Instance-aware Image Colorization

Jheng-Wei Su[1]   Hung-Kuo Chu[1]   Jia-Bin Huang[2]

[1]National Tsing Hua University   [2]Virginia Tech

https://ericsujw.github.io/InstColorization

Figure 1. **Instance-aware colorization.** We present an *instance-aware* colorization method that is capable of producing natural and colorful results on a wide range of scenes containing multiple objects with diverse context (*e.g.*, vehicles, people, and man-made objects).
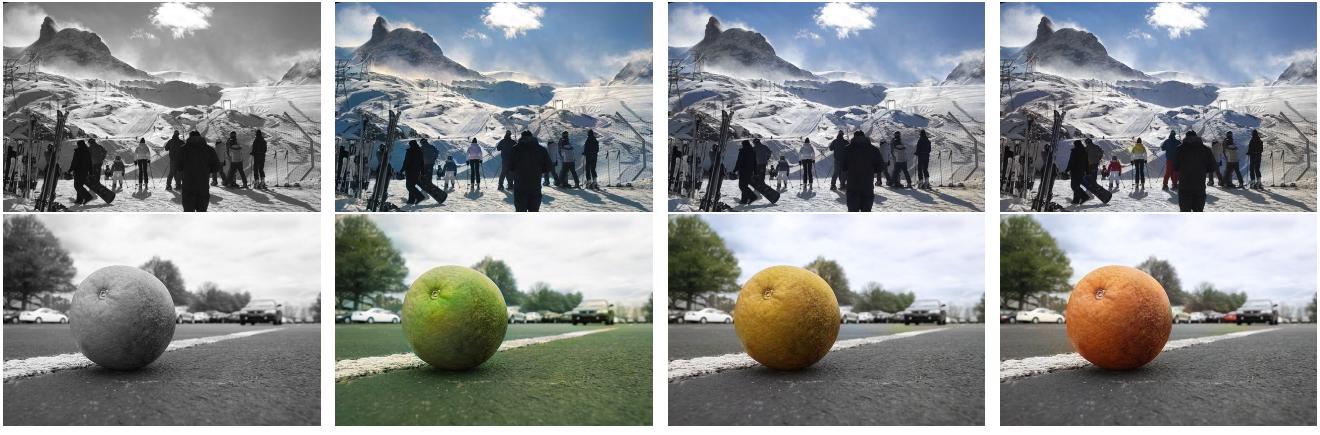
Abstract

Image colorization is inherently an ill-posed problem with multi-modal uncertainty. Previous methods leverage the deep neural network to map input grayscale images to plausible color outputs directly. Although these learning-based methods have shown impressive performance, they usually fail on the input images that contain multiple objects. The leading cause is that existing models perform learning and colorization on the entire image. In the absence of a clear figure-ground separation, these models cannot effectively locate and learn meaningful object-level semantics. In this paper, we propose a method for achieving instance-aware colorization. Our network architecture leverages an off-the-shelf object detector to obtain cropped object images and uses an instance colorization network to extract object-level features. We use a similar network to extract the full-image features and apply a fusion module to full object-level and image-level features to predict the final colors. Both colorization networks and fusion modules are learned from a large-scale dataset. Experimental results show that our work outperforms existing methods on different quality metrics and achieves state-of-the-art performance on image colorization.

1. Introduction

Automatically converting a grayscale image to a plausible color image is an exciting research topic in computer vision and graphics, which has several practical applications such as legacy photos/video restoration or image compression. However, predicting two missing channels from a given single-channel grayscale image is inherently an ill-posed problem. Moreover, the colorization task could be multi-modal [3] as there are multiple plausible choices to colorize an object (*e.g.*, a vehicle can be white, black, red, *etc.*). Therefore, image colorization remains a challenging yet intriguing research problem awaiting exploration.

Traditional colorization methods rely on user intervention to provide some guidance such as color scribbles [20, 12, 35, 26, 22, 31] or reference images [34, 14, 3, 9, 21, 5] to obtain satisfactory results. With the advances of deep learning, an increasing amount of efforts has focused on leveraging deep neural network and large-scale dataset such as ImageNet [28] or COCO-Stuff [2] to learn colorization in an end-to-end fashion [4, 13, 17, 38, 41, 15, 42, 11, 8, 27, 6, 24, 1]. A variety of network architectures have been proposed to address image-level semantics [13, 17, 38, 42] at training or predict per-pixel color distributions to model multi-modality [17, 38, 42]. Although these learning-based

1

| (a) Input | (b) Deoldify [1] | (c) Zhang *et al*. [41] | (d) Ours |

Figure 2. **Limitations of existing methods.** Existing learning-based methods fail to predict plausible colors for multiple object instances such as skiers (top) and vehicles (bottom). The result of Deoldify [1](bottom) also suffers the context confusion (biasing to green color) due to the lack of clear figure-ground separation.

methods have shown remarkable results on a wide variety of images, we observe that existing colorization models do not perform well on the images with multiple objects in a cluttered background (see Figure 2).

In this paper, we address the above issues and propose a novel deep learning framework to achieve *instance-aware* colorization. Our key insight is that a clear figure-ground separation can dramatically improve colorization performance. Performing colorization at the instance level is effective due to the following two reasons. First, unlike existing methods that learn to colorize the entire image, learning to colorize instances is a substantially easier task because it does not need to handle complex background clutter. Second, using localized objects (*e.g.*, from an object detector) as inputs allows the instance colorization network to learn object-level representations for accurate colorization and avoiding color confusion with the background. Specifically, our network architecture consists of three parts: (i) an off-the-shelf pre-trained model to detect object instances and produce cropped object images; (ii) two backbone networks trained end-to-end for instance and full-image colorization, respectively; and (iii) a fusion module to selectively blend features extracted from layers of the two colorization networks. We adopt a three-step training that first trains instance network and full-image network separately, followed by training the fusion module with two backbones locked.

We validate our model on three public datasets (ImageNet [28], COCO-Stuff [2], and Places205 [43]) using the network derived from Zhang *et al*. [41] as the backbones. Experimental results show that our work outperforms existing colorization methods in terms of quality metrics across all datasets. Figure 1 shows sample colorization results generated by our method.

Our contributions are as follows:

- A new learning-based method for fully automatic instance-aware image colorization.
- A novel network architecture that leverages off-the-shelf models to detect the object and learn from large-scale data to extract image features at the instance and full-image level, and to optimize the feature fusion to obtain the smooth colorization results.
- A comprehensive evaluation of our method on comparing with baselines and achieving state-of-the-art performance.

