Automatic Object Recoloring Using Adversarial Learning

Siavash Khodadadeh*
Dept. of Computer Science
University of Central Florida
Orlando, Florida, USA
siavash.khodadadeh@knights.ucf.edu

Saeid Motiian
Adobe Inc.
San Jose, California, USA
motiian@adobe.com

Zhe Lin
Adobe Inc.
San Jose, California, USA
zlin@adobe.com

Ladislau Bölöni
Dept. of Computer Science
University of Central Florida
Orlando, Florida, USA
lboloni@cs.ucf.edu

Shabnam Ghadar
Adobe Inc.
San Jose, California, USA
ghadar@adobe.com

Abstract

We propose a novel method for automatic object recoloring based on Generative Adversarial Networks (GANs). The user can simply give commands of the form `recolor <object> to <color>` *which will be executed without any need of manual edit. Our approach takes advantage of pre-trained object detectors and saliency mask segmentation networks. The segmented mask of the given object along with the target color and the original image form the input to the GAN. The use of cycle consistency loss ensures the realistic look of the results. To our best knowledge, this is the first algorithm where the automatic recoloring is only limited by the ability of the mask extractor to map a natural language tag to a specific object in the image (several hundred object types at the time of this writing). For a performance comparison, we also adapted other state of the art methods to perform this task. We found that our method had consistently yielded qualitatively better recoloring results.*

1. Introduction

Color manipulation is an active field of study that aims to transform the RGB values of an image to convey a novel artistic vision or achieve the goals of the customer. While manipulating colors has a long history in painting and photography, in recent years highly innovative new approaches emerged that combine image editing and color manipulation to enable color transfer [8], style transfer [16], and appearance transfer [13]. For instance, Chang *et al*. [2] use object color distributions to change the color of objects automatically.

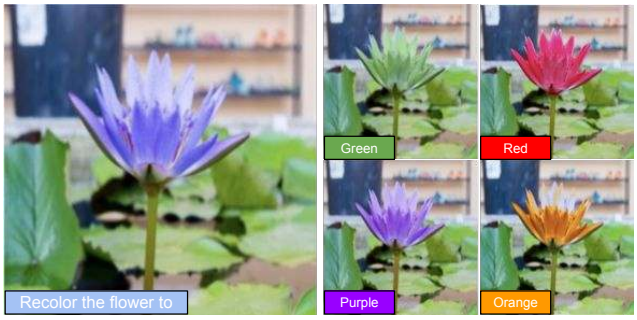

Figure 1. Left is the given image, on the right are the corresponding results for these commands: recolor <flower> to (<green>, <red>, <purple>, <orange>). Image is from **solahuddin - stock ID #100036041**.

In this paper, we describe a method to change the color of a specific object within an image. From the user's point of view, this is done with a single command of the form `recolor <object> to <color>` (see Figure 1 for several examples).

Object recoloring has numerous applications. For instance, photographers often re-touch images for commercial purposes. The re-touching process is usually manual, and in many cases, tedious. Our algorithm can significantly improve the experience and productivity of the photographer. Another use case is searching for images in social curation websites like Pinterest or Juxtapost. Fully auto-

*The paper was written when the student was an intern in Adobe.

matic recoloring can help to improve search and retrieval. To give an example, when a user searches for a "Person with magenta shirt", the system might be able to provide an appropriate, recolored image, even if no such original image exists.

The starting point for this work was adapting state of the art colorization and re-colorization methods to use saliency masks. Even though these methods are very powerful for their original use case, we found that they introduce defects and artifacts in the output when used with masks. Our proposed algorithm, Fully Automatic Color Transformer (FACT) takes the saliency masks into the account during training, allowing us to generate recolored images with significantly less defects and artifacts. The main contributions of this paper can be summarized as follows:

- We introduce a learning-based object recoloring approach that combines conditional GAN based recoloring with an off-the-shelf masking algorithm.
- We propose a method for collecting and automatically annotating a dataset suitable for training the recoloring GAN.
- Through a series of experiments, we show that our approach performs qualitatively better than previous fully automatic approaches.

2. Related work

2.1. Colorization

Colorization of grayscale images is an actively studied area of computer vision. In contrast to the object recoloring task which is the subject of this paper, colorization usually aims to colorize the whole image, rather than a single object pointed out through language.

Early approaches for colorization often relied on additional user input, such as scribbling color points or strokes [22, 25, 20]. For instance, Levin *et al*. [14] propose an approach that is relying on an optimization process where adjacent pixels with similar luminance are assigned similar colors as the ones scribbled by the user.

The deep learning revolution enabled the creation of networks that take a grayscale image and generate a colorized version of it without any additional input [28]. Evaluating the result of this approach using the "colorization Turing test" which asks a human to choose between colored image and a real image had shown that the approach was able to fool the humans test subjects more often than previous work.

A more recent approach by Yoo *et al*. [26] proposes colorization with limited data. The authors use memory networks to get features that best match the color of the query image. This allows the system to recall from the memory network the colors of previously seen similar objects, and thus create more reasonable-looking colorized images.

