal{C}\times\mathcal{C}\times1\times1}$. To this end, we perform top-k channel selection (TCS) that selects the $\mathcal{K} = 256$ channels of m with the highest mean activation for G-BRS. The resulting selected feature map $m^* \in \mathbb{R}^{\mathcal{H}\times\mathcal{W}\times\mathcal{K}}$ is used as the input for the G-BRS layer and the unselected channels in m are not modified. We formulate the optimization problem as below:

$$\mathcal{L}_r = \sum_{i=1}^{n}(l_i - \hat{f}(x,p)_{u_i,v_i})^2 \tag{13}$$

We compute the Sum Squared Error (SSE) for $\mathcal{L}_r$ and formulate $\mathcal{L}_c$ the same as Equation 11 using $\lambda_{de} = 1 \times 10^{-1}$. The push mode is also formulated following Equation 12.

3. Experiments

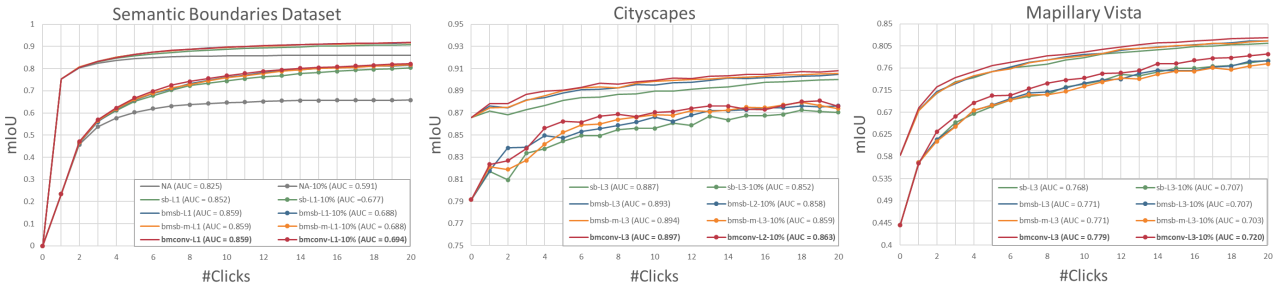
We perform experiments on five benchmark datasets and evaluate on the test/validation sets with publicly available ground truth that enables automatic click generation. We compare the quantitative results of the four types of G-BRS layers. For architectures with multiple G-BRS insertions, we incrementally include insertions for features with higher resolution. In addition to results on the complete test/validation sets, we report results for the 10% of the instances with the lowest initial scores for two reasons: first, since the selected state-of-the-art models can already achieve high average initial accuracy, separate evaluation can better demonstrate the effectiveness of G-BRS on instances with more prominent error. Second, for real-world applications, instances that are high-priority targets for refinement are instances with the worst initial estimation.

For additional analysis, we perform ablation study on the effectiveness of the proposed consistency loss. Since [25] suggested that backpropagating refinement can also be applied using the RGB input as parameters instead of features, we include results using RGB-BRS. Qualitative examples of interactive refinement for all applications are shown in Figure 4. Additional qualitative comparisons between different settings are included in the supplementary document.

3.1. Evaluation Protocol

We compute the standard metrics for all four tasks on each provided click. For a thorough analysis, we additionally compute the following metrics: (1) Area Under Curve (AUC) of the selected metrics to account for convergence time, (2) best score achieved in total number of clicks. We first report results obtained using the consistency loss and provide ablation study in a later section. To find the optimal learning rate for each G-BRS setting, we select a subset of each test set to evaluate using 10 learning rates ranging from 0.1 to 0.1×0.5^9. We report top scores achieved for each type of G-BRS layer and include all experimental results, learning rates used and run time analysis in the supplementary document due to the space limit.