2. Related Work

Scribble-based colorization. Due to the multi-modal nature of image colorization problem, early attempts rely on additional high-level user scribbles (*e.g.*, color points or strokes) to guide the colorization process [20, 12, 35, 26, 22, 31]. These methods, in general, formulate the colorization as a constrained optimization problem that propagates user-specified color scribbles based on some low-level similarity metrics. For instance, Levin *et al*. [20] encourage assigning a similar color to adjacent pixels with similar luminance. Several follow-up approaches reduce color bleeding via edge detection [12] or improve the efficiency of color propagation with texture similarity [26, 22] or intrinsic distance [35] . These methods can generate convincing results with detailed and careful guidance hints provided by the user. The process, however, is labor-intensive. Zhang *et al*. [41] partially alleviate the manual efforts by combining the color hints with a deep neural network.

Example-based colorization. To reduce intensive user efforts, several works colorize the input grayscale im-

age with the color statistics transferred from a reference image specified by the user or searched from the Internet [34, 14, 3, 9, 21, 5, 11]. These methods compute the correspondences between the reference and input image based on some low-level similarity metrics measured at pixel level [34, 21], semantic segments level [14, 3], or super-pixel level [9, 5]. The performance of these methods is highly dependent on how similar the reference image is to the input grayscale image. However, finding a suitable reference image is a non-trivial task even with the aid of automatic retrieval system [5]. Consequently, such methods still rely on manual annotations of image regions [14, 5]. To address these issues, recent advances include learning the mapping and colorization from large-scale dataset [11] and the extension to video colorization [36].

Learning-based colorization Exploiting machine learning to automate the colorization process has received increasing attention in recent years [7, 4, 13, 17, 38, 41, 15, 42]. Among existing works, the deep convolutional neural network has become the mainstream approach to learn color prediction from a large-scale dataset (*e.g.*, ImageNet [28]). Various network architectures have been proposed to address two key elements for convincing colorization: semantics and multi-modality [3].

To model semantics, Iizuka *et al.* [13] and Zhao *et al.* [42] present a two-branch architecture that jointly learns and fuses local image features and global priors (*e.g.*, semantic labels). Zhang *et al.* [38] employ a cross-channel encoding scheme to provide semantic interpretability, which is also achieved by Larsson *et al.* [17] that pre-trained their network for a classification task. To handle multi-modality, some works proposed to predict per-pixel color distributions [17, 38, 42] instead of a single color. These works have achieved impressive performance on images with moderate complexity but still suffer visual artifacts when processing complex images with multiple foreground objects as shown in Figure 2.

Our observation is that learning semantics at either image-level [13, 38, 17] or pixel-level [42] cannot sufficiently model the appearance variations of objects. Our work thus learns object-level semantics by training on the cropped object images and then fusing the learned object-level and full-image features to improve the performance of any off-the-shelf colorization networks.
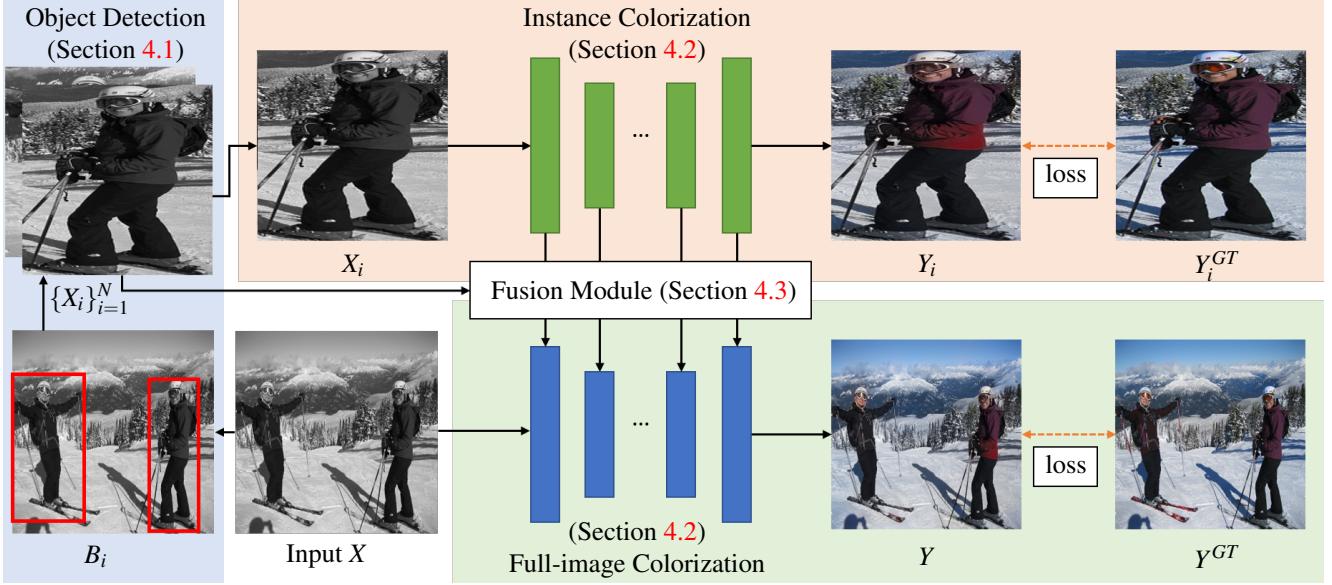
Colorization for visual representation learning. Colorization has been used as a proxy task for learning visual representation [17, 38, 18, 39] and visual tracking [32]. The learned representation through colorization has been shown to transfer well to other downstream visual recognition tasks such as image classification, object detection, and segmentation. Our work is inspired by this line of re-

search on self-supervised representation learning. Instead of aiming to learn a representation that generalizes well to object detection/segmentation, we focus on leveraging the off-the-shelf pre-trained object detector to improve image colorization.

Instance-aware image synthesis and manipulation. Instance-aware processing provides a clear figure-ground separation and facilitates synthesizing and manipulating visual appearance. Such approaches have been successfully applied to image generation [30], image-to-image translation [23, 29, 25], and semantic image synthesis [33]. Our work leverages a similar high-level idea with these methods but differs in the following three aspects. First, unlike DA-GAN [23] and FineGAN [30] that focus only on one *single* instance, our method is capable of handling complex scenes with *multiple* instances via the proposed feature fusion module. Second, in contrast to *Insta*GAN [25] that processes *non-overlapping* instances *sequentially*, our method considers all *potentially overlapping* instances *simultaneously* and produces spatially coherent colorization. Third, compared with Pix2PixHD [33] that uses instance boundary for improving synthesis quality, our work uses *learned* weight maps for blending features from multiple instances.

