NORTHWESTERN
UNIVERSITY

Computer Science Department

Technical Report
Number: NU-CS-2023-08

March, 2023

Multi-Stage Automatic Line-Art Colorization with Style and Color Priors

William Daniels

Abstract

In this work, we propose the SALAC model, a fully automatic method for character line-art colorization that utilizes concepts from the style extraction and color distribution literature. This framework allows for fully unannotated colorizations of line-art sketches in varied and non-deterministic color styles. We show that the proposed approach outperforms previous GAN-based automatic line-art colorization methods while retaining flexibility with regard to the output color distribution.

Keywords

Line-Art, Sketch, Automatic, Colorization, GAN, Stage, Segmentation

NORTHWESTERN UNIVERSITY

Multi-Stage Automatic Line-Art Colorization with Style and Color Priors

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

MASTER OF SCIENCE

Computer Science

By

William Daniels

EVANSTON, ILLINOIS

March 2023

ABSTRACT

Deep generative models have long been able to perform increasingly complex tasks, and recent work in image generation has been extended to sketch colorization. Because of the increased difficulty of generalizing from a low-information "line-art" sketch input, much of this recent work has relied on user supplied information to the colorization model, thereby making the process semi-automatic. In this paper, we propose the SALAC model, a fully automatic method for character line-art colorization that utilizes concepts from the style extraction and color distribution literature. This framework allows for fully unannotated colorizations of line-art sketches in varied and non-deterministic color styles. Generations from this model are compared to state-of-the-art automatic approaches in the relevant sketch colorization literature. We show that the proposed approach greatly outperforms previous GAN-based automatic line-art colorization methods while retaining flexibility with regard to the output color distribution.

ACKNOWLEDGEMENTS

I would like to thank Dr. Aggelos Katsaggelos for giving me the opportunity to engage in this research and for taking the time to explore and discuss my work with me. I'm very grateful for Dr. Emma Alexander's guidance in the last stages of this work, as it was a substantial help to me as this thesis concluded. I also want to thank Dr. Mohammed Alam for giving constructive input and for also leading thought-provoking and exploratory classes given here at Northwestern.

A special acknowledgement and heart-felt thanks goes out to my loving and immensely supportive family, especially my parents Kathy Daniels and Bill Daniels, who have adequately and thoughtfully prepared me for this point in my life. Without them, this work and the path I've taken would not have been possible. I am also thankful for the lifelong support and love from my grandparents Pam Daniels and the late Walter Daniels, who have always had me in mind.

List of Abbreviations

FID: Fréchet Inception Distance

GAN: Generative Adversarial Network

HSV: Hue-Saturation-Value

ReLU: Rectified linear unit

RGB: Red-Green-Blue

SC: Style Clusters

SOTA: State-of-the-art

SRU: Super-resolution upscaler

TBS: Trapped-ball segmentation

XDoG: Extended Difference-of-Guassians

TABLE OF CONTENTS

Acknowledgments 3

List of Algorithms 8

List of Figures 9

List of Tables 11

Chapter 1: Introduction 12

Chapter 2: Related Work 15

2.1 GAN-based line-art colorization 15

2.2 Line-art extraction algorithms 16

2.3 Unsupervised Image Segmentation 17

2.4 Color difference and color distribution representations 19

Chapter 3: The SALAC model 21

3.1 Model Overview 21

3.2 Semantic Segmentation Stage 22

3.2.1 Low-level Segmentation 22

3.2.2 High-level Segmentation 22

3.3 Color Proposal Stage 24

3.3.1 Color Category Prediction 24

3.3.2 Generative Color Proposal 27

3.4 Refinement Stage 32

Chapter 4: Experimentation and Results 33

4.1 Dataset and Data Preparation 33

4.2 Training Settings 34

4.3 Results and Evaluation 35

Chapter 5: Conclusions and Future Work 38

5.1 Contributions and Significance 38

5.2 Limitations and Future Work 39

References 45

Appendix A: Output Comparisons 47

LIST OF ALGORITHMS

1 CreateColorProposals 26

2 CreateColorTBS 28

LIST OF FIGURES

2.1 Example of various XDoG outputs 17

2.2 Visual example of trapped-ball segmentation 18

2.3 The "Segmentation in Style" segmentation process 19

3.1 The SALAC model flow visualized 22

3.2 Visualized output of the segmentation stage 24

3.3 Color category proposal visualization 25

3.4 An abstraction of the color proposal creation process 26

3.5 TBS and color-filled TBS ground truth 28

3.6 Generative color proposal output 31

3.7 Differences in image domain before refinement 32

4.1 An example final output from the SALAC model 36

4.2 Selected output examples from a fully trained SALAC model 37

A.1 Selected output examples from AlacGAN 48

A.2 Selected output examples from SALAC (generative only) 49

A.3 Varying color outputs for one sketch 50

A.4 Visualization of color clusters through random sampling 51

A.5 Examples of SALAC failure modes 52

LIST OF TABLES

4.1 Quantitative comparison of FID scores 36

CHAPTER 1
INTRODUCTION

Line-art colorization is a critical undertaking when attempting to create illustrations from character sketches, but it is also a challenging task that requires hours of professional work for just one colorization. This presents a dilemma as it raises the barrier to entry for creating these types of illustrations and reduces productivity in the design process for those left who can clear that barrier. In response to these issues, techniques from the deep image generation literature have been adapted in an attempt to automate the colorization portion of the illustration process. Results from a naïve approach using these techniques are subpar as line-art sketches are very low information priors for deep generative models.

Recent line-art colorization approaches have remedied this with frameworks that supply the generative model with supplemental information along with the input sketch. The most popular of these additional information sources include user-inputted color tags [1], reference images to guide the network in terms of output style and color [2], [3], and user-guided color scribbles to accurately distribute colors to the correct semantic regions in the output image [4]–[7]. Due to these supplementary methods, output from these models show improved colorization on quantitative measurements like FID (Fréchet inception distance) [8] and qualitative user evaluations.

Though the aforementioned approaches perform better for line-art colorization, they trade this increased performance against the ease of use when generating samples. For instance, if using a model where the supplemental prior is tag-based, the user must manually input the relevant semantic color tags ("white shoes", "blue cap", etc.) for each generated image.

Furthermore, these approaches disincentivize exploration of varying color styles per input image, as inference for each different color style must be accompanied by the respective supplementary information. For example, for the color-scribble approach the user would have to redraw the scribble information in varying colors for each differently colored output the user wants to explore. So, while these models lower the previously mentioned barrier to entry for line-art illustration, there is an opportunity for a great productivity increase through a fully automatic method.

Historically, fully-automatic methods have exhibited the inverse tradeoff compared to user-guided approaches by trading performance (plausible colorizations) for efficiency (time and effort per colorization). Predictably, this is due to the generator being given too complex a task given the current state of generative adversarial network (GAN) models: output a plausible line-art colorization render with requisite shading and polish given a greyscale 2D raster of strokes that designate outlines of semantic information. The majority of previous attempts focus on prediction given one pass through a network, but we suggest that a framework of this kind is not merely a colorization network, but implicitly a segmentation and refinement network as well. In this paper we propose a framework for fully-automatic line-art colorization that rests on the intuition that this task should be deconstructed into three separate tasks to avoid relying on one function to learn all three tasks simultaneously: 1) explicit semantic segmentation, 2) color proposal, and 3) refinement.