An important subproblem of this field is coloring a grayscale image with respect to a reference image. These methods compute the correspondences between the reference and input images based on low-level similarity metrics [23, 10, 4]. Recently, Zhang *et al*. [27] demonstrated the colorization of video frames with respect to a style image while considering temporal consistency across frames.

2.2. Recoloring

Most approaches for recoloring images involve a two-step, palette-based model. The first step extracts a palette from the image while the second step finds a mapping based on the target color (or target palette) to be applied to every pixel within the image.

Chang *et al*. [3] propose a method for palette-based recoloring which takes into account constraints that ensure the smoothness of recoloring. Some of these constraints are staying in gamut, pixel continuity, monotonicity in L channel, and one-to-one mapping function. By considering all of these constraints, their method improves the recoloring outcome.

Gong *et al*. [9] introduce a very efficient technique for editing the primary color of consumer product images. First, they cluster the intensity distribution in CIE L*a*b*. They define the primary cluster as the largest cluster in this space, and find a mapping that maintains all other clusters, but maps the primary cluster to the target color. By solving this cluster mapping equation, we get a 3×3 matrix that can be applied to any pixel value in the original image.

By emerging advances in deep learning, many different approaches for coloring based on deep neural networks have been proposed. Zhang *et al*. [29] proposed an interactive approach using deep learning. The user inputs a grayscale image and the RGB value of some of the pixels. The network takes the user color input and the grayscale image and generates a colored image. Although this work aimed for solving the coloring problem, the same pipeline can be used for recoloring. Afifi *et al*. [2] use a deep learning method to obtain semantic segmentation from each image. Based on these segmentation masks, they cluster the possible color of an object. For example, the sky can be blue (daytime), yellow/red (dusk/dawn), and dark (nighttime). They apply palette mapping between these masks to recolor the scene. Cho *et al*. [5] propose a method based on deep learning which takes an image and a palette as input and outputs the target image with the target palette. They use adversarial training and encoder-decoder architecture to achieve this.

2.3. Image-to-image translation

Both coloring and recoloring could be seen as image-to-image translation problems. The input is an image with

a source color or source palette and the output is another image with a target color or target palette. [11] proposes a method which can convert labels to street scenes, day to night, aerial image to map and black and white images to RGB images. Given a dataset of paired images, they use conditional generator and paired discriminator to learn this translation from images of source domain to target domain. [30] solve the image-to-image translation between two domains when there is no dataset of paired images. They leverage adversarial training alongside a novel cycle consistency which allows generating novel images of the target domain without having a dataset of paired images. [6] propose StarGAN. They apply image-to-image translation between multiple domains instead of just two domains when there is no dataset of paired images. They use a shared generator which takes an image and target domain and outputs the image in the target domain. They also encode the target domain as an input to discriminator to train a conditional discriminator. [15] propose open edit which takes as input an image and a sentence and outputs the desired image with applying the effect from text. This can be used for recoloring directly by just giving text commands based on color and input image. [19] also apply image-to-image translation by mapping the local statistic from one domain to the other while preserving the semantic content.

3. Object recoloring

An ideal training dataset to train an object recoloring network using supervised learning would consist of pairs of images, identical except the color of the target object. Collecting such a dataset from real-world images is very difficult - it would require an extraordinarily degree of effort from the photographer even in the case of static objects, and it is essentially impossible to achieve for pictures containing humans. Our approach proposes a technique which can learn recoloring without requiring paired images of the same scene.

Collecting such a dataset is not a trivial task. Instead, we leverage unpaired image to image translation methods which do not require paired images of the same scene.

Our method is inspired by conditional GANs [17] and the technique of cycle consistency loss [30], and allows the transfer of colors by considering each color as a separate domain. We start by building a dataset of unpaired images of different domains as described in Section 3.1.

Next, we train a generator network conditioned on the input image, the object mask, and the target color. The object mask can be generated using off-the-shelf bounding box and mask detection algorithms which take as input the command from the user. These are used for the preprocessing of the input and are not part of the training workflow.

Figure 3 shows the steps of the proposed algorithm. Our early experiments made clear that the manner in which the color and mask information is entered at the various phases of the neural network has critical importance on the speed of the training and the quality of the generated output. These architectural details are discussed in Sections 3.1 through 3.4.

3.1. Dataset creation

To train our algorithm we need a training dataset that combines tuples of images, object tags, colors, and masks of the tagged object in the image. To generate such a dataset, we leveraged the Google Images search engine. The process is shown in Figure 2. First, we collect a list of object tags and colors commonly used in image captions. To account for variations in spelling and overlapping meanings, we performed a minimal preprocessing step (e.g. replacing "reddish" with "red"). By collecting a list of 14 colors and 421 tags we generated $421 \times 14 = 5894$ distinct queries, such as "green apple", "pink backpack" or "orange house". Figure 4 shows the results of several queries from Google Images. We used the Selenium [1] browser automation tool to scroll down the search page and extract the image URLs. After extracting all the image URLs, we download them using 100 threads in parallel. For every downloaded image, we detect objects and top-5 tags for each object. We keep the image if the query tag matches one of the five predicted tags of at least one of the detected objects. If there are multiple objects with the same tag, we pick the one with the highest detection confidence score. Finally, we extract a soft mask for that object with the same tag of the query. As a result, we have a dataset $D = \{x_i, t_i, c_i, m_i\}$ such that x_i, t_i, c_i, m_i are $256 \times 256 \times 3$ image, tag, color, and 256×256 soft mask for item $i \in \{1, \ldots, M\}$.

