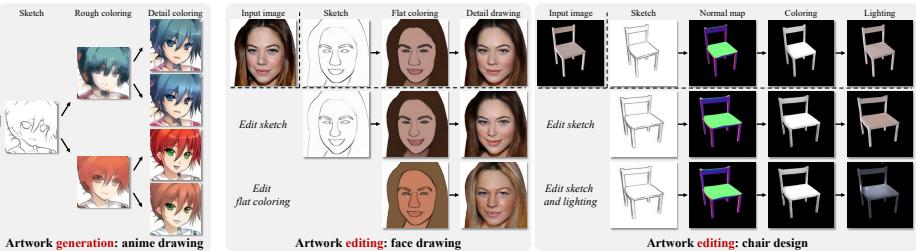


# Modeling Artistic Workflows for Image Generation and Editing

Hung-Yu Tseng<sup>\*1</sup>, Matthew Fisher<sup>2</sup>, Jingwan Lu<sup>2</sup>, Yijun Li<sup>2</sup>, Vladimir Kim<sup>2</sup>,  
Ming-Hsuan Yang<sup>1</sup>

<sup>1</sup>University of California, Merced      <sup>2</sup>Adobe Research



**Fig. 1.** We model the sequential creation stages for a given artistic workflow by learning from examples. At test time, our framework can guide the user to create new artwork by sampling different variations at each stage (left), and infer the creation stages of existing artwork to enable the user to perform natural edits by exploring variations at different stages (middle and right).

**Abstract.** People often create art by following an artistic workflow involving multiple stages that inform the overall design. If an artist wishes to modify an earlier decision, significant work may be required to propagate this new decision forward to the final artwork. Motivated by the above observations, we propose a generative model that follows a given artistic workflow, enabling both multi-stage image generation as well as multi-stage image editing of an existing piece of art. Furthermore, for the editing scenario, we introduce an optimization process along with learning-based regularization to ensure the edited image produced by the model closely aligns with the originally provided image. Qualitative and quantitative results on three different artistic datasets demonstrate the effectiveness of the proposed framework on both image generation and editing tasks.

## 1 Introduction

Creating artwork from scratch is a herculean task for people without years of artistic experience. For novices to the world of art, it would be more feasible to

---

\* Work done during HY's internship at Adobe Research.

accomplish this task if there are clear creation steps to follow. Take a watercolor painting for example. One may be guided to first sketch the outline with pencils, then fill out areas with large brushes, and finalize details such as the color gradient and shadow with small brushes. At each stage, some aspects (i.e., variations) of the overall design are determined to carry forward to the final piece of art.

Inspired by these observations, we aim to model workflows for creating art, targeting two relevant artistic applications: multi-stage artwork creation and multi-stage artwork editing. As shown in Figure 1, multi-stage artwork generation guides the user through the creation process by starting from the first stage then selecting the variation at each subsequent creation stage. In the multi-stage artwork editing, we are given a final piece of artwork and infer all the intermediate creation stages, enabling the user to perform different types of editing on various stages and propagate them forward to modify the final artwork.

Existing artwork creation approaches use conditional generative adversarial networks (conditional GANs) [23, 57, 33] to produce the artwork according to user-provided input signals. These methods can take user inputs such as a sketch image [10] or segmentation mask [40, 51] and perform a single-step generation to synthesize the final artwork. To make the creation process more tractable, recent frameworks adopt a multi-step generation strategy to accomplish the generation tasks such as fashion simulation [46] and sketch-to-image [15]. However, these approaches typically do not support editing existing artwork. To manipulate an existing artwork image without degrading the quality, numerous editing schemes [6, 37, 43, 54, 55] have been proposed in the past decade. Nevertheless, these methods either are designed for specific applications [37, 43, 54] or lack flexible controls over the editing procedure because of the single-stage generation strategy [6, 55].

In this paper, we develop a conditional GAN-based framework that 1) synthesizes novel artwork via multiple creation stages, and 2) edits existing artwork at various creation stages. Our approach consists of an artwork generation module and a workflow inference module. The artwork generation module learns to emulate each artistic stage by a series of multi-modal (i.e., one-to-many) conditional GAN [57] networks. Each network in the artwork generation module uses a stage-specific latent representation to encode the variation presented at the corresponding creation stage. At test time, the user can determine the latent representation at each stage sequentially for the artwork generation module to synthesize the desired artwork image.