To enable quantitative evaluation of our refinement pro-



(a) Interactive Segmentation (b) Semantic Segmentation - Cityscapes (c) Semantic Segmentation - Mapillary Vista

Figure 2. Quantitative results on interactive segmentation and semantic segmentation using various G-BRS settings with consistency loss. The number of layers that achieve the best scores are reported for each type of layer (e.g. L3 indicates 3 active layers.)

cedures, we use two different automatic click generation strategies. For interactive segmentation and semantic segmentation that requires pixel-wise classification, let us define the binary error mask $\xi_c \in \{0,1\}^{H\times W}$ that represents the misclassified region for class c. We generate the next click with the target label at the location defined below,

$$c^* = \arg\max_c \left(max(\mathcal{D}(\xi_c))\right)$$
$$(u,v) = \arg\max_{u,v} \mathcal{D}(\xi_{c^*}) \tag{14}$$

where $\mathcal{D}$ denotes a distance transform function and c^* is the selected class. Note the region with the ignored label is excluded from the error mask computation. To enable automatic radius generation, we select the connected component ξ_e from ξ_c that contains (u,v) and compute the maximum Euclidian distance between (u,v) and the boundary of ξ_e.

For image matting and depth estimation that requires pixel-wise regression, we use a similar click generation strategy that first transforms the regression error mask to segmentation error mask ξ using Otsu thresholding. Second, as ξ_c can be computed for each class in segmentation tasks, we divide the error mask ξ with positive and negative error into ξ_+ and ξ_-. The clicking location (u,v) can then be generated by following the same strategy as Equation 14. For radius generation, we observe that an insufficient radius is counterproductive as it prevents accurate refinement outside of the small attention region and drastically impacts performance. To this end, we apply dilation with a kernel size of 15 to the selected $\xi_{+/-}$ and compute the radius following the aforementioned strategy for segmentation.

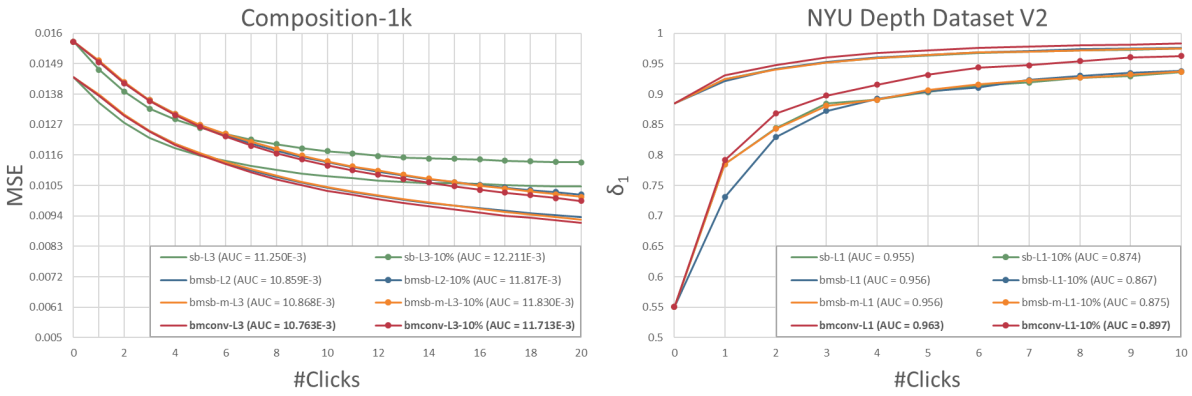
3.2. Evaluation - Interactive Segmentation

We evaluate on the Semantic Boundaries Dataset (SBD), which is currently the largest dataset for interactive segmentation with 2,820 test images and 6,671 instance-level object masks. Since the input clicks that generate the interaction maps also achieve improvement without backpropagating refinement, we run experiments without G-BRS as a baseline comparison. Figure 2a shows the mean Intersection over Union (mIoU) computed over all object instances for 20 clicks on various settings. We compute AUC using mIoU for segmentation tasks. It is shown that the baseline approach (denoted as NA) has limited refinement capability comparing to methods that utilize backpropagating refinement. Note that the G-BRS-sb layer in this task is equivalent to auxiliary variables used in f-BRS [25]. Since f-BRS is not implemented on the other applications we tackle, we refer to this layer archicture as G-BRS-sb in our experiments. Results show that all three G-BRS layers proposed in this work outperform the G-BRS-sb layer (f-BRS), with G-BRS-bmconv layer achieving the top AUC_{mIoU} of 0.859 and 0.694 for the test set and the bottom 10% instances. The G-BRS-bmconv layer also achieves the best peak mIoU obtained in the total number of clicks with a score of 0.918.