3.2. Color transformer network

The objective of the color transformer network is to transform the original image into the recolored image. The object to be recolored and the desired color is provided to the network in form of the mask and a color mask respectively.

The color mask M is created as follows (see Figure 5). Given an image $I \in [0, 255]^{H \times W \times 3}$, a soft mask $S \in [0, 1]^{H \times W \times 1}$ and an RGB value of the target color, we first generate a matrix of zeros with 4 channels of the shape $(h, w, 4)$. We set the values of the pixel (i, j) in $M[:, :, [R, G, B]]$ to the RGB value of the target color if $S[i, j] > 0$. In the last channel of M we copy the soft mask, $M[:, :, [last]] = S$.

The architecture we used for the color transformer network, $\mathcal{G}$, is based on the U-Net architecture with concatenating layers from the encoder to the decoder (Figure 6). The network takes an image as input. The internal encoder layers receive an additional input in the form of the soft mask, as at this point we only need to convey the position of the

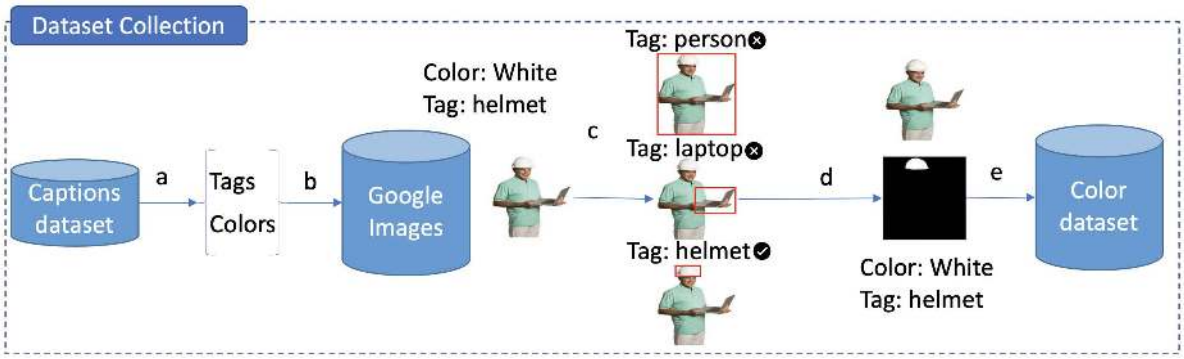

Figure 2. Dataset collection. a. Collecting a list of tags and colors. b. Download images from Google Images using queries based on the tags and colors. c. Extract bounding boxes corresponding to tags. d. Extraction of soft maps from bounding boxes corresponding to the appropriate tags. e. Adding the tuple of image, mask, color and tag to the dataset. Image is from **paulovilela - stock ID #100153754**.

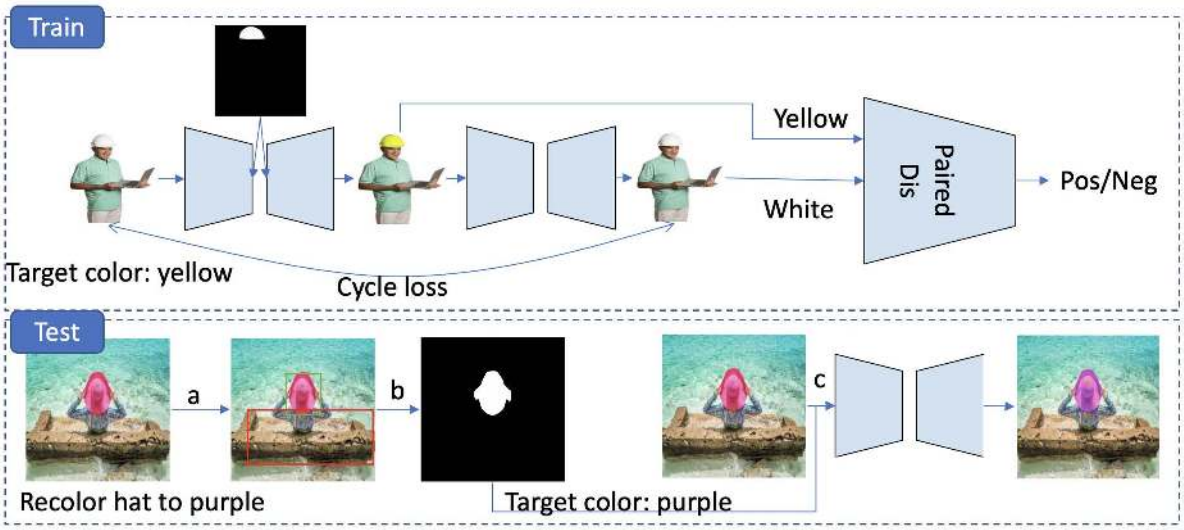

Figure 3. Top: The training process: based on dataset samples, we train a color transformer network with a combination of adversarial loss, cycle loss, and a novel paired discriminator. Bottom: The test process: a. Starting from an image and a command from user, we search for a matching bounding box (shown in green). b. We extract the soft mask from the bounding box. c. The original image, mask and target color forms the input to the color transformer that generates the recolored image with a forward pass. Images are from **paulovilela - stock ID #100153754**, **Alex Tor - stock ID #108213415**.