In the first task, through inspiration from the graphics vectorization and style extraction literature, we extract both low- and high-level semantic segmentation information from the input line art. Line-art images contain no gradient information compared to many other domains of images, and it is therefore much harder for a model to differentiate low-level regions (groups of pixels enclosed by a line or sketch boundary), which is helpful as there is

usually one particular color per region in line-art illustrations. While this low-level region information can help improve colorizations [9], the model also needs an understanding of higher level regions (groups of low-level regions) that may be semantically relevant. For example, in complicated line-arts hair renditions may be comprised of hundreds of enclosed regions, but they should be assigned to the same color cluster. The second task draws on the color distance and distribution literature to generate color proposals for each low-level segment. One model translates low and high level information to per-pixel predictions of a predefined color category, and a follow-up model generates an intermediate coloring from this information. Finally, for the third task, this proposal is used as a prior to another model that generates the refined colorization output.

The main contributions of this paper are summarized as follows:

- We propose the SALAC (Multi-**S**tage **A**utomatic **L**ine-**A**rt **C**olorization) model, a multi-stage framework for fully-automatic character line-art colorization.
- We provide a mini model zoo of trained models for each subtask within the SALAC model framework.
- We make available an altered dataset of filtered, centered, and preprocessed 512x512-pixel line-art character illustrations.
- Quantitative evaluation comparisons that show that our model outperforms baseline GANs and the state-of-the-art (SOTA) (to the authors knowledge) previous fully-automatic GAN-based approach.
- Possible improvements, insights, and further exploration that may be useful when trying to improve the results of automatic line-art colorization models.

CHAPTER 2
RELATED WORK

2.1 GAN-based line-art colorization

Advancements in image generation frameworks over the past decade have resulted in their application to conditional image generation problems including – but not limited to – sketch colorization. Generative Adversarial Networks (GANs) [10]–[12], a machine learning framework where a generator attempts to output samples from a training distribution that fool a separate "discriminator", have achieved state of the art performance on generation tasks. Much of the GAN-based image-to-image translation literature traces back to Isola et al. and their work on "Image-to-Image Translation with Conditional Adversarial Networks." This paper introduced pix2pix [13], a powerful framework for training generative networks that takes an input image as a prior to create a structurally similar counterpart belonging different domain. Pix2pix takes advantage of a U-Net architecture [14] and a patch-based discriminator to achieve state of the art performance on image-to-image translation tasks. Future line-art colorization approaches have heavily drawn from pix2pix as the basis for the sketch-to-color training paradigm.

AlacGAN [4] takes advantage of both a pretrained sketch feature extractor and user-defined color hints to guide the colorization generator with essential semantic and color information. Specifically, ReLU activations from an intermediate layer of the pretrained Illustration2Vec [15] network, and color hints in the training set are simulated by sampling random pixels from the target image. In regard to GAN training, the Wasserstein [16]–[18]

(earth-mover) distance is used as the discriminator loss metric.

In contrast to using color hints as supplementary information, the Tag2Pix network [1] utilizes a paired dataset of line-arts and tags (color specific and semantic) are used to guide the generator. An in-house pretrained classification network is used to label purely semantic tags called "color invariant tags" (CITs) of the input line-art. CITs may include tags such as "hat", "glasses", and "backpack". CITs are supplied to the lowest levels of the generator decoder. On the other hand, "color variant tags" (CVTs) are user supplied and encoded into multiple layers of the generator decoder. CVTs may include tags such as "blue hat", "white background", and "yellow eyes."

2.2 Line-art extraction algorithms

The historically preferred approach for compiling sketch-to-color line-art datasets has been synthetic creation of line-art sketches from running sketch extraction algorithms on line-art illustrations as opposed to gathering pairs of real sketches and real colored illustrations on those sketches. A worry of using these methods is a type of input domain overfitting by colorization models on the styles of synthetic line-arts. Tag2Pix used a combination of three different extraction approaches to avoid overfitting, while AlacGAN mixed a small portion of real line-art sketches of varying styles in with synthetic ones.

The most extensively used sketch extraction algorithm is an extended formulation of the difference-of-Gaussians operator, or XDoG [19]. With the correctly tuned parameters, XDoG can render a colored line-art as a convincing pre-colored representation. Recently, late-resizing [20] has been proposed to mimic the original sketches even more closely by extracting sketches with XDoG before resizing, which results in a much slower extraction procedure but a more convincing output. Tag2Pix authors mention a similar strategy in their

work, even going as far to upscale images with a custom illustration-based super-resolution upscaler [21] (referred to as SRU) before extraction.

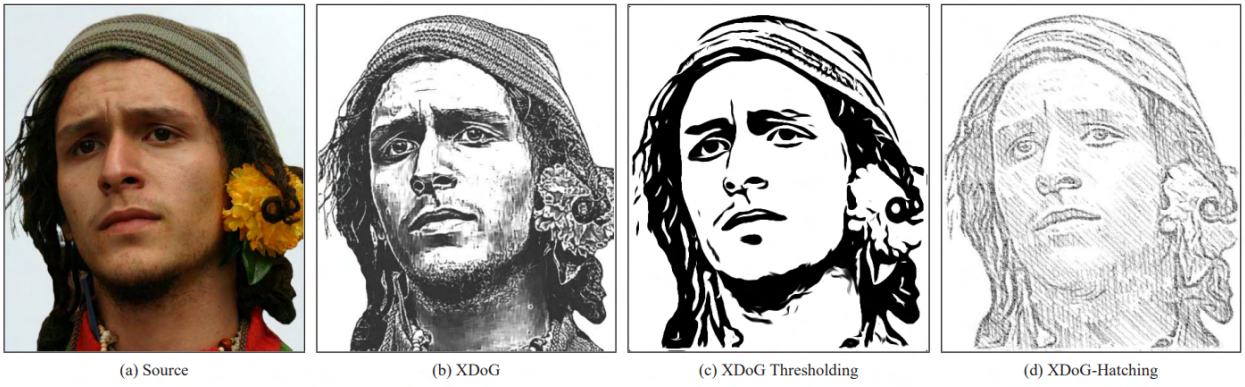
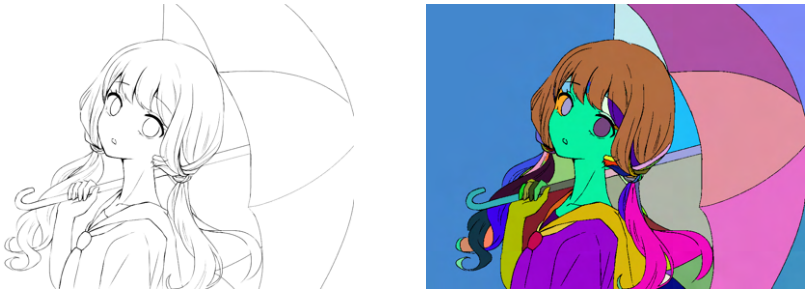


Figure 2.1: Various XDoG outputs as visualized in "XDoG: An eXtended difference-of-Gaussians compendium including advanced image stylization."