To enable editing existing artwork, we also design an inference module that learns to sequentially infer the corresponding images at all intermediate stages. We assume a one-to-one mapping from the final to intermediate stages, and use a series of uni-modal conditional GANs [23] to perform this inference. At test time, we predict the stage-specific latent representations from the inferred images at all intermediate stages. Depending on the desired type of edit, the user can edit any stage to manipulate the stage-specific image or latent representation and regenerate the final artwork from the manipulated representations.

We observe that directly applying our workflow inference module can cause the reconstructed image to differ slightly from the initially provided artwork. Such a reconstruction problem is undesirable since the user expects the generated image to be unchanged when no edits are performed. To address this problem, we design an optimization procedure along with learning-based regularization to refine the reconstructed image. This optimization aims to minimize the appearance difference between the reconstructed and the original artwork image, while the learning-based regularization seeks to guide the optimization process and alleviate overfitting.

We collect three datasets with different creation stages to demonstrate the use cases of our approach: face drawing, anime drawing, and chair design. We demonstrate the creation process guided by the proposed framework and present editing results made by artists. For quantitative evaluations, we measure the reconstruction error and Fréchet inception distance (FID) [17] to validate the effectiveness of the proposed optimization and learning-based regularization scheme. We make the code and datasets public available to stimulate the future research.<sup>1</sup>

In this work, we make the following three contributions:

- We propose an image generation and editing framework which models the creation workflow for a particular type of artwork.
- We design an optimization process and a learning-based regularization function for the reconstruction problem in the editing scenario.
- We collect three different datasets containing various design stages and use them to evaluate the proposed approach.

## 2 Related Work

**Generative adversarial networks (GANs).** GANs [4,7,16,25,26] model the real image distribution via adversarial learning schemes. Typically, these methods encode the distribution of real images into a latent space by learning the mapping from latent representations to generated images. To make the latent representation more interpretable, the InfoGAN [11] approach learns to disentangle the latent representations by maximizing the mutual information. Similar to the FineGAN [45] and VON [58] methods, our approach learns to synthesize an image via multiple stages of generation, and encode different types of variation into separate latent spaces at various stages. Our framework extends these approaches to also enable image editing of different types of artwork.

**Conditional GANs.** Conditional GANs learn to synthesize the output image by referencing the input context such as text descriptions [52], scene graphs [50], segmentation masks [18,40], and images [23]. According to the type of mapping from the input context to the output image, conditional GANs can be categorized as uni-modal (one-to-one) [23,56] or multi-modal (one-to-many) [20,34,38,57]. Since we assume there are many possible variations involved for the generation at each stage of the artwork creation workflow, we use the multi-modal conditional

---

<sup>1</sup> <https://github.com/hytseng0509/ArtEditing>

GANs to synthesize the next-stage image, and utilize the uni-modal conditional GANs to inference the prior-stage image.

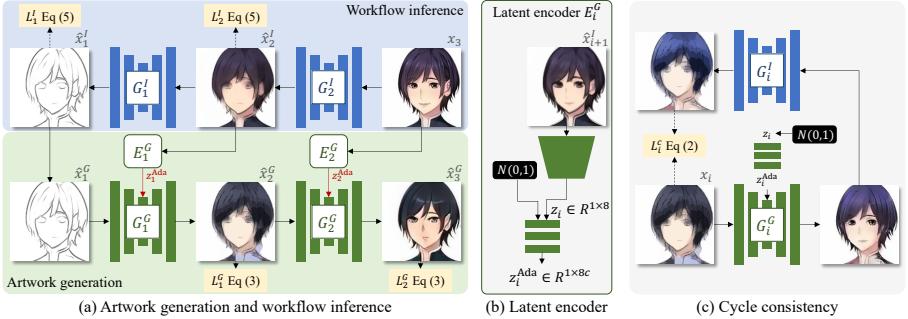
**Image editing.** Image editing frameworks enable user-guided manipulation without degrading the realism of the edited images. Recently, deep-learning-based approaches have made significant progress on various image editing tasks such as colorization [22,30,53,54], image stylization [19,37], image blending [21], image inpainting [39,42], layout editing [35], and face editing [9,12,43]. Unlike these task-specific methods, the task-agnostic iGAN [55] and GANPaint [6] models map the variation in the training data onto a low-dimensional latent space using GAN models. Editing can be conducted by manipulating the representation in the learned latent space. Different from iGAN and GANPaint, we develop a multi-stage generation method to model different types of variation at various stages.