object of interest. The decoder layers, on the other hand, receive both the soft mask and the color mask as described above. The output is an image of the same shape of input image.

3.3. Paired discriminator

Introducing the paired discriminator is one of the main novelties of this paper. Since we want to design a network that can work with all possible tags and colors, we cannot use the traditional binary discriminators. One possible choice is to use conditional discriminators which did not produce good results in our setup (See Figure 7).

STARGAN-v2 [7] proposes an architecture for discriminator based on sharing the weights of first layers and using different heads for each domain. Since we want to just check the color of the image which is semantically simpler compared to style, and inspired by [18], we suggest using a paired discriminator that looks at two images at the same time and as a result can make better decisions. Our discriminator network has a shared visual feature extractor similar to ResNet50. We resize and concatenate the color masks to internal bottleneck layers 4, 5, and 6 of the ResNet50 architecture. Figure 8 shows our discriminator network architecture that takes a pair of images and their corresponding generated color masks as input and outputs if the pair is positive or negative. Finally, to make the network more robust against order of the inputs, we switch the image features by a random chance of 50%.

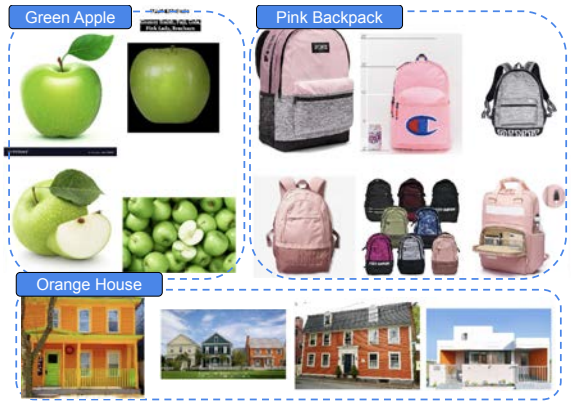

Figure 4. Example images in the dataset corresponding to the color / tag pairs Green / Apple, Pink / Backpack and Orange / House.

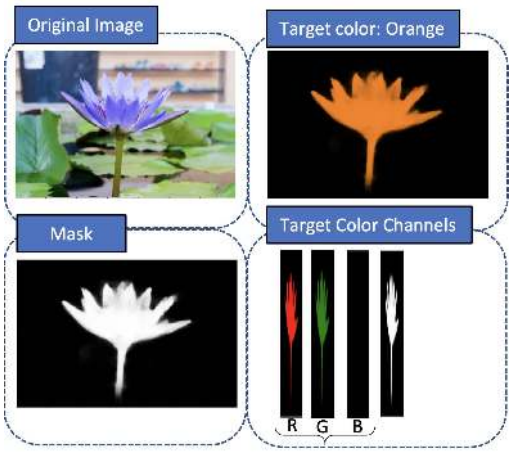

Figure 5. The procedure of generating target color channels. We have an image, a mask and the target color, orange. We generate an RGB mask based on orange RGB values and concatenate it with another channel which has mask data. Image is from **solahuddin - stock ID #100036041**.

3.4. Training and losses

We denote our color transform network with $\mathcal{G}$ and our discriminator with $\mathcal{D}$. During training, following notation introduced in section 3.1, we generate positive and negative pairs. For an anchor data point $\{x_i, t_i, c_i, m_i\}$, we sample a positive data point, $\{x_p, t_p, c_p, m_p\}$, such that $t_i = t_p$. One option to obtain negative pairs is to sample $\{x_n, t_n, c_n, m_n\}$ such that $t_i \neq t_n$. The other option is to sample two data points with the same tag and falsify their colors. We can falsify the color of anchor, negative data point, or both of them. Figure 8.b shows a couple of these generated pairs. Finally, we apply random cropping on x_i, x_p, and x_n.

Adversarial losses: Our paired discriminator takes two images and their corresponding color masks and tells if the two images have the same tag, and their color masks match the actual objects' colors. For example, the discriminator outputs 'one' if the inputs are an azure bicycle with an azure mask and a green bicycle with a green mask (we call it a positive pair). It outputs 'zero' if the inputs are an azure bicycle with an azure mask and a black bear with a black mask since the tags are different (we call it a negative pair). It also outputs 'zero' if the inputs are an azure bicycle with an azure mask and a green bicycle with a black mask since one of the color masks are wrong (a negative pair).