3. Overview

Our system takes a grayscale image $X \in \mathbb{R}^{H \times W \times 1}$ as input and predicts its two missing color channels $Y \in \mathbb{R}^{H \times W \times 2}$ in the CIE $L^*a^*b^*$ color space in an end-to-end fashion. Figure 3 illustrates our network architecture. First, we leverage an off-the-shelf pre-trained object detector to obtain multiple object bounding boxes $\{B_i\}_{i=1}^{N}$ from the grayscale image, where N is the number of instances. We then generate a set of instance images $\{X_i\}_{i=1}^{N}$ by resizing the images cropped from the grayscale image using the detected bounding boxes (Section 4.1). Next, we feed each instance image X_i and input grayscale image X to the *instance colorization network* and *full-image colorization network*, respectively. The two networks share the same architecture (but different weights). We denote the extracted feature map of instance image X_i and grayscale image X at the j-th network layer as $f_j^{X_i}$ and f_j^X (Section 4.2). Finally, we employ a *fusion module* that fuses all the instance features $\{f_j^{X_i}\}_{i=1}^{N}$ with the full-image feature f_j^X at each layer. The fused full image feature at j-th layer, denoted as $f_j^{\bar{X}}$, is then fed forward to $j+1$-th layer. This step repeats until the last layer and obtains the predict color image Y (Section 4.3). We adopt a sequential approach that first trains the full-image network, followed by the instance network, and finally trains the feature fusion module by freezing the above two networks.

Figure 3. **Method overview.** Given a grayscale image X as input, our model starts with detecting the object bounding boxes (B_i) using an off-the-shelf object detection model. We then crop out every detected instance X_i via B_i and use instance colorization network to colorize X_i. However, as the instances' colors may not be compatible with respect to the predicted background colors, we propose to fuse all the instances' feature maps in every layer with the extracted full-image feature map using the proposed fusion module. We can thus obtain globally consistent colorization results Y. Our training process sequentially trains our full-image colorization network, and the instance colorization network, and the proposed fusion module.

4. Method

4.1. Object detection

Our method leverages detected object instances for improving image colorization. To this end, we employ an off-the-shelf pre-trained network, Mask R-CNN [10], as our object detector. After detecting each object's bounding box B_i, we crop out corresponding grayscale instance image X_i and color instance image Y_i^{GT} from X and Y^{GT}, and resize the cropped images to a resolution of 256×256.

4.2. Image colorization backbone

As shown in Figure 3, our network architecture contains two branches of colorization networks, one for colorizing the instance images and the other for colorizing the full image. We choose the architectures of these two networks so that they have the same number of layers to facilitate feature fusion (discussed in the next section). In this work, we adopt the main colorization network introduced in Zhang *et al.* [41] as our backbones. Although these two colorization networks alone could predict the color instance images Y_i and full image Y, we found that a naïve blending of these results yield visible visual artifacts due to the inconsistency of the overlapping pixels. In the following section, we elaborate on how to fuse the intermediate feature maps from both instance and full-image networks to produce accurate and coherent colorization.

4.3. Fusion module

Here, we discuss how to fuse the full-image feature with multiple instance features to achieve better colorization. Figure 4 shows the architecture of our fusion module. Since the fusion takes place at multiple layers of the colorization networks, for the sake of simplicity, we only present the *fusion module* at the j-th layer. Apply the module to all the other layers is straightforward.

The fusion module takes inputs: (1) a full-image feature f_j^X; (2) a bunch of instance features and corresponding object bounding boxes $\{f_j^{X_i}, B_i\}_{i=1}^N$. For both kinds of features, we devise a small neural network with three convolutional layers to predict full-image weight map W_F and per-instance weight map W_I^i. To fuse per-instance feature $f_j^{X_i}$ to the full-image feature f_j^X, we utilize the input bounding box B_i, which defines the size and location of the instance. Specifically, we resize the instance feature $f_j^{X_i}$ as well as the weight map W_I^i to match the size of full-image and do zero padding on both of them. We denote resized the instance feature and weight map as $\bar{f}_j^{X_i}$ and $\bar{W}_I^i$. After that, we stack all the weight maps, apply softmax on each pixel, and obtain the fused feature using a weighted sum as follows:

$$f_j^{\tilde{X}} = f_j^X \circ W_F + \sum_{i=1}^N \bar{f}_j^{X_i} \circ \bar{W}_I^i, \qquad (1)$$
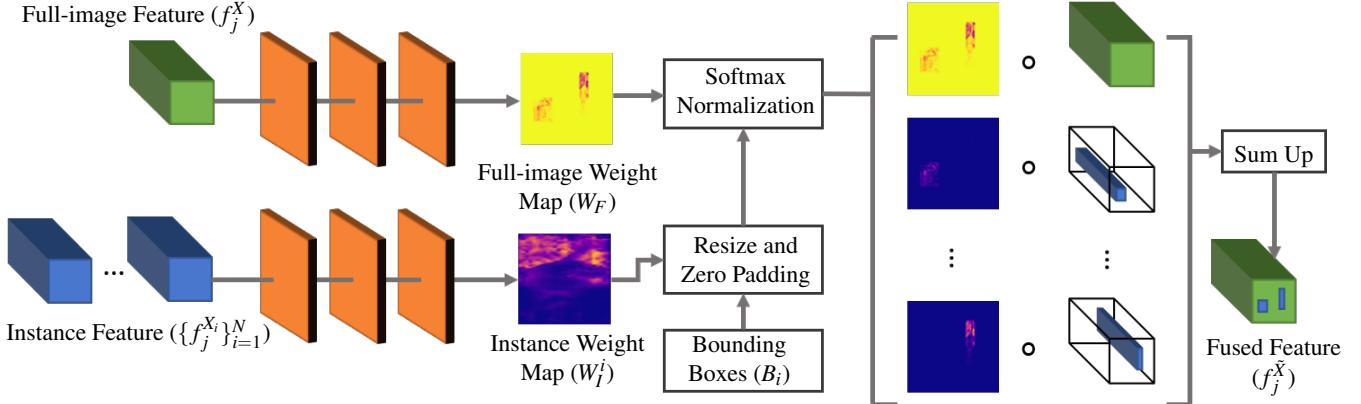
Figure 4. **Feature fusion module.** Given the full-image feature f_j^X and a bunch of instance features $\{f_j^{X_i}\}_{i=1}^N$ from the j-th layer of the colorization network, we first predict the corresponding weight map W_F and W_I^i through a small neural network with three convolutional layers. Both instance feature and weight map are resized, padded with zero to match the original size and local in the full image. The final fused feature $f_j^{\tilde{X}}$ is thus computed using the weighted sum of retargeted features (see Equation 1).

where N is the number of instances.

4.4. Loss Function and Training

Following Zhang *et al.* [41], we adopt the smooth-ℓ_1 loss with $\delta = 1$ as follows:

$$\ell_\delta(x,y) = \tfrac{1}{2}(x-y)^2 \mathbb{1}_{\{|x-y|<\delta\}} + \delta(|x-y| - \tfrac{1}{2}\delta)\mathbb{1}_{\{|x-y|\geqslant\delta\}} \quad (2)$$

We train the whole network sequentially as follows. First, we train the full-image colorization and transfer the learned weights to initialize the instance colorization network. We then train the instance colorization network. Lastly, we freeze the weights in both the full-image model and instance model and move on training the fusion module.