2.3 Unsupervised Image Segmentation

There are many applications where region segments that signify closed areas (areas in an image that have a sharp pixel outline or boundary) are useful as supplementary information to many types of models. Trapped-ball segmentation [22] is an algorithm designed specifically for segmentation of regions in a cartoon frame that are closed by sketch boundaries. Through the use of a set of morphological operations on an image, large and enclosed regions (whose size is identified by some starting parameter) are identified and filled with a unique value. When all regions are filled, smaller regions are considered. This loop continues until all regions in the image are filled.



(a) Line-art sketch (b) Trapped-ball segmentation

Figure 2.2: Visual example of trapped-ball segmentation.

The StyleGAN [23]–[26] family of models have historically achieved impressive results on image generation tasks with narrow domains (faces, landscapes, etc.) The success of these models is attributed to cleverly injected style information and progressive growing [18] of generated images. Pakhomov et al. proposed an unsupervised segmentation approach that probes the StyleGAN2 architecture to extract semantically meaningful clusters [27]. By using any generic clustering algorithm in the feature space of a trained StyleGAN model (here, the 7th or 9th layers of an unmodified StyleGAN2 model), semantic segmentation masks can be collected for each generated image. Clusters from layers closer to the output appear finer in detail but have less semantic meaning, while clusters closer to the input layer are coarse but more semantically relevant.

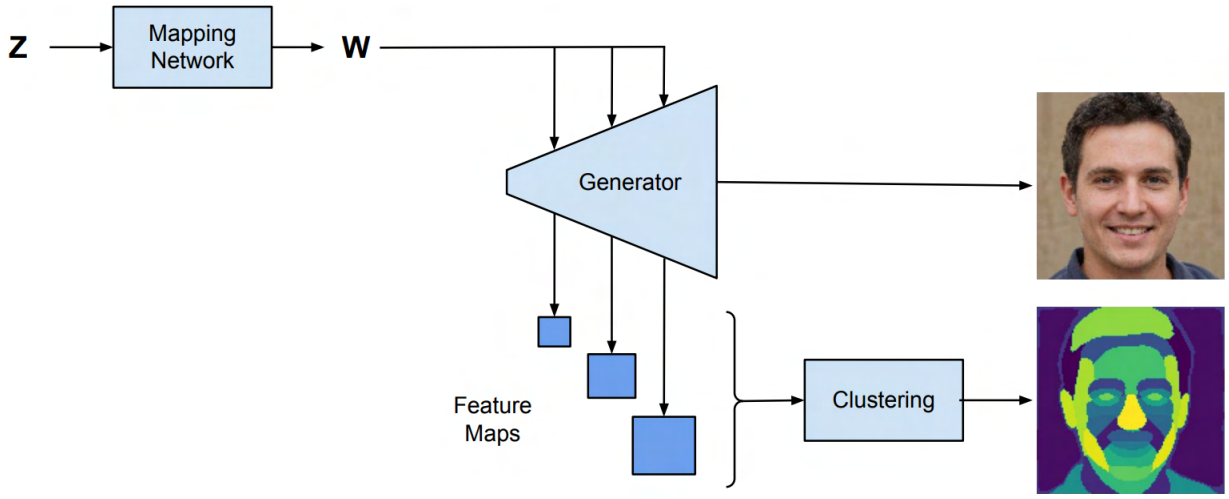


Figure 2.3: The segmentation process as visualized in "Segmentation in Style: Unsupervised Semantic Image Segmentation with StyleGAN and CLIP."

2.4 Color difference and color distribution representations

Color distance (or color difference) is the term used to describe the amount of separation between two colors in an arbitrary color space. Standard distance metrics like Euclidean distance are poor representations of color difference when the "separation between two colors" is likely to be gauged perceptually (visually by humans) instead of by strict difference in RGB (or some other color) space. Perceptual color distance metrics such as CIEDE76, CIEDE94 [28], and CIEDE2000 [29] attempt to construe the color distance problem such that differences between colors are subjectively agreeable visually.

Color distributions in images can be represented in various ways. Condensed representations such as color palettes [30] are useful for signaling the top-n most common colors in an image, but can be less helpful when the most common colors are not easily clustered into n categories, such as when the color distribution approaches uniformity. More information-saturated methods like color histograms can represent color distributions in two or more

dimensions. Condensed representations of image color can be utilized to guide generative models in the direction of the color distribution for an output image. In the case of HistoGAN [31], an image is converted to the log-chroma space [32] with respect to all RGB channels of the image, yielding three 2-dimensional histograms that are combined into a 3-dimensional histogram. This representation can made differentiable by using a learnable kernel to control bin thresholding [33].

CHAPTER 3
THE SALAC MODEL

3.1 Model Overview

Input to the SALAC model, and therefore the segmentation stage, is a greyscale 512x512-pixel image. A trapped-ball segmentation map (referred to as TBS map) is created sequentially in accordance with Zhang et al.'s [22] usage. The sketch and TBS map are passed to two trained style cluster (referred to as SC) segmentation algorithms to retrieve corresponding SCs.

The color proposal stage takes a sketch, TBS map, and SC batch as input from the segmentation stage output. A trained per-pixel segmentation algorithm then predicts one of six color categories for each pixel on the input sketch, resulting in a rough color proposal for the sketch. This rough color proposal is sent to a generative (GAN-trained encoder-decoder architecture) color proposal model for a more detailed color proposal per segment of the TBS map.

The final refinement stage, using a separate generative model, takes a sketch and the refined color proposal from the output of the color proposal stage and pushes the generated output towards the target illustration distribution. The outputs of this stage are the outputs of the SALAC model.

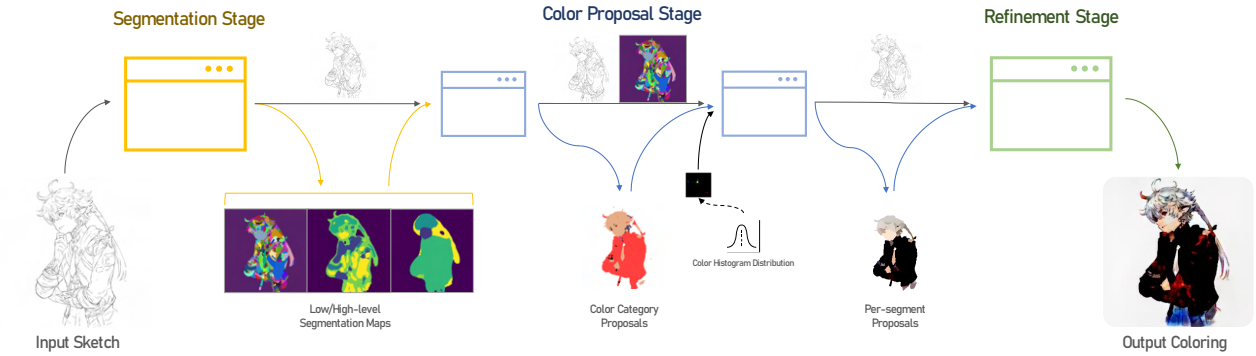


Figure 3.1: The SALAC model flow visualized.