Therefore this binary classification can be formulated as:

$$\begin{aligned}\mathcal{L}_0 = \mathbf{E}[\log(\mathcal{D}((x_i, m_i, c_i), (x_i^p, m_i^p, c_i^p))] + \\ \mathbf{E}[\log(1 - \mathcal{D}((x_i, m_i, c_i), (x_i^n, m_i^n, c_i^n)))],\end{aligned} \quad (1)$$

where (x_i^p, m_i^p, c_i^p) is a triplet of image-mask-color from possible positive set for x_i and (x_i^n, m_i^n, c_i^n) is a triplet of image-mask-color from possible negative set for x_i.

Therefore, the adversarial loss looks like:

$$\mathcal{L}_1 = \mathcal{L}_0 + \mathbf{E}[\log(1 - \mathcal{D}((x_i, m_i, c_i), (x_i^t, m_i, c_t))], \quad (2)$$

where $c_t \in \{c_1, c_2, \cdots c_{14}\}$ is randomly drawn from our color set and $x_i^t = \mathcal{G}(x_i, m_i, c_t)$ is the recolored image. We also use cycle consistency to get the original image back:

$$\begin{aligned}\mathcal{L}_2 = \mathcal{L}_0 + \mathbf{E}[\|x_i - \hat{x}_i\|_1] + \\ \mathbf{E}[\log(1 - \mathcal{D}((x_i, m_i, c_i), (\hat{x}_i, m_i, c_i))],\end{aligned} \quad (3)$$

where $\hat{x}_i = \mathcal{G}(x_i^t, m_i, c_i)$ and $x_i^t = \mathcal{G}(x_i, m_i, c_t)$.

Identity losses: For the i-th image with color c_i if we recolor it with c_i, we should get the same image. Therefore, we can write a reconstruction loss as:

$$\mathcal{L}_3 = \mathbf{E}[\|x_i - \mathcal{G}(x_i, m_i, c_i)\|_1]. \quad (4)$$

Also, the recolored image should be the same as the input image everywhere except inside its mask. Therefore, we can add another reconstruction loss as follow:

$$\mathcal{L}_4 = \mathbf{E}[\|(x_i - \mathcal{G}(x_i, m_i, c_t)) \circ (1 - m_i)\|_1], \quad (5)$$

where $\circ$ represents pixel-wise product.

Full objective: The full objective is the summation of the losses described above:

$$\mathcal{L}(\mathcal{G}, \mathcal{D}) = \lambda_1 \cdot \mathcal{L}_1 + \lambda_2 \cdot \mathcal{L}_2 + \lambda_3 \cdot \mathcal{L}_3 + \lambda_4 \cdot \mathcal{L}_4, \quad (6)$$

where λ_1, λ_2, λ_3, and λ_4 are hyper-parameters that strike a balance between the different losses. The desired color transformer network can be found by solving:

$$\mathcal{G}^* = \arg\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}(\mathcal{G}, \mathcal{D}) \quad (7)$$

4. Experimental validation

4.1. Qualitative experiments

In the first set of experiments, we test whether the proposed recoloring process works. For this implementation,

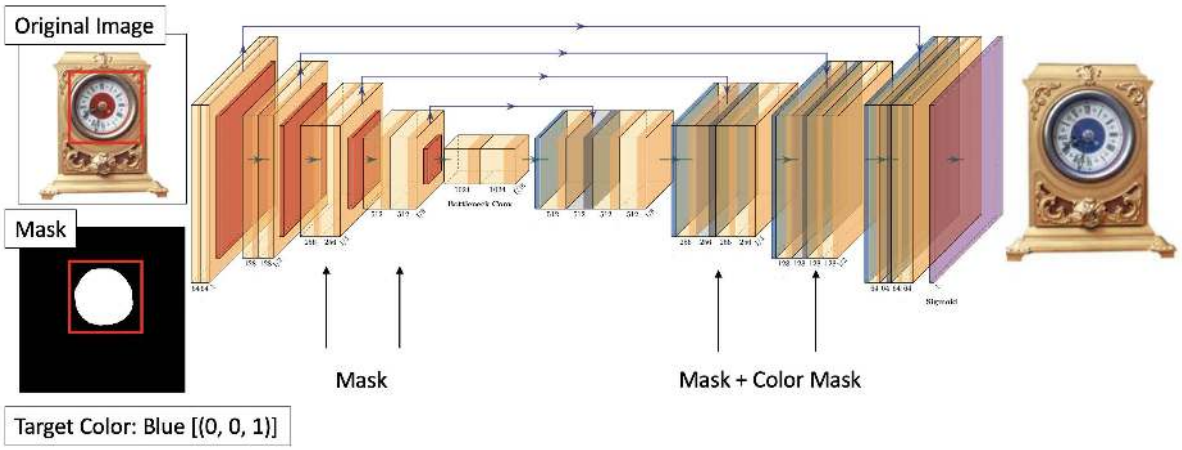

Figure 6. Color transformer network architecture. The network takes an image as input and has U-Net connections. For the encoder, we resize the soft mask and concatenate it with middle layers' features. For the decoder we concatenate both mask and target color mask with middle layers' features. Arrows show the concatenation operation. Image is from **guas - stock ID #100008804**.