Methods	$\mathcal{L}_{brs}$ [12]		$\mathcal{L}_c$ (Ours)	
	AUC	mIoU_{max}	AUC	mIoU_{max}
DistMap-BRS	0.832	0.894	0.845	0.891
RGB-BRS	**0.853**	**0.908**	**0.851**	**0.905**

Table 1. Comparison between refinement settings using the input.

To compare with backpropagating refinement settings that use the input as parameters, we first perform DistMap-BRS [12] that uses the input distance maps as parameters. RGB-BRS that uses the RGB input is also performed, which should be an equivalent solution as suggested by [25]. As the proposed original DistMap-BRS by [12] uses a corrective energy and inertial energy with the L-BFGS optimizer, we refer to this loss minimization method as $\mathcal{L}_{brs}$ and compare it with our method that uses the consistency loss $\mathcal{L}_c$ with the Adam optimizer. Table 1 shows that RGB-BRS outperforms DistMap-BRS and has a slightly higher AUC of 0.853 when using $\mathcal{L}_{brs}$. However, the overall advantage of $\mathcal{L}_{brs}$ greatly diminishes as the L-BFGS optimizer is extremely memory intensive and therefore inapplicable for many applications. As a result, in a later section, we show results obtained using RGB-BRS with $\mathcal{L}_c$ for all applications to compare between backpropagating refinement using the input and the features. For interactive segmentation, Figure 2a shows that all four types of G-BRS layers outperform RGB-BRS.



(a) Image Matting (b) Depth Estimation

Figure 3. Quantitative results on image matting and depth estimation with consistency loss.

3.3. Evaluation - Semantic Segmentation

Since the ground truth of the test sets is not publicly available for automatic click generation, we select the validation sets of Cityscapes and Mapillary Vista for the evaluation of this task. Cityscapes provides 500 test images with 19 classes while Mapillary Vista presents a more challenging task with 2,000 instances and 65 classes. We resize the input resolution for Mapillary Vista to match the area of 1920×960 due to GPU memory constraints. Figure 2b shows that our G-BRS layers outperform the G-BRS-sb layer with the G-BRS-bmconv layer achieving the top AUC_{mIoU} of 0.897 and 0.863 on Cityscapes and its bottom 10% instances. Figure 2c shows that the G-BRS-bmconv layer also achieves the top AUC_{mIoU} of 0.779 and 0.720 on Mapillary Vista and its bottom 10% instances.

We emphasize that our approach is capable of transforming existing state-of-the-art models into interactive methods that further achieve significant improvement. On Cityscapes, as the initial estimation from multi-scale HRNet-OCR [33] already achieves a high mIoU of 0.866, the proposed G-BRS-bmconv layer is able to improve the mIoU to 0.9 in only 10 clicks. For the Mapillary Vista dataset, despite a much lower mIoU of 0.582 from the initial estimation, the G-BRS-bmconv layer achieves a mIoU of 0.822 in 20 clicks, improving the initial score by 41.2%. Additionally, for the bottom 10% instances with a greater need for refinement, we achieve a 77.1% improvement from a mIoU of 0.445 to 0.788 in 20 clicks.

3.4. Evaluation - Image Matting

We evaluate on the Composition-1k, which consists of 1,000 test images composited using 50 unique foreground objects. Standard metrics of Sum of Absolute Differences (SAD), Mean Squared Error (MSE), Gradient (Grad) and Connectivity (Conn) error are included in the supplementary document. For simplicity, we show the MSE for 20 clicks on various settings. Figure 3a shows that the G-BRS-bmconv layer achieves the lowest AUC_{mse} of 10.763×10^{-3} and 11.713×10^{-3} on Composition-1k and its bottom 10% instances. It also decreases the MSE by 36.6% from the initial score of 14.420×10^{-3} to 9.146×10^{-3} in 20 clicks. Our proposed G-BRS layers show a tendency for continuing improvement even after 20 clicks while the G-BRS-sb layer struggles to improve after 10 clicks due to the inability to make localized refinement.