3.2 Semantic Segmentation Stage

3.2.1 Low-level Segmentation

For low-level segmentation, a Python implementation [34] of the original trapped-ball segmentation algorithm is used to extract region-level segments from a sketch input. This algorithm is deterministic and not learned, and therefore does not require training.

3.2.2 High-level Segmentation

For high-level segmentation, a method for extracting segmentation maps from StyleGAN models is considered [27]. First, a pretrained, modified StyleGAN2 model [24] is fine-tuned on a separate partition of 500 hand-selected images from the target illustration distribution until convergence. Two k-means clustering models are trained on the intermediate output features for the 7th and 9th layer respectively. These clustering models can predict clusters from any image generated by the aforementioned fine-tuned generator, but we want to able to predict semantic clusters from arbitrary sketch inputs to our SALAC model. To achieve this, we will transform the StyleGAN generator outputs to sketches with the XDoG algorithm

(more on this in section 4.1) and train two semantic segmentation models (for the 7th and 9th feature layers respectively) to predict the semantic clusters from transformed sketches.

We generate and save 5,000 images from the fine-tuned StyleGAN model, along with the k-means predictions for the clusters of both the 7th and 9th layer for each image. TBS maps for each sketch are also gathered. We use a modified version of a U-Net-like network called RefineNet [35], [36] for the segmentation task. Inputs to the models are the StyleGAN-generated image's extracted sketch and TBS map. Outputs are 7x512x512- and 6x512x512-integer array segmentation masks for the 7th and 9th layer, respectively. We train two semantic segmentation models on these 5,000 sketch-to-cluster pairs, one to predict 7th-layer clusters and one to predict 9th-layer clusters. During training, image augmentation is used to synthetically increase the size of the dataset and avoid overfitting.

To optimize our segmentation model, we minimize the cross entropy loss ℓ between model predictions x and target clusters y:

$$l_n = -\sum_{c=1}^{C} \log \frac{\exp(x_{n,c})}{\sum_{i=1}^{C} \exp(x_{n,i})} y_{n,c}$$

$$\ell(x, y) = \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} 1} l_n$$

where C is the number of classes per cluster, and N is the batch size.

(a) Line-art sketch (b) TBS map (c) 7th-layer clusters (d) 9th-layer clusters

Figure 3.2: Visualized output of the segmentation stage.

3.3 Color Proposal Stage

3.3.1 Color Category Prediction

For the color proposal stage, we again consider a model that will act as a per-pixel segmentation algorithm, predicting a class of color categories that pixels from the input image belong to. Six color categories are selected subjectively on perceived value as a color proposal for line-art colorizations: a black spectrum, white spectrum, brown spectrum, "red" spectrum (red/yellow/orange), "blue" spectrum (blue/green/purple), and fair/pink spectrum.

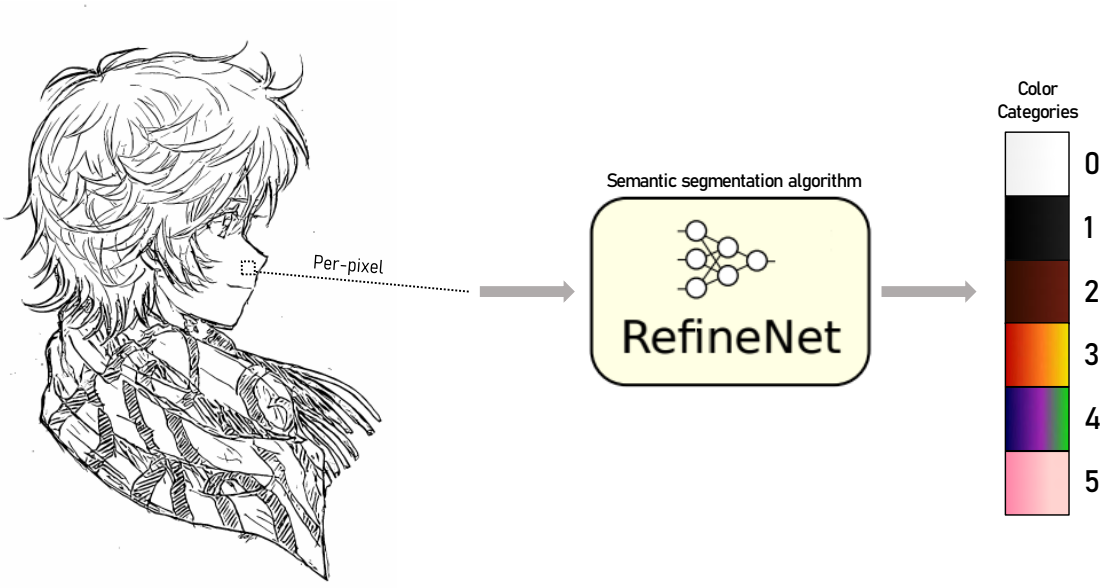


Figure 3.3: Color category proposal visualization.

To train a model that maps an input sketch, TBS map, and SC to a color proposal mask, we need sketch and color proposal mask training pairs. To do this we formulate an algorithm for translating the final target illustrations in our training set to color proposal masks. We utilize the CIEDE2000 color distance formula [29] to assign pixels in line-art illustrations to the predefined color category bins. For example, for a pixel at HxW, where H is the row and W is the column in a target RGB illustration array, its distance from every pixel value for a color category is calculated. The category color that is closest to the illustration pixel considered is assigned to the same HxW coordinate in a new image array. Each color proposal mask is a 6x512x512-integer array.

We optimize this segmentation model in the same way as the last: minimizing the cross entropy loss ℓ between model predictions x and target categories y.

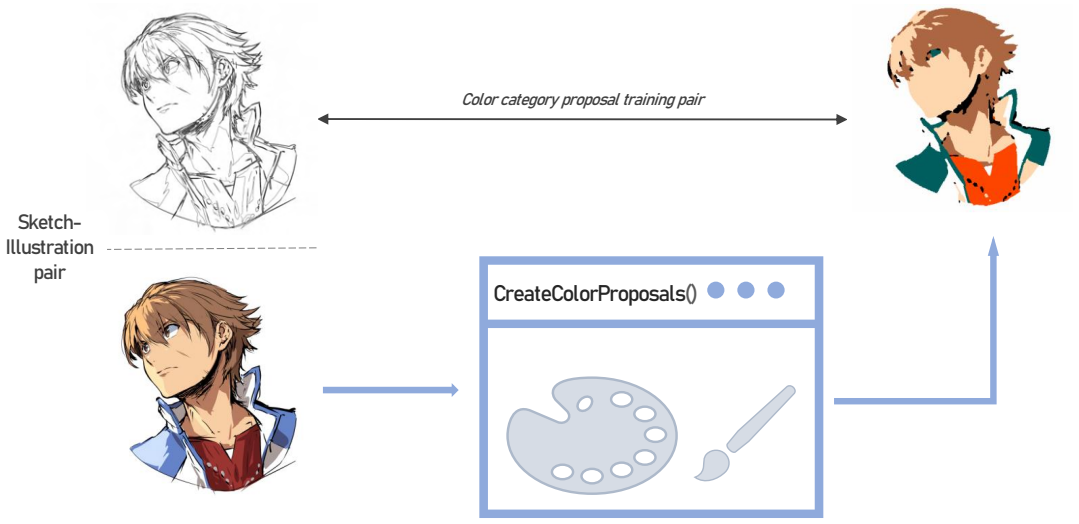


Figure 3.4: An abstraction of the color proposal creation process. For visualization purposes, the output shown is a RGB representation of the mask output.