Figure 7. Top: Original image and mask. Middle: Paired discriminator. Bottom: Conditional discriminator. Recoloring to blue, green, purple, orange and red from left to right. We saw that during training on RGB images, a conditional discriminator is not able to generalize well and most of the time learns to generate just one particular color. Images are from **FlexDreams - stock ID #112111227**, **romankosolapov - stock ID #111320345**.

we used a bounding box detector and tag extractors based on Faster-RCNN with hierarchical softmax trained on Open-Images 500 [12]. We used the algorithm described in [24] for extracting the saliency soft mask from the bounding box.

Figure 9 shows the qualitative results of applying the resulting network, for recoloring a towel and a potato respectively to a number of possible colors. We note that the recolored images maintain the details of the original object, even when recoloring to a color which does not naturally occur in nature (such a blue potato).

4.1.1 Comparing recoloring quality with state of the art baselines

The majority of the current state of the art recoloring algorithms do not use object masks. To evaluate the quality of the recoloring of our algorithm, we have adapted two state of the art recoloring algorithms to use masks, making them directly comparable with our proposed approach.

The first baseline we are considering is the method for primary color editing of product images by using color correction and color blending introduced in [9]. This method does not use deep learning and is able to change the primary color of the given image with any resolution. The approach performs clustering on the whole image and maps the primary cluster to target color cluster while making sure the colors in other clusters remain the same by optimizing for a mapping with these constraints. We introduce our mask information into this method by clustering only the pixels within our extracted mask and restricting the transferring of colors to the same area. Figure 10 shows the result of applying [9].

The second baseline we are considering is an algorithm designed for coloring grayscale images based on user input [29]. The user assigns the desired color to some of the pixels in the image and assigns a color to them. Based on the grayscale image and these inputs, the model generates a new colored image. To change the color of a specific object within the image, we randomly select a couple of points within the mask and set their color to target color and pass the generated user input and grayscale image to the model.

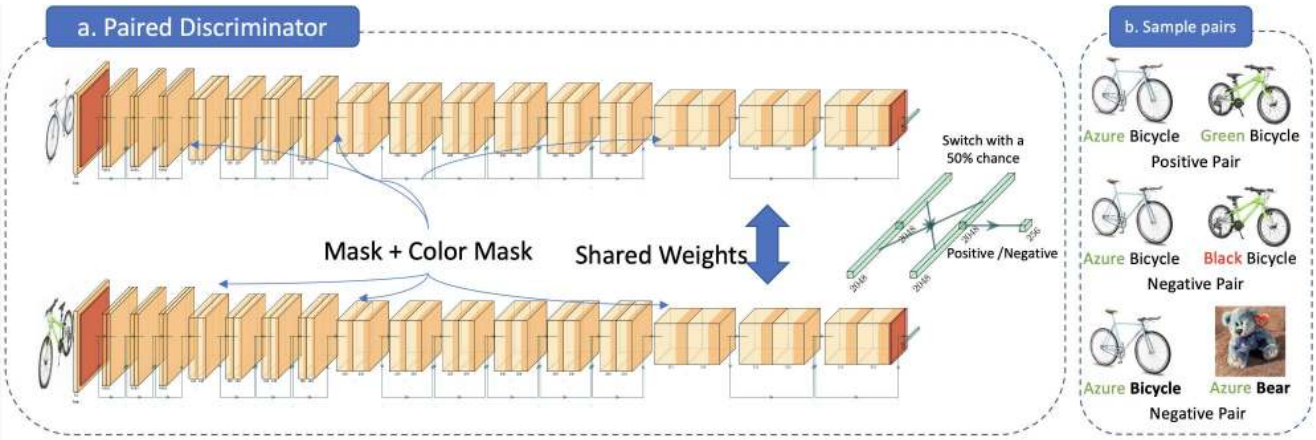

Figure 8. a. Discriminator network architecture. The network takes two images and passes them through ResNet50 initialized randomly. The color masks of the images are also concatenated with feature maps in bottleneck layers 4, 5, and 6. The discriminator switches the 2048 flattened outputs with a 50% chance. The concatenated features then go to a fully connected layer with size of 256 and then a fully connected layer that outputs positive/negative. b. A pair of images and their color and tag is positive if both tag and color of both images are correct. Otherwise the pair is negative. Along with these pairs, we also update the network with adversarial training.

Figure 9. Qualitative evaluation of the process to recolor a towel (top) and a potato (bottom) to a variety of colors. The leftmost image is the original, while the images in the right are recolored versions. Note that the recoloring process retains the details of the recolored image. Images are from **Studio KIVI - stock ID #100177220**, **pedphoto36pm - stock ID #100045692**.