5. Experiments

In this section, we present extensive experimental results to validate the proposed instance-aware colorization algorithm. We start by describing the datasets used in our experiments, performance evaluation metrics, and implementation details (Section 5.1). We then report the quantitative evaluation of three large-scale datasets and compare our results with the state-of-the-art colorization methods (Section 5.2). We show sample colorization results on several challenging images (Section 5.3). We carry out three ablation studies to validate our design choices (Section 5.4). Beyond standard performance benchmarking, we demonstrate the application of colorizing legacy black and white photographs (Section 5.6). We conclude the section with examples where our method fails (Section 5.7). Please refer to the project webpage for the dataset, source code, and additional visual comparison.

5.1. Experimental setting

Datasets. We use three datasets for training and evaluation.

ImageNet [28]: ImageNet dataset has been used by many existing colorization methods as a benchmark for performance evaluation. We use the original training split (1.3 million images) for training all the models and use the testing split (ctest10k) provided by [17] with 10,000 images for evaluation.

COCO-Stuff [2]: In contrast to the *object-centric* images in the ImageNet dataset, the COCO-Stuff dataset contains a wide variety of natural scenes with multiple objects present in the image. There are 118K images (each image is associated with a bounding box, instance segmentation, and semantic segmentation annotations). We use the 5,000 images in the original validation set for evaluation.

Places205 [43]: To investigate how well a colorization method performs on images from a different dataset, we use the 20,500 testing images (from 205 categories) from the Places205 for evaluation. Note that we use the Place205 dataset only for evaluating the transferability. We do not use its training set and the scene category labels for training.

Evaluation metrics. Following the experimental protocol by existing colorization methods, we report the PSNR and SSIM to quantify the colorization quality. To compute the SSIM on color images, we average the SSIM values computed from individual channels. We further use the recently proposed perceptual metric LPIPS by Zhang *et al.* [40] (version 0.1; with VGG backbone).

Training details. We adopt a three-step training process on the ImageNet dataset as follows.

(1) *Full-image colorization network*: We initialize the network with the pre-trained weight provided by [41]. We train the network for two epochs with a learning rate of 1e-5. (2) *Instance colorization network*: We start with the

Table 1. **Quantitative comparison at the full-image level**. The methods in the first block are trained using the ImageNet dataset. The symbol ∗ denotes the methods that are finetuned on the COCO-Stuff training set.

Method	Imagenet ctest10k			COCOStuff validation split			Places205 validation split		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
lizuka *et al*. [13]	0.200	23.636	0.917	0.185	23.863	0.922	0.146	25.581	0.950
Larsson *et al*. [17]	0.188	25.107	0.927	0.183	25.061	0.930	0.161	25.722	0.951
Zhang *et al*. [38]	0.238	21.791	0.892	0.234	21.838	0.895	0.205	22.581	0.921
Zhang *et al*. [41]	0.145	26.166	0.932	0.138	26.823	0.937	0.149	25.823	0.948
Deoldify *et al*. [1]	0.187	23.537	0.914	0.180	23.692	0.920	0.161	23.983	0.939
Lei *et al*. [19]	0.202	24.522	0.917	0.191	24.588	0.922	0.175	25.072	0.942
Ours	**0.134**	**26.980**	**0.933**	**0.125**	**27.777**	**0.940**	**0.130**	**27.167**	**0.954**
Zhang *et al*. [41]*	0.140	26.482	0.932	0.128	27.251	0.938	0.153	25.720	0.947
Ours*	**0.125**	**27.562**	**0.937**	**0.110**	**28.592**	**0.944**	**0.120**	**27.800**	**0.957**

pre-trained weight from the trained full-image colorization network above and finetune the model for five epochs with a learning rate of 5e-5 on the extracted instances from the dataset. (3) *Fusion module*: Once both the full-image and instance network have been trained (*i.e*., warmed-up), we integrate them with the proposed fusion module. We finetune all the trainable parameters for 2 epochs with a learning rate of 2e-5. In our implementation, the numbers of channels of full-image feature, instance feature and fused feature in all 13 layers are 64, 128, 256, 512, 512, 512, 512, 256, 256, 128, 128, 128 and 128.

In all the training processes, we use the ADAM optimizer [16] with $\beta_1 = 0.99$ and $\beta_2 = 0.999$. For training, we resize all the images to a resolution of 256×256. Training the model on the ImageNet takes about three days on a desktop machine with one single RTX 2080Ti GPU.