Algorithm 1 CreateColorProposals

procedure CREATECOLORPROPOSALS(I, C)
  $I \leftarrow rgb2lab(I)$ ▷ Transform image to the CIELAB color space
  $D \leftarrow M + \lambda_1$ where $M \in \mathbb{R}^{\mathbb{K},\mathbb{I}_\mathbb{H},\mathbb{I}_\mathbb{W}}$ ▷ Initialize a distance matrix
  for $k \leftarrow 0$ `to` K **do**
    $c \leftarrow Fill(\text{index}(C, k))$ ▷ HxW array filled with k-th color
    $distance \leftarrow CIEDE2000(c, I)$ ▷ Per-pixel color distance for k-th category
    `index`$(D,\ k) \leftarrow distance$ ▷ Update distance matrix
  end for
  $mins \leftarrow \arg\min_K D$ ▷ Return indices of smallest distances along K
  $mask \leftarrow$`index`$(Identity,\ mins)$ ▷ Create K masks from HxW "mins" array
  return $mask$ ▷ Return a KxHxW color proposal mask
end procedure

The creation process for color proposal masks to train against is formulated in Algorithm 1, and the symbols used are defined here:

- I: Input colored illustration to bin pixels based on.
- C: Array of colors that represent category bins.
- K: Number of categories C.
- M: Empty KxI_HxI_W array where H/W are height/width of I.
- λ_1: Initial distance value of 1000.
- *CIEDE2000*: The CIEDE2000 color distance formula.
- *Identity*: The Identity matrix.

Again, a modified RefineNet is used as the network for learning this color proposal, with inputs to the model being a sketch, TBS map, and style cluster batch.

3.3.2 Generative Color Proposal

For more detailed color proposal information, a second model is proposed that takes the color proposal masks (plus sketch and TBS map) as a prior and generates an altered version of a TBS map with color proposals for each region segment. As with the previous color proposal model, we require a ground truth to train against. For each training sketch, a color-filled version of the TBS map is created, which fills each segment with the median color value in the same region of the illustration instead of a unique region label.

(a) Existing TBS map

(b) Color-filled (ground-truth) TBS map

Figure 3.5: Creating ground truths for a more detailed color proposal generation model.

Algorithm 2 CreateColorTBS

1: **procedure** CreateColorTBS(F, C)
2: $S \leftarrow populate(F)$ ▷ Create mapping of segments to lists of 2-D image coordinates
3: $M \leftarrow getColor(S, C)$ ▷ Get median segment color given coordinates w.r.t. C.
4: $rgb \leftarrow getRGB(M)$ ▷ Project median color tuples to their respective coordinates.
5: **return** rgb ▷ Return an RGB image array.
6: **end procedure**

The creation process for color-filled TBS maps to train against is formulated above, and the symbols used are defined here:

- F: Input TBS (or fill) map.
- C: Corresponding colored illustration.
- S: Data structure for mapping segment labels to a list of 2-D coordinates the segment is assigned to (e.g. '2': [[245, 72], ...], where key '2' is the segment/region label and value [[245, 72], ...] is a list of coordinates).

- *getColor()*: Function that, for each region label (key) in S, gets the median out of all colors retrieved by masking C by the coordinates (value) the segment is assigned to.
- M: Mapping of (median) color tuple to the 2-D coordinate that color belongs to in the color-filled TBS.
- *getRGB()*: Translates M into an RGB image (the color-filled TBS map) by filling an empty RNG image array with the red, green, and blue values at their assigned 2-D coordinates.
- *rgb*: Output color-filled TBS map which will act as the ground truth for the current algorithm.

To enable our model to propose colors from the entire RGB spectrum, we structure the learning problem as a generative task in a GAN framework. Here, a generator $\mathcal{G}$ will create a new proposal P' conditional on the prior P color category proposal (plus the carried over sketch and TBS map) and a color histogram [33] derived from the target color-filled TBS map. This new color histogram feature is helpful for guiding the generator toward the correct color distribution, and can be any sampled histogram at inference time. The generator is optimized by minimizing a loss function which is a combination of a structure loss, color loss, and adversarial loss. The generator loss function is formulated as:

$$\mathcal{L}_{\mathcal{G}} = \mathcal{L}_{struct} * \lambda_1 + \mathcal{L}_{color} + \mathcal{L}_{adv}$$

$\mathcal{L}_{struct}$ is the mean squared error of the output and target images with a weight of $\lambda_1 = 5$. $\mathcal{L}_{color}$ is the Hellinger distance between a generated image's color histogram H_g and a target

illustration's color histogram H_t defined as:

$$\frac{1}{\sqrt{2}}\|H_g^{1/2} - H_t^{1/2}\|_2$$

where $\|\cdot\|_2$ is the Euclidean norm. $\mathcal{L}_{adv}$ is an adversarial loss term defined as:

$$\mathcal{L}_{G_{full}} * \lambda_2 + \mathcal{L}_{G_{patch}}$$

where

$$\mathcal{L}_{G_{full}} = log(exp(-\mathcal{D}_{full}(\mathcal{G}(x,\, H_t),\, H_t)) + 1),$$

$$\mathcal{L}_{G_{patch}} = log(exp(-\mathcal{D}_{patch}(\mathcal{G}(x,\, H_t))) + 1),$$

$$and\ \lambda_2 = 5\text{e-}2$$

Here, $\mathcal{D}_{full}$ is a full image discriminator which takes a image (in this case $\mathcal{G}(x, H_t)$) and a target histogram H_t as input, and $\mathcal{D}_{patch}$ is a patch-based discriminator that only takes an image. $\mathcal{G}(x, H_t)$ is the output of the previously mentioned generator $\mathcal{G}$ given x (the combined input of a sketch, TBS map, and prior P) and H_t.

Our discriminators $\mathcal{D}_{full}$ and $\mathcal{D}_{patch}$ are trained in tandem, and their loss functions are defined as:

$$\mathcal{L}_{D_{full}} = log(exp(\mathcal{D}_{full}(y,\, H_t)) + 1) + R_1\left(\psi_{D_{full}}\right) + log(exp(\mathcal{D}_{full}(\mathcal{G}(x,\, H_t),\, H_t)) + 1)$$

$$\mathcal{L}_{D_{patch}} = log(exp(\mathcal{D}_{patch}(y)) + 1) + R_1\left(\psi_{D_{patch}}\right) + log(exp(\mathcal{D}_{patch}(\mathcal{G}(x,\, H_t))) + 1)$$

where y is a target color-filled TBS map and $R_1\left(\psi_{\mathcal{D}}\right) = \frac{\gamma}{2}E_{p_D(x)}\left[||\nabla D_\psi\left(x\right)||^2\right]$ is a regular-

ization term [37] on the weights ψ of a discriminator $\mathcal{D}$ with a γ of 10 and input x.