**Optimization for reconstruction.** In order to embed an existing image to the latent space learned by a GAN model, numerous approaches [13,29,57] propose to train an encoder to learn the mapping from images to latent representations. However, the generator sometimes fails to reconstruct the original image from the embedded representations. To address this problem, optimization-based methods are proposed in recent studies. Abdal et al. [1] and Bau et al. [6] adopt the gradient descent scheme to optimize the latent representations and modulations for the feature activations, respectively. The goal is to minimize the appearance distance between the generated and original images. We also utilize the optimization strategy to reconstruct existing artwork images. In addition, we introduce a learning-based regularization function to guide the optimization process.

**Regularizations for deep learning.** These approaches [14,28,31,47,48,49] aim to prevent the learning function from overfitting to a specific solution. Particularly, the weight decay scheme [28] regularizes by constraining the magnitude of learning parameters during the training phase. Nevertheless, regularization methods typically involve hyper-parameters that require meticulous hand-tuning to ensure the effectiveness. The MetaReg [5] method designs a learning-to-learn algorithm to automatically find the hyper-parameters of the weight decay regularization to address the domain generalization problem. Our proposed learning-based regularization is trained with a similar strategy but different objectives to alleviate the overfitting problem described in Section 3.2.

### 3 Method

Our approach is motivated by the sequential creation stages of artistic workflows. We build a model that enables a user to 1) follow the creation stages to generate novel artwork and 2) conduct edits at different stages. Our framework is composed of an artwork generation and a workflow inference module. As shown in Figure 2(a), the artwork generation module learns to model the creation stages of the artist workflow. To enable editing an existing piece of art, the workflow inference module is trained to sequentially infer the corresponding images at all creation stages. When editing existing artwork, it is important that



**Fig. 2. Overview of the proposed framework.** (a) Given  $N$  creation stages ( $N = 3$  in this example), our approach consists of  $N - 1$  workflow inference networks and  $N - 1$  artwork generation networks. The workflow inference module produces the intermediate results of the input artwork at all creation stages. The artwork generation module computes the latent representation  $z$  and transformation parameter  $z^{\text{Ada}}$  for each stage, then reconstructs the input artwork images from these transformation parameters. (b) The latent encoder  $E_i^G$  extracts the stage-specific latent representation  $z$  from the example, and computes the transformation parameters  $z^{\text{Ada}}$  for the AdaIN normalization layers ( $c$  channels). (c) We introduce a cycle consistency loss for each stage to prevent the artwork generation model (which accounts for detail coloring in this example) from memorizing the variation determined at the previous stages (sketching and flat coloring).

the artwork remains as close as possible to the original artwork, and only desired design decisions are altered. To enable this, we design an optimization process together with a learning-based regularization that allows faithful reconstruction of the input image. We provide the implementation and training details for each component in the proposed framework as supplemental material.

### 3.1 Artwork Generation and Workflow Inference

**Preliminaries.** The proposed approach is driven by the number of stages in the training dataset and operates in a supervised setting with aligned training data. Denoting  $N$  as the number of stages, the training dataset is comprised of a set of image groups  $\{(x_1, x_2, \dots, x_N)\}$ , where  $x_N$  denotes the artwork image at the final stage. We construct the proposed framework with  $N - 1$  workflow inference models  $\{G_i^I\}_{i=1}^N$  as well as  $N - 1$  artwork generation models  $\{(E_i^G, G_i^G)\}_{i=1}^N$ . We show an example of 3 stages in Figure 2(a). Since the proposed method is based on the observation that artists sequentially determine a design factor (i.e., variation) at each stage, we assume that the generation from the image in the prior stage to the later one is multi-modal (i.e., one-to-many mapping), while the inference from the final to the previous stages is uni-modal (i.e., one-to-one mapping).