5.2. Quantitative comparisons
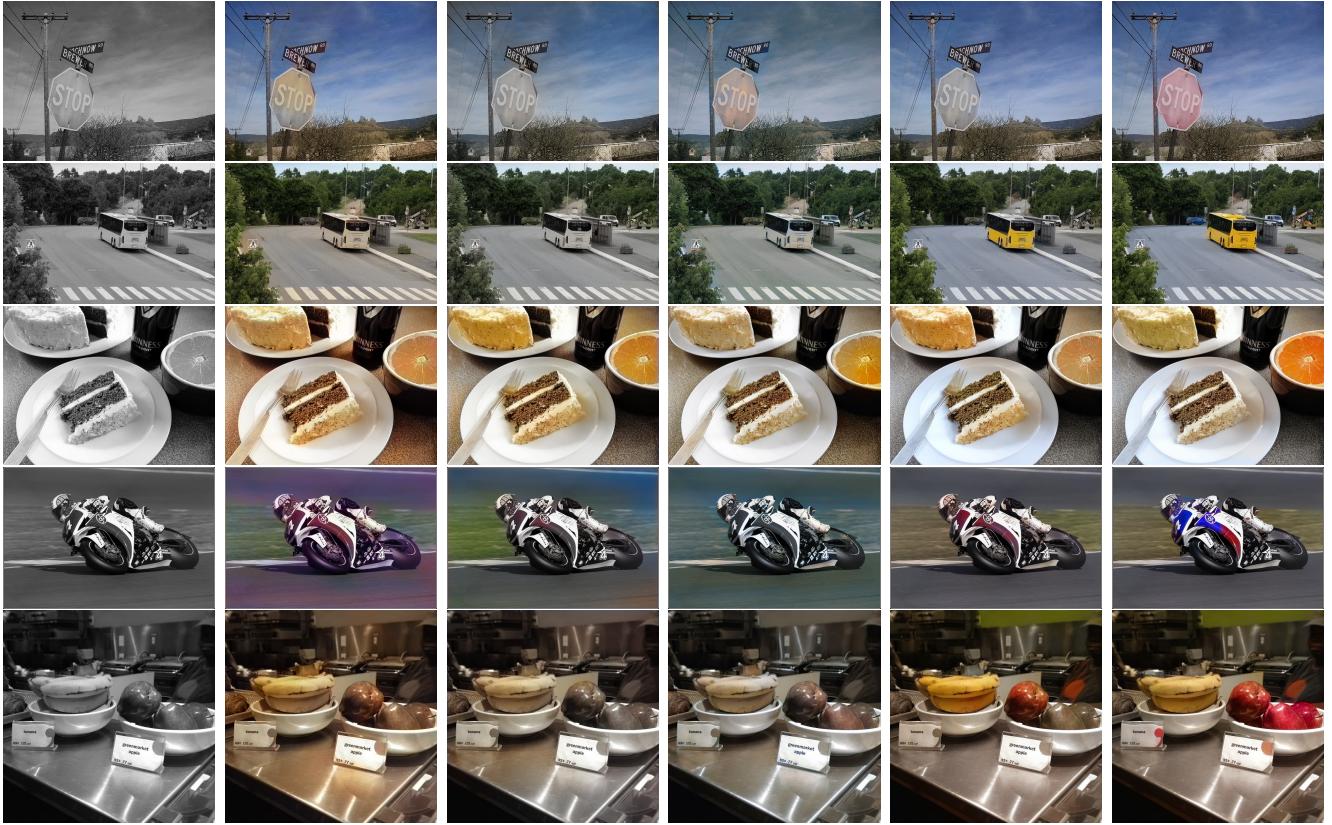
Comparisons with the state-of-the-arts. We report the quantitative comparisons on three datasets in Table 1. The first block of the results shows models trained on the ImageNet dataset. Our instance-aware model performs favorably against several recent methods [13, 18, 38, 41, 1, 19] on all three datasets, highlighting the effectiveness of our approach. Note that we adopted the automatic version of Zhang *et al*. [41] (*i.e*., without using any color guidance) in all the experiments. In the second block, we show the results using our model finetuned on the COCO-Stuff training set (denoted by the "*"). As the COCO-Stuff dataset contains more diverse and challenging scenes, our results show that finetuning on the COCO-Stuff dataset further improves the performance on the *other two datasets* as well. To highlight the effectiveness of the proposed instance-aware colorization module, we also report the results of Zhang *et al*. [41] finetuned on the same dataset as a strong baseline for a fair comparison. For evaluating the performance at the *instance-level*, we take the full-image ground-truth/prediction and crop the instances us-

Table 2. **Quantitative comparison at the instance level.** The methods in the first block are trained using the ImageNet dataset. The symbol ∗ denotes the methods that are finetuned on the COCO-Stuff training set.

Method	COCOStuff validation split		
	LPIPS ↓	PSNR ↑	SSIM ↑
lizuka *et al*. [13]	0.192	23.444	0.900
Larsson *et al*. [17]	0.179	25.249	0.914
Zhang *et al*. [38]	0.219	22.213	0.877
Zhang *et al*. [41]	0.154	26.447	0.918
Deoldify *et al*. [1]	0.174	23.923	0.904
Lei *et al*. [19]	0.177	24.914	0.908
Ours	**0.115**	**28.339**	**0.929**
Zhang *et al*. [41]*	0.149	26.675	0.919
Ours*	**0.095**	**29.522**	**0.938**

ing the ground-truth bounding boxes to form instance-level ground-truth/predictions. Table 2 summarizes the performance computed by averaging over all the instances on the COCO-Stuff dataset. The results present a significant performance boost gained by our method in all metrics, which further highlights the contribution of instance-aware colorization to the improved performance.

User study. We conducted a user study to quantify the user-preference on the colorization results generated by our method and another two strong baselines, Zhang *et al*. [37] (finetuned on the COCO-Stuff dataset) and a popular online colorization method DeOldify [1]. We randomly select 100 images from the *COCO-Stuff* validation dataset. For each participant, we show him/her a pair of colorized results and ask for the preference (forced-choice comparison). In total, we have 24 participants casting 2400 votes in total. The results show that on average our method is preferred when compared with Zhang *et al*. [37] (61% v.s. 39%) and DeOldify [1] (72% v.s. 28%). Interestingly, while DeOldify does not produce accurate colorization evaluated in the benchmark experiment, the saturated colorized results are

(a) Input (b) Iizuka *et al*. [13] (c) Larrson *et al*. [18] (d) Deoldify [1] (e) Zhang *et al*. [41] (f) Ours

Figure 5. **Visual Comparisons with the state-of-the-arts.** Our method predicts visually pleasing colors from complex scenes with multiple object instances.

sometimes more preferred by the users.
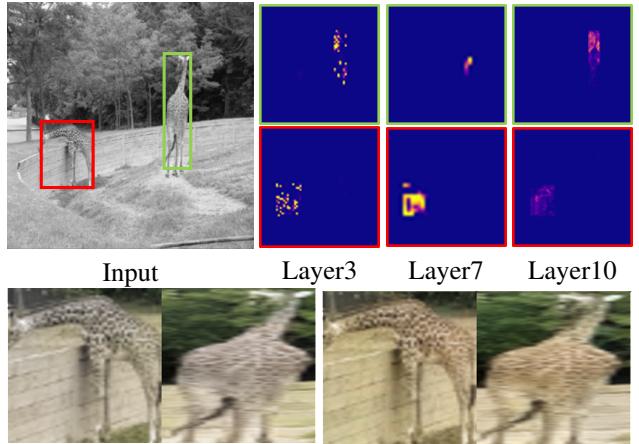
5.3. Visual results

Comparisons with the state-of-the-art. Figure 5 shows sample comparisons with other competing baseline methods on *COCO-Stuff*. In general, we observe a consistent improvement in visual quality, particularly for scenes with multiple instances.

Visualizing the fusion network. Figure 6 visualizes the learned masks for fusing instance-level and full-image level features at multiple levels. We show that the proposed instance-aware processing leads to improved visual quality for complex scenarios.

5.4. Ablation study

Here, we conduct ablation study to validate several important design choices in our model in Table 3. In all ablation study experiments, we use the COCO-Stuff validation dataset. First, we show that fusing features extracted from the instance network with the full-image network improve the performance. Fusing features for both encoder and decoder perform the best. Second, we explore different strate-

Input Layer3 Layer7 Layer10

Zhang *et al*. [41] Our results

Figure 6. **Visualizing the fusion network.** The visualized weighted mask in layer3, layer7 and layer10 show that our model learns to *adaptively* blend the features across different layers. Fusing instance-level features help improve colorization.

gies of selecting object bounding boxes as inputs for our instance network. The results indicate that our default set-

Table 3. **Ablations.** We validate our design choices by comparing with several alternative options.