The formulated structure loss keeps the shape content of generations similar to the input sketch and the color loss penalizes the generator from outputting images whose color distribution strays too far away from the target image. The adversarial loss component is informed by two separate discriminator networks that push the generator output distribution toward the combined discriminators' approximation of the real target distribution.

(a) Color proposal prediction (b) Raw generative proposal output (c) Color-filled TBS of raw output

Figure 3.6: Example output from our generative color proposal network, using the same example image from Figure 3.4.

Outside of training, these generative outputs are cleaned by creating a color-filled TBS map with respect to itself. These cleaned outputs are output from the color proposal stage, and are sent to the final refinement stage for further distribution shift.

3.4 Refinement Stage

Our per-segment color proposals provide us with a useful and diverse spatial color prior, but reside in a separate data distribution from the illustration distribution that we want as our final output.

(a) Generated color proposal (b) Ground truth illustration

Figure 3.7: Differences in image domain before refinement.

To bridge this domain gap between proposal and final ground truth we introduce a final generative refinement model. As in the generative proposal stage, the model framework is a GAN paradigm where a generator $\mathcal{G}$ generates illustration approximations that minimize the function:

$$\mathcal{L}_{\mathcal{G}} = \mathcal{L}_{struct} + \mathcal{L}_{adv}$$

Here, the loss function is the same as for the generative color proposal stage except that λ_1 and $\mathcal{L}_{color}$ are removed from the loss function and H_t is no longer sent as color information to the generator and discriminators.

Outputs from this final trained generative model are outputs of the SALAC model.

CHAPTER 4
EXPERIMENTATION AND RESULTS

4.1 Dataset and Data Preparation

A baseline dataset is collected from a work-safe version of the Danbooru 2021 512x512-resolution illustration dataset [38]. Images are filtered based on metadata tags to select for illustrations that closely resemble the colorized versions of line-art character sketches (static background, one character, humanoid, non-greyscale, etc.) Duplicate images are removed and a custom centering algorithm is used to preprocess images so that their content resides in the horizontal center of the image. Further filtering is done by ranking the images by color and removing all images in the lowest 20 percent of "colorfulness" based on a color perception metric based on the HSV color space [39]. Finally, the remaining data is "color boosted" to an extent inversely proportional to its rank (higher rank number is more colorful) by boosting the saturation value of the image in the HSV color space. This ensures that, during training our colorization model, regressing to the mean in terms of output color is heavily penalized, as our discriminator will easily be able to tell that "muddier" images in terms of color are fakes on average.

To obtain the sketch training inputs for each illustration, conventions from both late-resizing and Tag2Pix are used by upscaling images to 4 times their original size with the SRU and running the XDoG algorithm on the upscaled images. Images are resized down to 512x512-resolution after extraction. To combat overfitting, XDoG sketch extraction parameters are randomly assigned values within separate ranges for each image.

```
def make_xdog(image):
    s = random.uniform(.3, .7)
    k = random.uniform(2, 5)
    g = random.uniform(.94, .96)
    return get_xdog_image(image, sigma=s, k=k, gamma=g,
                    epsilon=-0.5, phi=10 ** 9)
```

Listing 4.1: XDoG parameter settings

A final dataset of 87,215 sketch-to-image pairings are split into a set of 80,000 training pairs and 7,215 held-out pairs.

4.2 Training Settings

The PyTorch framework [40] is used to implement all models and training paradigms discussed here. All training was carried out on a single NVIDIA GTX 3090 GPU.

Due to the large amount of varied image pairs along with data augmentations at training time, there is no risk of overfitting on training samples and the full training set is iterated over when training models in the color proposal and refinement stages (as the segmentation stage trains on StyleGAN generated illustration outputs). Going forward when training networks, unless otherwise indicated, the ADAM optimizer [41] is used with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, α (learning rate) is initially set at 1e-4 and decayed exponentially with $\gamma = 0.9$, and a batch size of 4 is used.

For the segmentation stage, the two SC segmentation networks are pretrained on greyscale representations of the StyleGAN illustration outputs for 25 epochs, then fine-tuned on the sketch extractions of these illustrations for an additional 25 epochs with an α of 2e-5.

For the color proposal stage, the color category segmentation network is trained for 10

epochs, and the color proposal GAN is trained for 15 epochs with an initial α of 2e-4 with $\gamma = 1.2$. Due to memory consumption from backpropagation on the difference in color histograms, images were resized to 256x256-pixels from this point onward, and a batch size of 8 was used.

In the last stage, the refinement GAN was trained on 15 epochs with a batch size of 64.

4.3 Results and Evaluation

For evaluation of generated samples and comparison to other methods, we employ the Fréchet Inception Distance (FID) [8] metric. FID is extensively used as a evaluation metric for measuring the distance between distributions for image generation tasks. It is derived from calculating a mean and standard deviation of the deepest layer in InceptionV3 [42] for each image distribution (real and generated) as samples are passed through the network. Metrics are gathered with respect to the automatic AlacGAN method discussed previously. As work on automatic line-art colorization has stagnated, this model, as far as the authors are aware, is still considered near state-of-the-art for this particular automatic colorization task. Another baseline method is added for further comparison, which is a R1-regularized GAN model with the same framework as the SALAC model's generative color proposal and refinement stage (a generator playing a GAN game versus a full image discriminator and a patch discriminator).

One might recall that ground truth histograms were a conditional for the generative color proposal during training, but at inference we do not have this ground truth information, and to supply it from the user would make the process semi-automatic. Instead, color histograms are sampled from a histogram distribution (in this case the training set distribution) during inference.

Model	FID ↓
AlacGAN (automatic)	49.05
SALAC (generative only)	48.32
Full SALAC model	**27.64**

Table 4.1: Quantitative comparison of Fréchet Inception Distance scores between different models (smaller is better). Results are calculated from results on all training set sketches.

As seen in Table 4.1, the SALAC model approach performs much better for automatic image colorization than previous methods, almost halving the FID score of the baseline SALAC generative setup with modern GAN training techniques and the automatic form of AlacGAN.

(a) Original sketch input (b) Final SALAC output (c) Ground truth illustration

Figure 4.1: An example final output from the SALAC model.

Figure 4.2: Selected 256x256-pixel output examples from a fully trained SALAC model (left) and their corresponding ground truths (right).

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

5.1 Contributions and Significance

In this paper, we introduced the SALAC model approach for multi-stage fully-automatic colorization for character line-art sketches. This method outperforms previous approaches due to the powerful style and color priors that are learned as sequential sub-tasks. Furthermore, because the intermediate outputs of each stage are accessible not only as explicit information, but in human-viewable form, the SALAC model provides a more explainable generation paradigm. (For instance, if the segmentation model misses labeling some semantic information, this can be detected by the model user by merely observing the segmentation stage output masks.)