Again we make sure to apply this method on just the masked object as previous one. In other terms, given image I, we apply these adapted methods on it and get image I'. Then we generate the final result as follows:

$$I_{final} = I' \cdot mask + I \cdot (1 - mask). \tag{8}$$

Figure 11 demonstrates the results for adapting [29]. Note that in the third column, we see the output of just applying this method on the image, and the fourth column is the output after applying equation 8. In Figure 12, we compare our method with proposed baselines.

4.2. Quantitative experiments

We compare our method with the proposed baseline on 200 images of our test dataset. We recolor each image to 14 different colors and evaluate Inception Score [21] on the generated images. In addition, we report the number of not detected objects plus the objects that are additionally detected in each image after recoloring. Table 1 shows the result of these experiments on the whole dataset. For more details, please refer to supplementary material.

Figure 10. The process of [9] with masks. Top left is the original image, top middle is the mask, top right is recolored to orange. Bottom images are the result of recoloring to purple, yellow, and blue from left to right. Even though this method is powerful with recoloring when hyper-parameters are adjusted (number of clusters and how many clusters to edit), make it fully automatic is challenging since the same number of clusters does not always work. In this case, the input is fixed and only the target color changes, it is even more challenging to fix hyper-parameters for all possible input images. Image is from **allamaistrenko - stock ID #101639333**.

Table 1. Quantitative comparison of our method with proposed baselines on Inception Score (IS) and the number of mismatched detected objects (# MDO).

Metric	Oracle	FACT(ours)	Adapted [29]	Adapted [9]
# MDO	0	**83.36**	85.00	106.14
IS	8.75	**8.31**	8.22	7.94

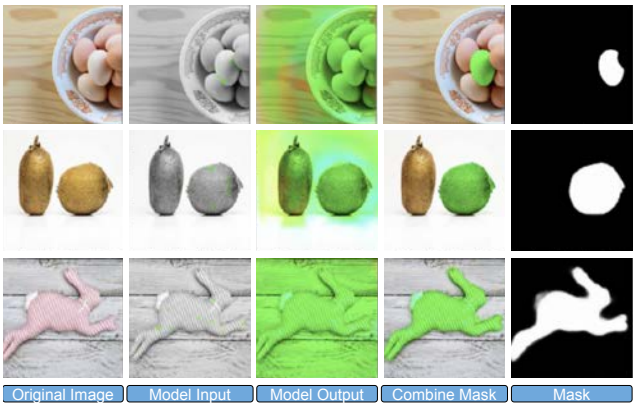

Figure 11. The process of guided colorization [29] with masks. First, we convert the image to grayscale and sample 10 point within the mask. We set the value of those point to target color and apply [29] on the image. The result is shown in third column. We combine the result with mask information and real image which will give us the fourth column. Since we do not know which pixels within the image correspond to main color, selecting random points may cause erasing of original texture which is not expected. Images are from **Andrey Solovev - stock ID #100133198**, **framefts - stock ID #100133206**, **Sandra Thiele - stock ID #100134232**.

Figure 12. Comparing our method with proposed baselines. Left column is the original image. The second column from left is the result of recoloring with [9]. The third column is adaptation of [29], and fourth column is FACT results. For more images please look at supplementary material. Images are from **lotosfoto - stock ID #104279367**, **allamaistrenko - stock ID #101639333**, **BRD - stock ID #95198050**, **khemfoto - stock ID #104131139** .

4.3. Robustness against mask perturbation

To evaluate the robustness of our algorithm, we performed a study by perturbing the mask. Instead of detected mask, we provide perturbed mask to our color transformer network. Figure 13 and Figure 14 show the results for this experiment. Please refer to supplementary material for more images. We saw our method is robust to up to 15% mask distortion and up to 5% background perturbation. These results are interesting since we did not apply any perturbation during the training of the network.

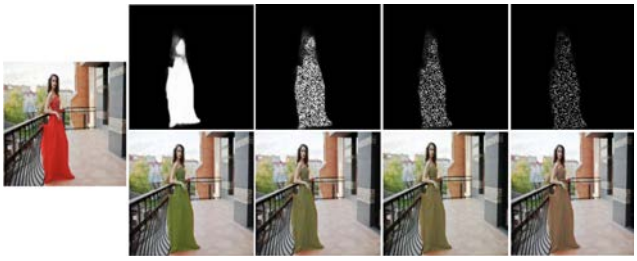

Figure 13. Effect of perturbation of mask on the generated recolored image. Left original image, perturbation is 0%, 10%, 20% and 30%. Image is from **AS Photo Project - stock ID #109730761**.

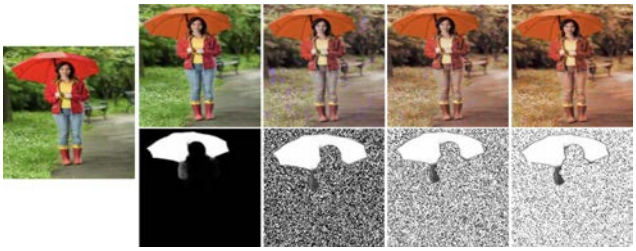

Figure 14. Effect of background perturbation on the generated recolored image. Left original image, perturbation is 0%, 10%, 20% and 30%. Image is from **bokan - stock ID #110692886**.