(a) Different Fusion Part

Fusion Part		COCOStuff validation split		
Encoder	Decoder	LPIPS ↓	PSNR ↑	SSIM ↑
×	×	0.128	27.251	0.938
✓	×	0.120	28.146	0.942
×	✓	0.117	27.959	0.941
✓	✓	**0.110**	**28.592**	**0.944**

(b) Different Bounding Box Selection

Box Selection	COCOStuff validation split		
	LPIPS ↓	PSNR ↑	SSIM ↑
Select top 8	**0.110**	**28.592**	**0.944**
Random select 8	0.113	28.386	0.943
Select by threshold	0.117	28.139	0.942
G.T. bounding box	0.111	28.470	**0.944**

(c) Different Weighted Sum

Weighted Sum	COCOStuff validation split		
	LPIPS ↓	PSNR ↑	SSIM ↑
Box mask	0.140	26.456	0.932
G.T. mask	0.199	24.243	0.921
Fusion module	**0.110**	**28.592**	**0.944**

ting of choosing the top eight bounding boxes in terms of confidence score returned by object detector performs best and is slightly better than using the ground-truth bounding box. Third, we experiment with two alternative approaches (using the detected box as a mask or using the ground-truth instance mask provided in the COCO-Stuff dataset) for fusing features from multiple potentially overlapping object instances and the features from the full-image network. Using our fusion module obtains a notable performance boost than the other two options. This shows the capability of our fusion module to tackle more challenging scenarios with multiple overlapping objects.

5.5. Runtime analysis

Our colorization network involves two steps: (1) colorizing the individual instances and outputting the instance features; and (2) fusing the instance features into the full-image feature and producing a full-image colorization. Using a machine with Intel i9-7900X 3.30GHz CPU, 32GB memory, and NVIDIA RTX 2080ti GPU, our average inference time over all the experiments is 0.187s for an image of resolution 256×256. Each of two steps takes approximately 50% of the running time, while the complexity of step 1 is proportional to the number of input instances and ranges from 0.013s (one instance) to 0.1s (eight instances).

(a) Missing detections   (b) Superimposed detections

Figure 8. **Failure cases.** (*Left*) our model reverts back to the full-image colorization when a lot of vases are missing in the detection. (*Right*) the fusion module may get confused when there are many superimposed object bounding boxes.

5.6. Colorizing legacy black and white photos

We apply our colorization model to colorize legacy black and white photographs. Figure 7 shows sample results along with manual colorization results by human expert[1].

5.7. Failure modes

We show 2 examples of failure cases in Figure 8. When the instances were not detected, our model reverts back to the full-image colorization network. As a result, our method may produce visible artifacts such as washed-out colors or bleeding across object boundaries.

6. Conclusions

We present a novel instance-aware image colorization. By leveraging an off-the-shelf object detection model to crop out the images, our architecture extracts the feature from our instance branch and full images branch, then we fuse them with our newly proposed fusion module and obtain a better feature map to predict the better results. Through extensive experiments, we show that our work compares favorably against existing methods on three benchmark datasets.

Input   Expert   Our results

Figure 7. **Colorizing legacy photographs.** The middle column shows the manually colorized results by the experts.

[1] bit.ly/color_history_photos

References

[1] Jason Antic. jantic/deoldify: A deep learning based project for colorizing and restoring old images (and video!). https://github.com/jantic/DeOldify, 2019. Online; accessed: 2019-10-16. 1, 2, 6, 7

[2] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *CVPR*, 2018. 1, 2, 5

[3] Guillaume Charpiat, Matthias Hofmann, and Bernhard Schölkopf. Automatic image colorization via multimodal predictions. In *ECCV*, 2008. 1, 3

[4] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *ICCV*, 2015. 1, 3

[5] Alex Yong-Sang Chia, Shaojie Zhuo, Raj Kumar Gupta, Yu-Wing Tai, Siu-Yeung Cho, Ping Tan, and Stephen Lin. Semantic colorization with internet images. *ACM TOG (Proc. SIGGRAPH Asia)*, 30(6):156:1–156:8, 2011. 1, 3

[6] Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, Min Jin Chong, and David Forsyth. Learning diverse image colorization. In *CVPR*, 2017. 1

[7] Aditya Deshpande, Jason Rock, and David Forsyth. Learning large-scale automatic image colorization. In *ICCV*, 2015. 3

[8] Sergio Guadarrama, Ryan Dahl, David Bieber, Mohammad Norouzi, Jonathon Shlens, and Kevin Murphy. Pixcolor: Pixel recursive colorization. In *BMVC*, 2017. 1

[9] Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin Ng, and Huang Zhiyong. Image colorization using similar images. In *MM*, 2012. 1, 3

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. In *ICCV*, 2017. 4

[11] Mingming He, Dongdong Chen, Jing Liao, Pedro V. Sander, and Lu Yuan. Deep exemplar-based colorization. *ACM TOG (Proc. SIGGRAPH)*, 37(4):47:1–47:16, 2018. 1, 3

[12] Yi-Chin Huang, Yi-Shin Tung, Jun-Cheng Chen, Sung-Wen Wang, and Ja-Ling Wu. An adaptive edge detection based colorization algorithm and its applications. In *ACM MM*, 2005. 1, 2

[13] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM TOG (Proc. SIGGRAPH)*, 35(4):110:1–110:11, 2016. 1, 3, 6, 7

[14] Revital Irony, Daniel Cohen-Or, and Dani Lischinski. Colorization by example. In *EGSR*, 2005. 1, 3

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 1, 3

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015. 6

[17] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, 2016. 1, 3, 5, 6

[18] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017. 3, 6, 7

[19] Chenyang Lei and Qifeng Chen. Fully automatic video colorization with self-regularization and diversity. In *CVPR*, 2019. 6

[20] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM TOG (Proc. SIGGRAPH)*, 23(3):689–694, 2004. 1, 2

[21] Xiaopei Liu, Liang Wan, Yingge Qu, Tien-Tsin Wong, Stephen Lin, Chi-Sing Leung, and Pheng-Ann Heng. Intrinsic colorization. *ACM TOG (Proc. SIGGRAPH Asia)*, 27(5):152:1–152:9, 2008. 1, 3

[22] Qing Luan, Fang Wen, Daniel Cohen-Or, Lin Liang, Ying-Qing Xu, and Heung-Yeung Shum. Natural image colorization. In *EGSR*, 2007. 1, 2

[23] Shuang Ma, Jianlong Fu, Chang Wen Chen, and Tao Mei. Da-gan: Instance-level image translation by deep attention generative adversarial networks. In *CVPR*, 2018. 3