Datasets	sb	bmsb	bmsb-m	bmconv
SBD	0.843	0.832	0.853	0.846
Cityscapes	0.881	0.889	0.886	0.883
Mapillary Vista	0.737	0.742	0.739	0.738
Composition-1k	0.0125	**0.0108**	0.0109	0.0112
NYU-Depth-V2	0.955	**0.962**	0.956	0.955

Table 2. Top AUC using each G-BRS layer type without the consistency loss. Scores that outperform settings using $\mathcal{L}_c$ are in bold.

3.5. Evaluation - Depth Estimation

We evaluate on the test set of NYU-Depth-V2 dataset that consists of 654 RGB-D indoor images. We compute the standard metrics of δ_{1-3}, Abs Rel, Sq Rel, RMSE and RMSE*log* and include all results in the supplementary document. For simplicity, we report results for δ_1 defined as $\delta_t = mean(max(\frac{d_{gt}}{d}, \frac{d}{d_{gt}}) < 1.25^t)$, where d_{gt} and d denote the ground truth and predicted depth map respectively. Figure 3b shows that the G-BRS-bmconv layer achieves the best AUC_δ of 0.963 and 0.897 on the test set and its bottom 10% instances. We improve the initial δ_1 from 0.885 to a near perfect score of 0.983 in 10 clicks. For the bottom 10% instances, there is also a drastic improvement of 74.8% from $\delta_1 = 0.551$ to $\delta_1 = 0.963$.

3.6. Ablation Study

We perform the same experiments for all datasets without using the proposed consistency loss and show the top results in Table 2. By comparing Table 2 with Figure 2 and 3, we show that using consistency loss is beneficial for nearly