5. Conclusions

We proposed a method based on GANs for automatic object re-colorization. Our method enables the user to give any command of form: recolor <object> to <color> to the system and it does not require any manual edit by user. We compared our method with two state of the art baselines. Future direction of this work will include the application of the method to higher resolution images.

6. Acknowledgement

We thank Baldo Faieta, Richard Zhang, Tracy King and the anonymous reviewers for their feedback on this paper. We are also thankful to Adobe Sensei and Search department for providing computational resources for this research. List of photo owners can be found in section 7.4 of supplemental material.

References

[1] Selenium. `http://www.openqa.org/selenium`.

[2] Mahmoud Afifi, Brian L Price, Scott Cohen, and Michael S Brown. Image recoloring based on object color distributions. In *Eurographics (Short Papers)*, pages 33–36, 2019.

[3] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. Palette-based photo recoloring. *ACM Trans. Graph.*, 34(4):139–1, 2015.

[4] Guillaume Charpiat, Matthias Hofmann, and Bernhard Schölkopf. Automatic image colorization via multimodal predictions. In *European Conf. on computer vision*, pages 126–139, 2008.

[5] Junho Cho, Sangdoo Yun, Kyoung Mu Lee, and Jin Young Choi. Palettenet: Image recolorization with given color palette. In *Proc. of the IEEE Conf. on computer vision and pattern recognition workshops*, pages 62–70, 2017.

[6] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proc. of the IEEE Conf. on computer vision and pattern recognition*, pages 8789–8797, 2018.

[7] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proc, of the IEEE/CVF Conf. on computer vision and pattern recognition*, pages 8188–8197, 2020.

[8] H Sheikh Faridul, Tania Pouli, Christel Chamaret, Jürgen Stauder, Erik Reinhard, Dmitry Kuzovkin, and Alain Trémeau. Colour mapping: A review of recent methods, extensions and applications. In *Computer Graphics Forum*, pages 59–88, 2016.

[9] Han Gong, Luwen Yu, and Stephen Westland. Simple primary colour editing for consumer product images. *arXiv preprint arXiv:2006.03743*, 2020.

[10] Revital Ironi, Daniel Cohen-Or, and Dani Lischinski. Colorization by example. In *Rendering Techniques*, pages 201–210, 2005.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proc. of the IEEE Conf. on computer vision and pattern recognition*, pages 1125–1134, 2017.

[12] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.

[13] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on graphics*, pages 1–11, 2014.

[14] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM Siggaph 2004 Papers*, pages 689–694. 2004.

[15] Xihui Liu, Zhe Lin, Jianming Zhang, Handong Zhao, Quan Tran, Xiaogang Wang, and Hongsheng Li. Open-edit: Open-domain image manipulation with open-vocabulary instructions. *arXiv preprint arXiv:2008.01576*, 2020.

[16] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *Proc. of the IEEE Conf. on computer vision and pattern recognition*, pages 4990–4998, 2017.

[17] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[18] Saeid Motiian, Quinn Jones, Seyed Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. In *Advances in neural information processing systems*, pages 6670–6680, 2017.

[19] Saeid Motiian, Quinn Jones, Stanislav Pidhorskyi, and Gianfranco Doretto. Unsupervised learning of paired style statistics for unpaired image translation. In *CVPR Workshops*, pages 112–121, 2019.

[20] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM Transactions on Graphics (TOG)*, pages 1214–1220, 2006.

[21] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[22] Daniel Sỳkora, John Dingliana, and Steven Collins. Lazybrush: Flexible painting tool for hand-drawn cartoons. In *Computer Graphics Forum*, pages 599–608, 2009.

[23] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. Transferring color to greyscale images. In *Proc. of the 29th annual Conf. on Computer graphics and interactive techniques*, pages 277–280, 2002.

[24] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas Huang. Deep GrabCut for object selection. In *Proc. of the British Machine Vision Conference*, pages 182.1–182.12, 2017.

[25] Liron Yatziv and Guillermo Sapiro. Fast image and video colorization using chrominance blending. *IEEE transactions on image processing*, pages 1120–1129, 2006.

[26] Seungjoo Yoo, Hyojin Bahng, Sunghyo Chung, Junsoo Lee, Jaehyuk Chang, and Jaegul Choo. Coloring with limited data: Few-shot colorization via memory augmented networks. In *Proc. of the IEEE Conf. on computer vision and pattern recognition*, pages 11283–11292, 2019.

[27] Bo Zhang, Mingming He, Jing Liao, Pedro V Sander, Lu Yuan, Amine Bermak, and Dong Chen. Deep exemplar-based video colorization. In *Proc. of the IEEE Conf. on computer vision and pattern recognition*, pages 8052–8061, 2019.

[28] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conf. on computer vision*, pages 649–666, 2016.

[29] 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. *arXiv preprint arXiv:1705.02999*, 2017.

[30] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. of the IEEE Int'l conf. on computer vision*, pages 2223–2232, 2017.