**Artwork generation.** The artwork generation module aims to mimic the sequential creation stages of the artistic workflow. Since we assume the generation

from the prior stages to the following ones is multi-modal, we construct a series of artwork generation networks by adopting the multi-modal conditional GAN approach in BicycleGAN [57] and the network architecture of MUNIT [20]. As shown in Figure 2 (a) and (b), each artwork generation model contains two components: latent encoder  $E_i^G$  and generator  $G_i^G$ . The latent encoder  $E_i^G$  encodes the variation presented at the  $i$ -th stage in a stage-specific latent space. Given an input image  $x_i$  and the corresponding next-stage image  $x_{i+1}$ , the latent encoder  $E_i^G$  extracts the stage-specific latent representation  $z_i$  from the image  $x_{i+1}$ , and computes the transformation parameter  $z_i^{\text{Ada}}$ . The generator  $G_i^G$  then takes the current-stage image  $x_i$  as input and modulates the activations through the AdaIN normalization layers [57] with the transformation parameter  $z_i^{\text{Ada}}$  to synthesize the next-stage image  $\hat{x}_{i+1}^G$ , namely

$$\hat{x}_{i+1}^G = G_i^G(x_i, E_i^G(x_{i+1})) \quad i \in \{1, 2, \dots, N-1\}. \quad (1)$$

We utilize the objective introduced in the BicycleGAN [57], denoted as  $L_i^{\text{bicycle}}$ , for training the generation model. The objective  $L_i^{\text{bicycle}}$  is detailed in the supplementary material.

Ideally, the artwork generation networks corresponding to a given stage would encode only new information (i.e., incremental variation), preserving prior design decisions from earlier stages. To encourage this property, we impose a cycle consistency loss to enforce the generation network to encode the variation presented at the current stage only, as shown in Figure 2(c). Specifically, we use the inference model  $G_i^I$  to map the generated next-stage image back to the current stage. The mapped image should be identical to the original image  $x_i$  at the current stage, namely

$$L_i^c = \|G_i^I(G_i^G(x_i, E_i^G(z_i))) - x_i\|_1 \quad z_i \sim N(0, 1). \quad (2)$$

Therefore, the overall training objective for the artwork generation model at the  $i$ -th stage is

$$L_i^G = L_i^{\text{bicycle}} + \lambda^c L_i^c, \quad (3)$$

where  $\lambda^c$  controls the importance of the cycle consistency.

**Workflow inference.** To enable the user to edit the input artwork  $x_N$  at different creation stages, our inference module aims to hallucinate the corresponding images at all previous stages. For the  $i$ -th stage, we use a unimodal conditional GAN network [23] to generate the image at  $i$ -th stage from the image at  $(i+1)$ -th stage, namely

$$\hat{x}_i^I = G_i^I(x_{i+1}) \quad i \in \{1, 2, \dots, N-1\}. \quad (4)$$

During the training phase, we apply the hinge version of GAN loss [7] to ensure the realism of the generated image  $\hat{x}_i^I$ . We also impose an  $\ell_1$  loss between the synthesized image  $\hat{x}_i^I$  and the ground-truth image  $x_i$  to stabilize and accelerate the training. Hence the training objective for the inference network at the  $i$ -th stage is

$$L_i^I = L_i^{\text{GAN}}(\hat{x}_i^I) + \lambda^1 \|\hat{x}_i^I - x_i\|_1, \quad (5)$$



**Fig. 3. Motivation of the AdaIN optimization and learning-based regularization.** The proposed AdaIN optimization and the learning-based regularization are motivated by the observations that 1) using the computed transformation parameters  $z^{Ada}$  in Figure 2 cannot well reconstruct the original input image (red outline in 1-st row)), and 2) the AdaIN optimization may degrade the quality of the editing results (yellow outline in 2-nd row).

where  $\lambda^1$  controls the importance of the  $\ell_1$  loss.

**Test-time inference.** As shown in Figure 2(a), given an input artwork image  $x_N$ , we sequentially obtain the images at all previous stages  $\{\hat{x}_i^I\}_{i=1}^N$  using the workflow inference module (blue block). We then use the artwork generation module (green block) to extract the latent representations  $\{z_i\}_{i=1}^{N-1}$  from the inferred images  $\{\hat{x}_i^I\}_{i=1}^N$ , and compute the transformation parameters  $\{z_i^{Ada}\}_{i=1}^{N-1}$ . Combining the first-stage image  $x_1^G = x_1^I$  and the transformation parameters  $\{z_i^{Ada}\}_{i=1}^{N-1}$ , the generation module consecutively generates the images  $\{\hat{x}_i^G\}_{i=2}^N$  at the following stages. The user can choose the stage to manipulate based on the type of edit desired. Edits at the  $i$ -th stage can be performed by either manipulating the latent representation  $z_i$  or directly modifying the image  $x_i^G$ . For example, in Figure 2(a), the user can choose to augment the representation  $z_1$  to adjust the flat coloring. After editing, the generation module generates the new artwork image at the final stage.