[24] Safa Messaoud, David Forsyth, and Alexander G. Schwing. Structural consistency and controllability for diverse colorization. In *ECCV*, 2018. 1

[25] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Instagan: Instance-aware image-to-image translation. 2019. 3

[26] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM TOG (Proc. SIGGRAPH)*, 25(3):1214–1220, 2006. 1, 2

[27] Amelie Royer, Alexander Kolesnikov, and Christoph H. Lampert. Probabilistic image colorization. In *BMVC*, 2017. 1

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. 1, 2, 3, 5

[29] Zhiqiang Shen, Mingyang Huang, Jianping Shi, Xiangyang Xue, and Thomas Huang. Towards instance-level image-to-image translation. In *CVPR*, 2019. 3

[30] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *CVPR*, 2019. 3

[31] Daniel Sýkora, John Dingliana, and Steven Collins. Lazybrush: Flexible painting tool for hand-drawn cartoons. In *CGH*, 2009. 1, 2

[32] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *ECCV*, 2018. 3

[33] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018. 3

[34] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. Transferring color to greyscale images. *ACM TOG (Proc. SIGGRAPH)*, 21(3):277–280, 2002. 1, 3

[35] L. Yatziv and G. Sapiro. Fast image and video colorization using chrominance blending. *TIP*, 15(5):1120–1129, 2006. 1, 2

[36] Bo Zhang, Mingming He, Jing Liao, Pedro V. Sander, Lu Yuan, Amine Bermak, and Dong Chen. Deep exemplar-based video colorization. In *CVPR*, 2019. 3

[37] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. Two-stage sketch colorization. *ACM TOG (Proc. SIGGRAPH Asia)*, 37(6):261:1–261:14, 2018. 6

[38] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016. 1, 3, 6

[39] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017. 3

[40] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[41] 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. *ACM TOG (Proc. SIGGRAPH)*, 36(4):119:1–119:11, 2017. 1, 2, 3, 4, 5, 6, 7, 11, 12

[42] J. Zhao, L. Liu, , C. G. M. Snoek, J. Han, and L. Shao. Pixel-level semantics guided image colorization. In *BMVC*, 2018. 1, 3

[43] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 2, 5

Instance-aware Image Colorization
Supplemental Material

In this supplementary document, we provide additional visual comparisons and quantitative evaluation to complement the main manuscript.

A. Visualization of Fusion Module

We show two images where multiple instances have been detected by the object detection model. We visualize the weighted mask predicted by our fusion module at multiple layers (3rd, 7th, and 10th layers). Note that our fusion module learns to *adaptively* blend the features extracted from the instance colorization branch to enforce coherent colorization for the entire image.
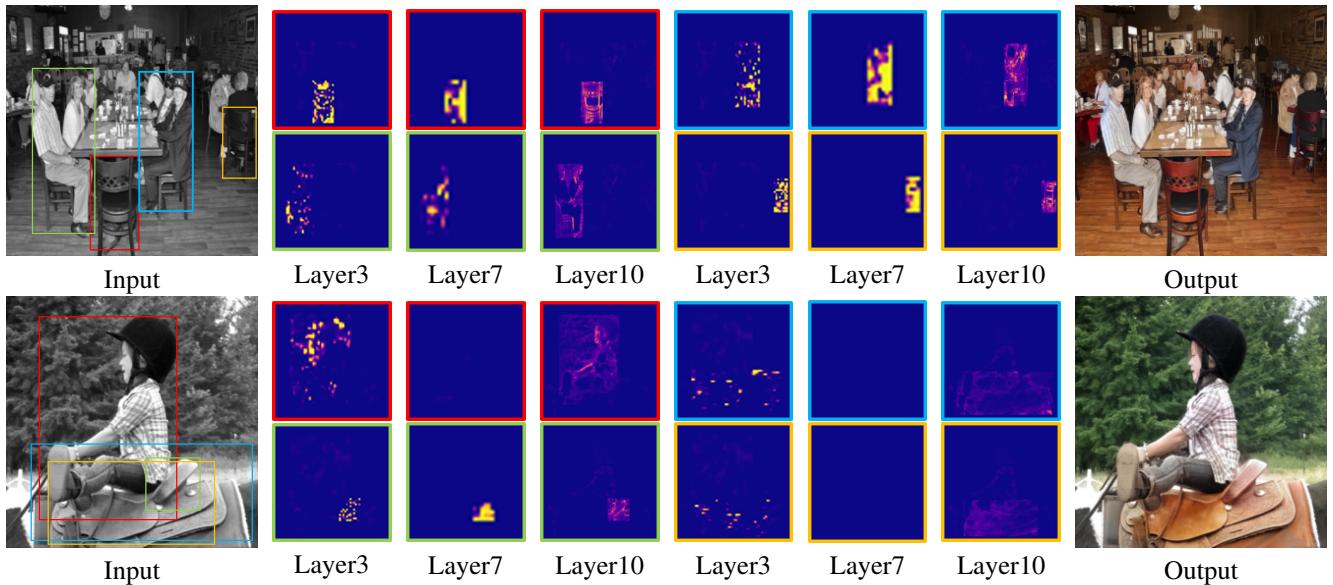
Figure 9. **Visualizing the fusion network.** The visualized weighted mask in layer3, layer7 and layer10.

B. Extended quantitative evaluation

In this section, we provide two additional quantitative evaluation.

Baselines using pre-trained weights. As we mentioned in our paper, we use Zhang et al. [41] as our backbone colorization model. However, the model in [41] is trained for guidance colorization with a resolution of 176×176. In our setting, we need a fully-automatic colorization model with resolution 256×256. In light of this, we retrain the model on the ImageNet dataset to fit our setting. Here we carry out an experiment on the COCOStuff dataset. We show a quantitative comparison and qualitative comparison between (1) the original pre-trained weight provided by the authors and (2) the weight after the fine-tuning step. In general, re-training the model under the resolution of 256×256 results in slight performance degradation under PSNR and similar performance in terms of LPIPS and SSIM with respect to the original pre-trained model.

Table 4. **Quantitative Comparison with different models weight on COCOStuff** We test our colorization backbone model [41] with different weight on COCOStuff.