The SALAC model also is more accommodating than other approaches in terms of diverse color output, as an arbitrary distribution of color histograms can be given to the model at inference. For example, users may wish to imagine their own line-art in hundreds of various color styles for inspiration. Moreover, the SALAC model can even be used in a user-guided fashion, as one could derive color histograms from reference images or dense color hints to feed to the model at inference. This functionality would be trivial to implement, but is out of the scope of this thesis.

The SALAC model has multiple interacting models, each with it's own weights and architectures. Some of these can even be used independently of each other, such as the modified TBS algorithm for retrieving color-filled TBS maps, or the two semantic segmentation al-

gorithms in the segmentation stage. We release all of the models in the SALAC framework and their weights in a model mini zoo for public use.

Finally, we also make available data gathering and preprocessing scripts for obtaining and recreating the dataset for training, as this dataset was extensively filtered, geometrically preprocessed, and even cleaned by hand.

5.2 Limitations and Future Work

This framework is not without its limitations and even drawbacks. As stated, this model is not trained end-to-end, and requires multiple passes through multiple stages before outputting a final result. This introduces many possible points of failure within the model, especially for sketches that lie outside of the distribution the model was trained on. Additionally, inference times for this framework far exceed any "one-pass" generative model such as AlacGAN. Notably, the trapped-ball segmentation algorithm used here takes a considerable amount of time to complete.

There may also be ways to reduce model size or even eliminate portions of stages. For example, feeding color histograms as supplementary input to the category proposal model allow for more color prediction categories, therefore short-cutting the generative color proposal. In concert with this, there are no ablation studies present in this work, and further testing would have to be undertaken to ascertain which combination of methods and sub-tasks are necessary or useful for better performance.

Another limitation present is the lack of testing of authentic line-arts. Although it is the case that, at least historically [4], [43], the extensive augmentation performed on extracted (synthetic) sketches primes the model to generalize well to authentic user-made sketches, room should be made to do further testing on these authentic sketches.

Future work on automatic line-art colorization opens opportunities to amend the limitations expounded on here, and call for new techniques involving new models. Namely, diffusion models [44]–[47] have emerged as powerful iterative generative methods, and could be used a more powerful and stable learner for some of the tasks touched on in this work.

REFERENCES

[1] H. Kim, H. Y. Jhoo, E. Park, and S. Yoo, "Tag2pix: Line art colorization using text tag with secat and changing loss," *CoRR*, vol. abs/1908.05840, 2019. arXiv: 1908.05840.

[2] X. Liu, W. Wu, C. Li, Y. Li, and H. Wu, "Reference-guided structure-aware deep sketch colorization for cartoons," *Computational Visual Media*, vol. 8, no. 1, pp. 135–148, Mar. 2022.

[3] Z. Li, Z. Geng, Z. Kang, W. Chen, and Y. Yang, *Eliminating gradient conflict in reference-based line-art colorization*, 2022.

[4] Y. Ci, X. Ma, Z. Wang, H. Li, and Z. Luo, "User-guided deep anime line art colorization with conditional adversarial networks," *CoRR*, vol. abs/1808.03240, 2018. arXiv: 1808.03240.

[5] L. Zhang, C. Li, E. Simo-Serra, Y. Ji, T.-T. Wong, and C. Liu, "User-guided line art flat filling with split filling mechanism," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[6] Y. HATI, G. JOUET, F. ROUSSEAUX, and C. DUHART, "Paintstorch: A user-guided anime line art colorization tool with double generator conditional adversarial network," in *Proceedings of the 16th ACM SIGGRAPH European Conference on Visual Media Production*, ser. CVMP '19, London, United Kingdom: Association for Computing Machinery, 2019, ISBN: 9781450370035.

[7] Colorization and L. Zhang, "Two-stage sketch," 2018.

[8] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a nash equilibrium," *CoRR*, vol. abs/1706.08500, 2017. arXiv: 1706.08500.

[9] R. Cao, H. Mo, and C. Gao, "Line art colorization based on explicit region segmentation," *Computer Graphics Forum*, 2021.

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative adversarial networks*, 2014.

[11] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014. arXiv: 1411.1784.

[12] A. Radford, L. Metz, and S. Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks*, 2015.

[13] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016. arXiv: 1611.07004.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. arXiv: 1505.04597.

[15] M. Saito and Y. Matsui, "Illustration2vec: A semantic vector representation of illustrations," in *SIGGRAPH Asia 2015 Technical Briefs*, ser. SA '15, Kobe, Japan: Association for Computing Machinery, 2015, ISBN: 9781450339308.

[16] M. Arjovsky, S. Chintala, and L. Bottou, *Wasserstein gan*, 2017.

[17] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *CoRR*, vol. abs/1704.00028, 2017. arXiv: 1704.00028.

[18] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *CoRR*, vol. abs/1710.10196, 2017. arXiv: 1710.10196.

[19] H. Winnemöller, J. E. Kyprianidis, and S. C. Olsen, "Xdog: An extended difference-of-gaussians compendium including advanced image stylization," *Computers and Graphics*, vol. 36, no. 6, pp. 740–753, 2012, 2011 Joint Symposium on Computational Aesthetics (CAe), Non-Photorealistic Animation and Rendering (NPAR), and Sketch-Based Interfaces and Modeling (SBIM).

[20] D. Kim, D. Je, K. Lee, M. Kim, and H. Kim, "Late-resizing: A simple but effective sketch extraction strategy for improving generalization of line-art colorization," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022, pp. 965–974.

[21] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *CoRR*, vol. abs/1501.00092, 2015. arXiv: 1501.00092.

[22] S.-H. Zhang, T. Chen, Y.-F. Zhang, S.-M. Hu, and R. R. Martin, "Vectorizing cartoon animations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 4, pp. 618–629, 2009.

[23] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *CoRR*, vol. abs/1812.04948, 2018. arXiv: 1812.04948.

[24] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," *CoRR*, vol. abs/1912.04958, 2019. arXiv: 1912.04958.

[25] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," *CoRR*, vol. abs/2006.06676, 2020. arXiv: 2006.06676.

[26] T. Karras, M. Aittala, S. Laine, *et al.*, "Alias-free generative adversarial networks," *CoRR*, vol. abs/2106.12423, 2021. arXiv: 2106.12423.

[27] D. Pakhomov, S. Hira, N. Wagle, K. E. Green, and N. Navab, "Segmentation in style: Unsupervised semantic image segmentation with stylegan and CLIP," *CoRR*, vol. abs/2107.12518, 2021. arXiv: 2107.12518.

[28] W. Mokrzycki and M. Tatol, "Color difference delta e - a survey," *Machine Graphics and Vision*, vol. 20, pp. 383–411, Apr. 2011.

[29] G. Sharma, W. Wu, and E. N. Dalal, "The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations," *Color Research & Application*, vol. 30, no. 1, pp. 21–30, 2005. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/col.20070.