### 3.2 Optimization for Reconstruction

As illustrated in Section 3.1, the artwork generation module would ideally reconstruct the input artwork image (i.e.,  $\hat{x}_N^G = x_N$ ) from the transformation parameters  $\{z_i^{Ada}\}_{i=1}^{N-1}$  before the user performs an edit. However, the reconstructed image  $\hat{x}_N^G$  may be slightly different from the input image  $x_N$ , as shown in the first row of Figure 3. Therefore, we adopt an AdaIN optimization algorithm to optimize the transformation parameters  $\{z_i^{Ada}\}_{i=1}^N$  of the AdaIN normalization layers in the artwork generation models. The goal of the AdaIN optimization is to minimize the appearance distance between the reconstructed and input image.

While this does improve the reconstruction of the input image, we observe that the optimization procedure causes the generation module to memorize input image details, which degrades the quality of some edited results, as shown in the

second row of Figure 3. To mitigate this memorization, we propose a learning-based regularization to improve the AdaIN optimization.

**AdaIN optimization.** The AdaIN optimization approach aims to minimize the appearance distance between the reconstructed image  $\hat{x}_N^G$  and the input artwork image  $x_N$ . There are many choices for what to optimize to improve reconstruction: we could optimize the parameters in the generation models or the extracted representations  $\{z_i\}_{i=1}^N$ . Optimizing model parameters is inefficient because of the large number of parameters to be updated. On the other hand, we find that optimizing the extracted representation is ineffective, as validated in Section 4.3. As a result, we choose to optimize the transformation parameters  $\{z_i^{\text{Ada}}\}_{i=1}^N$  of the AdaIN normalization layers in the generation models, namely the AdaIN optimization. Note that a recent study [1] also adopts a similar strategy.

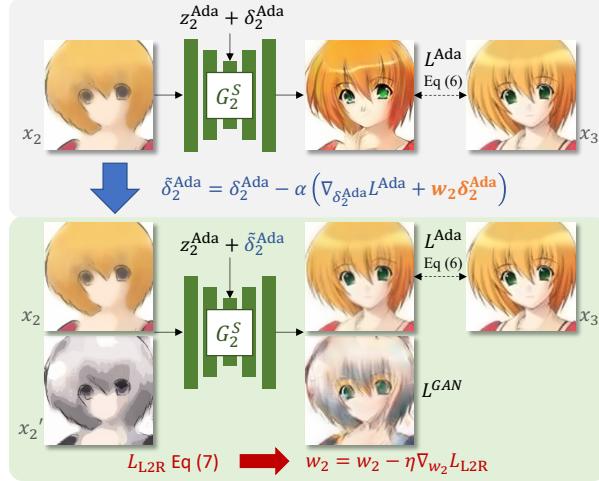
We conduct the AdaIN optimization for each stage sequentially. The transformation parameter at the early stage is optimized and then fixed for the optimization at the later stages. Except for the last stage (i.e.,  $i = N - 1$ ) that uses the input artwork image  $x_N$ , the inferred image  $x_{i+1}^I$  by the inference model serves as the reference image  $x^{\text{ref}}$  for the optimization. For each stage, we first use the latent encoder  $E_i^G$  to compute the transformation parameter  $z_i^{\text{Ada}}$  from the reference image for generating the image. Since there are four AdaIN normalization layers with  $c$  channels in each artwork generation model, the dimension of the transformation parameter is  $1 \times 8c$  (a scale and a bias term for each channel). Then we follow the standard gradient descent procedure to optimize the transformation parameters with the goal of minimizing the loss function  $L^{\text{Ada}}$  which measures the appearance distance between the synthesized image  $\hat{x}_i^G$  by the generator  $G_i^G$  and the reference image  $x^{\text{ref}}$ . The loss function  $L^{\text{Ada}}$  is a combination of the pixel-wise  $\ell_1$  loss and VGG-16 perceptual loss [24], namely

$$L^{\text{Ada}}(\hat{x}_i^G, x^{\text{ref}}) = \|\hat{x}_i^G - x^{\text{ref}}\|_1 + \lambda^p L^p(\hat{x}_i^G, x^{\text{ref}}), \quad (6)$$

where  $\lambda_p$  is the importance term. We summarize the AdaIN optimization in Algorithm 1. Note that in practice, we optimize the incremental term  $\delta_i^{\text{Ada}}$  for the transformation parameter  $z_i^{\text{Ada}}$ , instead of updating the parameter itself.