Method	COCOStuff validation split		
	LPIPS ↓	PSNR ↑	SSIM ↑
(a) Original model weight [41]	0.133	27.050	0.937
(b) Retrain on the ImageNet dataset	0.138	26.823	0.937
(c) Finetuned on the COCOStuff dataset	**0.128**	**27.251**	**0.938**

| (a) (Gray)	(b) Original model weight [41]	(c) Retrain on the ImageNet dataset	(d) Finetuned on the COCOStuff dataset
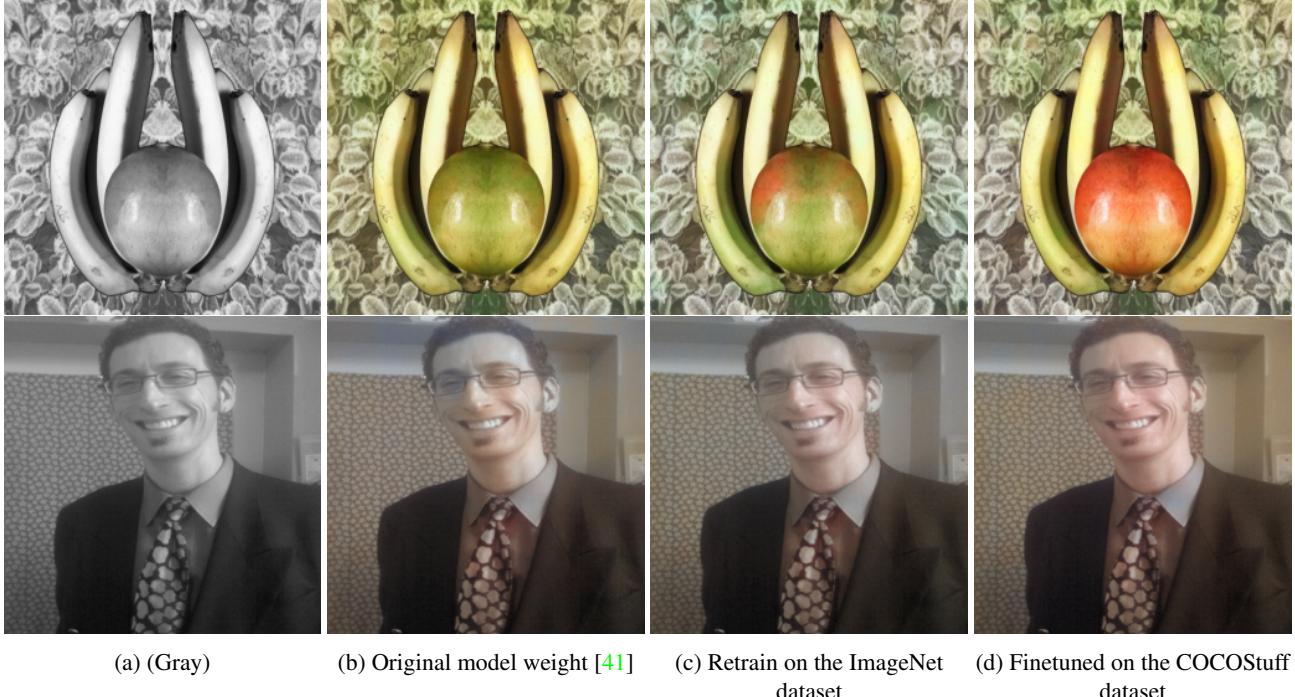
Figure 11. **Different model weight of [41]** We present some images that are inferencing from different weight of [41]. The images above from left to right is Grayscale, Original weight, retrain on Imagenet, and fine-tuned on COCOStuff.

C. User Study setup

Here we describe the detailed procedure of our user study. We first ask the subjects to read a document, including the instruction of the webpage and selection basis (according to the color correctness and naturalness). The subjects then start a pair-wise forced-choice *without* any ground truth reference or grayscale reference (i.e., no-reference tests). We embed two redundant comparisons for sanity check (i.e the same comparisons appear in the user study twice). We reject the votes if the subjects do not answer consistently for both redundant comparisons. We summarize the results in the main paper. We provide all the images used in our user study as well as the user preference votes in the supplementary web-based viewer (click the 'User Study Result' button).

D. Failure cases

While we have shown significantly improved results, single image colorization remains a challenging problem. Figure 13 shows two examples where our model is unable to predict the vibrant, bright colors from the given grayscale input image.

(a) Grayscale input   (b) Ground truth color image   (c) Our result
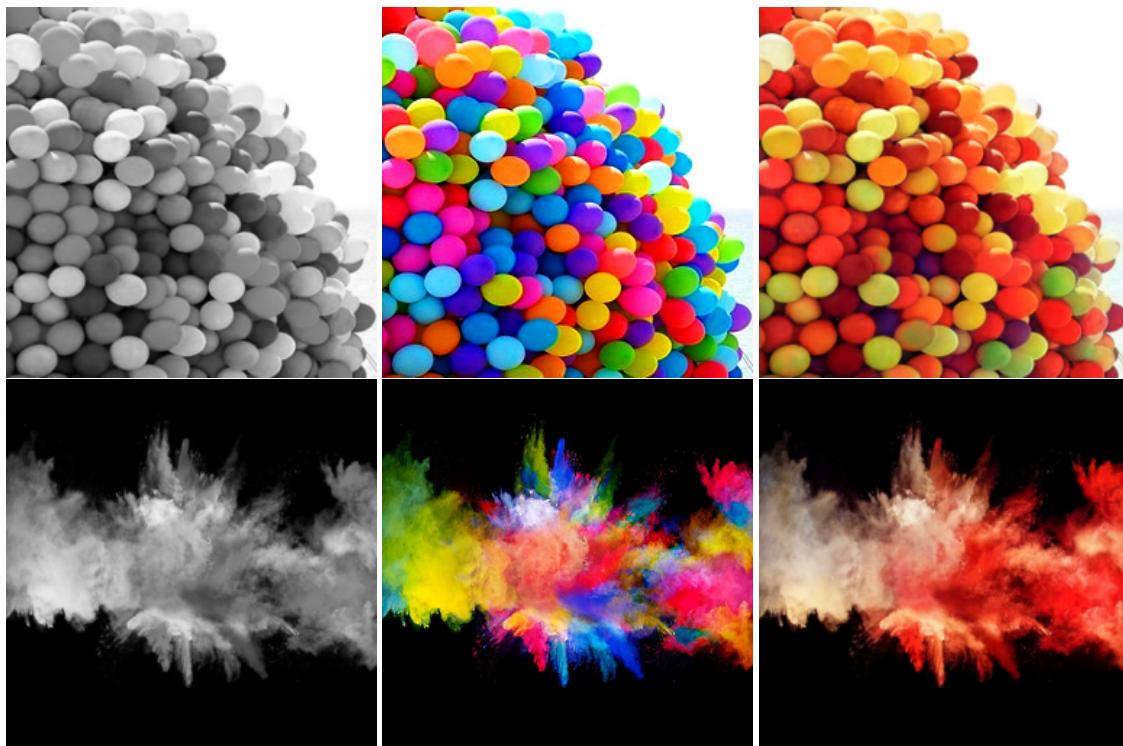
Figure 13. **Failure cases.** We present some images that are out of distribution and are capable of colorizing plausible color. The images above from left to right is (a) grayscale input, (b) ground truth color image and (c) our result.