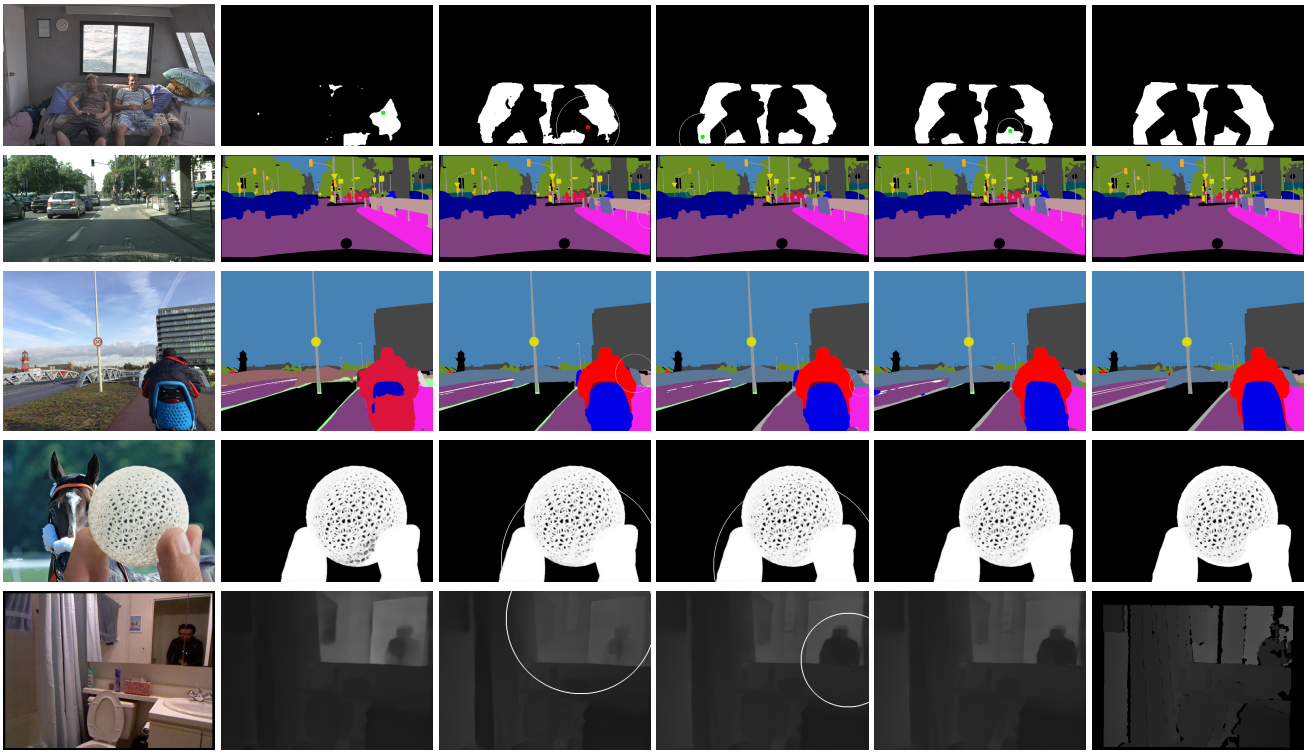


Figure 4. Qualitative examples on SBD, Cityscapes, Mapillary Vista, Composition-1k and NYU-Depth-V2 using G-BRS-bmconv. Clicks with attention radius are visualized. Black region for semantic segmentation and depth estimation is invalid. **Best viewed in magnification.**

Datasets	RGB-BRS		G-BRS-bmconv	
	AUC	SPC	AUC	SPC
SBD	0.851	1.542	0.859	0.584
Cityscapes	0.882	8.361	0.869	5.727
Mapillary Vista	0.675	8.125	0.673	5.130
Composition-1k	0.0100	2.473	0.0108	1.383
NYU-Depth-V2	0.961	3.205	0.963	2.107

Table 3. Comparison between RGB-BRS and G-BRS-bmconv.

all G-BRS settings. Results also show that settings of G-BRS-bmconv that achieve the top AUC for each dataset all utilize $\mathcal{L}_c$. Additional results for experiments with/without $\mathcal{L}_c$ are included in the supplementary document.

3.7. Comparison with RGB-BRS

We perform experiments using RGB-BRS with $\mathcal{L}_c$ for all datasets as discussed in Section 3.2 and measure the AUC as well as the seconds per click (SPC). Experiments for speed measurement are run using a RTX 2080 Ti GPU. Table 3 shows that despite the considerably higher inference time due to the need to backpropagate through the entire network, RGB-BRS and G-BRS-bmconv obtain comparable results. The additional memory consumption for RGB-BRS is also undesirable. For instance, RGB-BRS requires us to downsize the image resolution for semantic segmentation to 1024×512 to fit the memory limit (the same resolution is used for G-BRS-bmconv in this experiment for a fair comparison). As a result, we can see a drop of performance from the top AUC of 0.897 and 0.779 using G-BRS-bmconv (Figure 2) to an AUC of 0.882 and 0.675 using RGB-BRS for Cityscapes and Mapillary Vista respectively. RGB-BRS also has no flexibility for how backpropagating refinement is performed, preventing users from designing effective and efficient G-BRS layouts for different architectures.

4. Conclusion

In this work, we propose a novel set of Generalized Backpropagating Refinement Scheme (G-BRS) layers that bring significant improvement to the performance of state-of-the-art models with both global and localized modification of the intermediate features. By using a user-controlled attention mechanism during refinement, our proposed consistency loss achieves consistent improvement for various G-BRS settings. We show generality of our approach by targeting four different applications and converting the pre-trained state-of-the-art architecture for each application into an interactive method with the corresponding G-BRS layer configuration. Our work shows promising directions for adding interactive capability to architectures used for many other computer vision applications.