**Learning-based regularization.** Although the AdaIN optimization scheme addresses the reconstruction problem, it often degrades the quality of editing operations, as shown in the second row of Figure 3. This is because the AdaIN optimization causes overfitting (memorization of the reference image  $x^{\text{ref}}$ ). The incremental term  $\delta_i^{\text{Ada}}$  for the transformation parameter  $z_i^{\text{Ada}}$  is updated to extreme values to achieve better reconstruction, so the generator becomes sensitive to the change (i.e., editing) on the input image and produces unrealistic results.

To address the overfitting problem, we use weight decay regularization [28] to constrain the magnitude of the incremental term  $\delta_i^{\text{Ada}}$ , as shown in Line 6 in Algorithm 1. However, it is difficult to find a general hyper-parameter setting  $w_i \in R^{1 \times 8c}$  for different generation stages of various artistic workflows. Therefore, we propose a learning algorithm to optimize the hyper-parameter  $w_i$ . The core idea is that updating the incremental term  $\delta_i^{\text{Ada}}$  with the regularization  $w_i \delta_i^{\text{Ada}}$  should 1) improve the reconstruction and 2) maintain the realism of ed-



**Fig. 4. Training process for learning-based regularization.** For the  $i$ -th stage ( $i = 2$  in this example), we optimize the hyper-parameter  $w_i$  for the weight decay regularization (orange text) by involving the AdaIN optimization in the training process: after the incremental term  $\delta_i^{\text{Ada}}$  is updated via one step of AdaIN optimization and the weight decay regularization (blue arrow), the generation model should achieve improved reconstruction as well as maintain the quality of the editing result (green block). Therefore, we use the losses  $L^{\text{Ada}}, L^{\text{GAN}}$  computed from the updated parameter  $\tilde{\delta}_i^{\text{Ada}}$  to optimize the hyper-parameter  $w_i$  (red arrow).

its on an input image. We illustrate the proposed algorithm in Figure 4. In each iteration of training at the  $i$ -th stage, we sample an image pair  $(x_i, x_{i+1})$  and an additional input image  $x'_i$  from the training dataset. The image  $x'_i$  serves as the edited image of  $x_i$ . We first use the latent encoder  $E_i^G$  to extract the transformation parameter  $z_i^{\text{Ada}}$  from the next-stage image  $x_{i+1}$ . As shown in the grey block of Figure 4, we then update the incremental term from  $\delta_i^{\text{Ada}}$  to  $\tilde{\delta}_i^{\text{Ada}}$  via one step of the AdaIN optimization and the weight decay regularization. With the updated incremental term  $\tilde{\delta}_i^{\text{Ada}}$ , we use the loss function  $L^{\text{Ada}}$  to measure the reconstruction quality, and use the GAN loss to evaluate the realism of editing results, namely

$$L^{\text{L2R}} = L^{\text{Ada}}(G_i^G(x_i, z_i^{\text{Ada}} + \tilde{\delta}_i^{\text{Ada}}), x_{i+1}) + \lambda^{\text{GAN}} L^{\text{GAN}}(G_i^G(x'_i, z_i^{\text{Ada}} + \tilde{\delta}_i^{\text{Ada}})). \quad (7)$$

Finally, since the loss  $L^{\text{L2R}}$  indicates the efficacy of the weight decay regularization, we optimize the hyper-parameter  $w_i$  by

$$w_i = w_i - \eta \nabla_{w_i} L^{\text{L2R}}, \quad (8)$$

where  $\eta$  is the learning rate of the training algorithm for the proposed learning-based regularization.