[30] H. Chang, O. Fried, Y. Liu, S. DiVerdi, and A. Finkelstein, "Palette-based photo recoloring," *ACM Trans. Graph.*, vol. 34, no. 4, Jul. 2015.

[31] M. Afifi, M. A. Brubaker, and M. S. Brown, "Histogan: Controlling colors of gan-generated and real images via color histograms," *CoRR*, vol. abs/2011.11731, 2020. arXiv: 2011.11731.

[32] J. T. Barron, "Convolutional color constancy," *CoRR*, vol. abs/1507.00410, 2015. arXiv: 1507.00410.

[33] M. Afifi and M. S. Brown, "Sensor-independent illumination estimation for DNN models," *CoRR*, vol. abs/1912.06888, 2019. arXiv: 1912.06888.

[34] P. HAT, *Linefiller*.

[35] G. Lin, A. Milan, C. Shen, and I. D. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," *CoRR*, vol. abs/1611.06612, 2016. arXiv: 1611.06612.

[36] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," *CoRR*, vol. abs/2006.09011, 2020. arXiv: 2006.09011.

[37] L. M. Mescheder, "On the convergence properties of GAN training," *CoRR*, vol. abs/1801.04406, 2018. arXiv: 1801.04406.

[38] Anonymous, D. community, and G. Branwen, *Danbooru2020: A large-scale crowdsourced and tagged anime illustration dataset*, https://gwern.net/Danbooru2020, dataset, Accessed: DATE, Jan. 2021.

[39] D. Hasler and S. Suesstrunk, "Measuring colourfulness in natural images," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5007, pp. 87–95, Jun. 2003.

[40] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[41] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014.

[42] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. arXiv: 1512.00567.

[43] Y. Liu, Z. Qin, Z. Luo, and H. Wang, "Auto-painter: Cartoon image generation from sketch by using conditional generative adversarial networks," *CoRR*, vol. abs/1705.01908, 2017. arXiv: 1705.01908.

[44] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *CoRR*, vol. abs/1907.05600, 2019. arXiv: 1907.05600.

[45] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *CoRR*, vol. abs/2006.11239, 2020. arXiv: 2006.11239.

[46] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *CoRR*, vol. abs/2010.02502, 2020. arXiv: 2010.02502.

[47] C. Saharia, W. Chan, H. Chang, *et al.*, "Palette: Image-to-image diffusion models," *CoRR*, vol. abs/2111.05826, 2021. arXiv: 2111.05826.

[48] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

APPENDIX A

OUTPUT COMPARISONS

Figure A.1: Selected 256x256-pixel output examples (left) and their corresponding ground truths from AlacGAN.

Figure A.2: Selected 256x256-pixel output examples (left) and their corresponding ground truths from SALAC (generative only).

Figure A.3: By generating with varying color histograms, we can get an array of different color outputs for one sketch.

Figure A.4: Variety of colorizations for one sketch achieved through random sampling of color histograms. Created through dimensionality reduction with t-SNE [48].

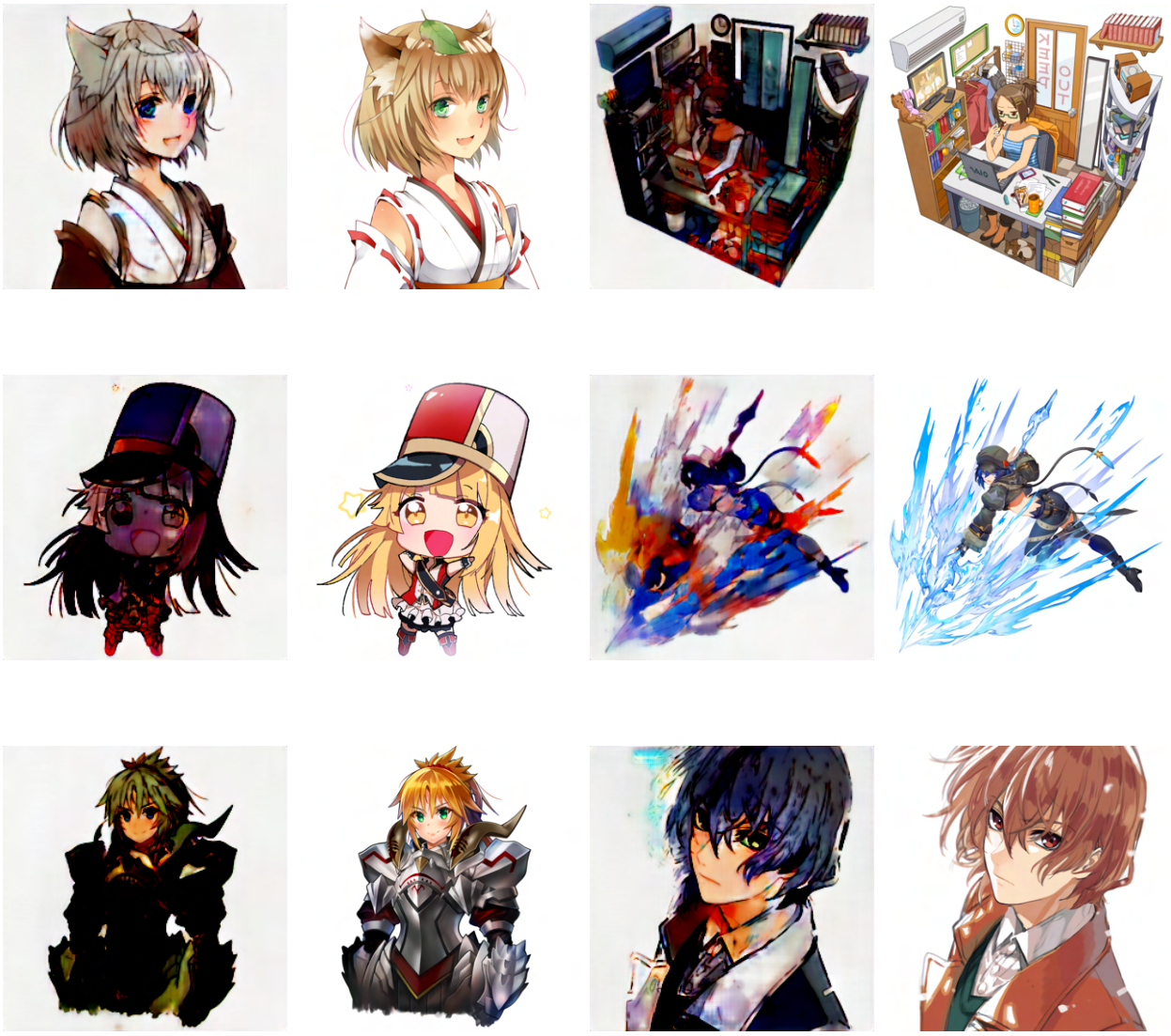


Figure A.5: Examples of failure modes (left) with corresponding ground truths (right):

Coloring over semantically different objects (top left)

Out-of-domain input sketches (top right)

Overly-saturated input color histograms (middle left)

SALAC not able to recognize semantic content (middle right)

Black color proposals overpowering sketch details (bottom left)

Visible artifacts in output (bottom right)