References

[1] Shaofan Cai, Xiaoshuai Zhang, Haoqiang Fan, Haibin Huang, Jiangyu Liu, Jiaming Liu, Jiaying Liu, Jue Wang, and Jian Sun. Disentangled image matting. In *ICCV*, 2019. 1

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. In *CVPR*, 2017. 1

[3] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 1

[4] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014. 1

[5] Marco Forte and François Pitié. F, b, alpha matting. In *ECCV*, 2020. 1

[6] Ohad Fried, Ayush Tewari, Michael Zollhöfer, Adam Finkelstein, Eli Shechtman, Dan B Goldman, Kyle Genova, Zeyu Jin, Christian Theobalt, and Maneesh Agrawala. Text-based editing of talking-head video. In *ACM Transactions on Graphics*, volume 38, 2019. 1

[7] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018. 1

[8] Nazim Haouchine, Frederick Roy, Hadrien Courtecuisse, Matthias Nießner, and Stephane Cotin. Calipso: physics-based image and video editing through cad model proxies. In *The Visual Computer*, 2018. 1

[9] Yang Hu, Andrea Soltoggio, Russell Lock, and Steve Carter. A fully convolutional two-stream fusion network for interactive image segmentation. In *Neural Networks*, volume 109, 2019. 1

[10] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 5

[11] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 1

[12] Won-Dong Jang and Chang-Su Kim. Interactive image segmentation via backpropagating refinement scheme. In *CVPR*, 2019. 1, 2, 3, 6

[13] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3DV*, 2016. 1

[14] Hoang Le, Long Mai, Brian Price, Scott Cohen, Hailin Jin, and Feng Liu. Interactive boundary prediction for object selection. In *ECCV*, 2018. 1

[15] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. In *arXiv:1907.10326*, 2019. 5

[16] Jun Hao Liew, Yunchao Wei, Wei Xiong, Sim-Heng Ong, and Jiashi Feng. Regional interactive image segmentation networks. In *ICCV*, 2017. 1

[17] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017. 1

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[19] Hao Lu, Yutong Dai, Chunhua Shen, and Songcen Xu. Indices matter: Learning to index for deep image matting. In *ICCV*, 2019. 5

[20] Sebastian Lutz, Konstantinos Amplianitis, and Aljosa Smolic. Alphagan: Generative adversarial networks for natural image matting. In *BMVC*, 2018. 1

[21] Sabarinath Mahadevan, Paul Voigtlaender, and Bastian Leibe. Iteratively trained interactive segmentation. In *BMVC*, 2018. 1

[22] Jiaxu Miao, Yunchao Wei, and Yi Yang. Memory aggregation networks for efficient interactive video object segmentation. In *CVPR*, 2020. 1

[23] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Fast user-guided video object segmentation by interaction-snd-propagation networks. In *CVPR*, 2019. 1

[24] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: inverted residuals and linear bottlenecks. In *CVPR*, 2018. 5

[25] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *CVPR*, 2020. 1, 2, 3, 5, 6

[26] Ke Sun, Yang Zhao, Borui Jiang, Tianheng Cheng, Bin Xiao, Dong Liu, Yadong Mu, Xinggang Wang, Wenyu Liu, and Jingdong Wang. High-resolution representations for labeling pixels and regions. In *arXiv:1904.04514*, 2019. 4

[27] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. In *arXiv:2005.10821*, 2020. 4

[28] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: image synthesis using neural textures. In *ACM Transactions on Graphics*, volume 38, 2019. 1

[29] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *CVPR*, 2017. 1

[30] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas Huang. Deep interactive object selection. In *CVPR*, 2016. 1

[31] Rui Xu, Xiaoxiao Li, Bolei Zhou, and Chen Change Loy. Deep flow-guided video inpainting. In *CVPR*, 2019. 1

[32] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Free-form image inpainting with gated convolution. In *ICCV*, 2019. 1

[33] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *ECCV*, 2020. 4, 7

[34] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros. Real-time user-guided image colorization with learned deep priors. In *SIGGRAPH*, 2017. 1

[35] Yunke Zhang, Lixue Gong, Lubin Fan, Peiran Ren, Qixing Huang, Hujun Bao, and Weiwei Xu. A late fusion cnn for digital matting. In *CVPR*, 2019. 1

[36] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 1

[37] Hao Zhou, Sunil Hadap, Kalyan Sunkavalli, and David W. Jacobs. Deep single-image portrait relighting. In *ICCV*, 2019. 1

[38] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrel, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NIPS*, 2017. 1