**Algorithm 1:** AdaIN optimization at  $i$ -th stage

---

```

1 Require: reference image  $x^{\text{ref}} = x_N$  or  $x^{\text{ref}} = \hat{x}_{i+1}^I$ , input image  $\hat{x}_i^G$ , learning
    rate  $\alpha$ , iterations  $T$ , regularization parameter  $w_i$ 
2  $z_i^{\text{Ada}} = E_i^G(x^{\text{ref}})$ ,  $\delta_i^{\text{Ada}} = \mathbf{0} \in R^{1 \times 8c}$ 
3 while  $t = \{1, \dots, T\}$  do
4    $\hat{x}_{i+1}^G = G_i^G(\hat{x}_i^G, z_i^{\text{Ada}} + \delta_i^{\text{Ada}})$ 
5    $L^{\text{Ada}} = \|\hat{x}_{i+1}^G - x^{\text{ref}}\|_1 + \lambda^p L^p(\hat{x}_{i+1}^G, x^{\text{ref}})$ 
6    $\delta_i^{\text{Ada}} = \delta_i^{\text{Ada}} - \alpha \left( \nabla_{\delta_i^{\text{Ada}}} L_{\text{Ada}} + w_i \delta_i^{\text{Ada}} \right)$ 
7 end
8 Return:  $z_i^{\text{Ada}} + \delta_i^{\text{Ada}}$ 

```

---

**Table 1. Summarization of the datasets.** Three datasets are processed for evaluating the proposed framework.

Dataset	Face drawing	Anime drawing	Chair design
Source	CelebaHQ [32]	EdgeConnect [39]	ShapeNet [8]
# Training images	29000	33323	12546
# Testing images	1000	1000	1000
Stages	1. sketch 2. flat coloring 3. detail drawing	1. sketch 2. rough coloring 3. detail coloring	1. sketch 2. normal map 3. coloring 4. lighting

## 4 Experimental Results

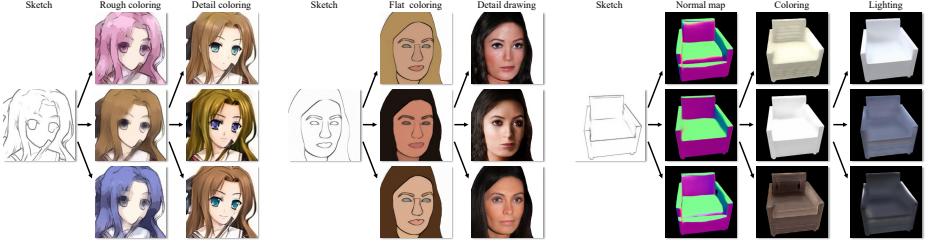
### 4.1 Datasets

To evaluate our framework, we manually process face drawing, anime drawing, and chair design datasets. Table 1 summarizes the generation stages, the number of training images, the number of testing images, and the source of the images for each dataset. We describe more details in the supplementary material.

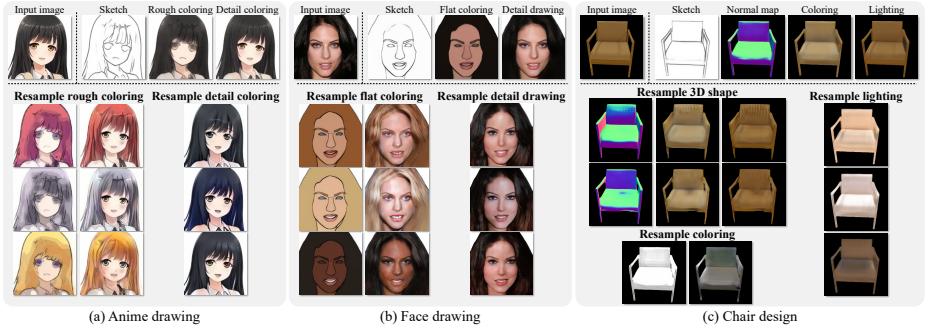
### 4.2 Qualitative Evaluation

**Generation.** We present the generation results at all stages in Figure 5. In this experiment, we use the testing images at the first stage as inputs, and randomly sample various latent representation  $z \in \{z_i\}_{i=1}^{N-1}$  at each stage of the proposed artwork generation module. The generation module sequentially synthesizes the final result via multiple stages. It successfully generates variations by sampling different random latent codes at different stages. For example, when generating anime drawings, manipulating the latent code at the final stage produces detailed color variations, such as modifying the saturation or adding the highlights to the hair regions.

**Editing.** Figure 6 shows the results of editing the artwork images at different stages. Specifically, after the AdaIN optimization reconstructs the testing image at the final stage (first row), we re-sample the representations  $z \in \{z_i\}_{i=1}^{N-1}$  at



**Fig. 5. Results of image generation from the first stage.** We use the first-stage testing images as input and randomly sample the latent representations to generate the image at the final stage.

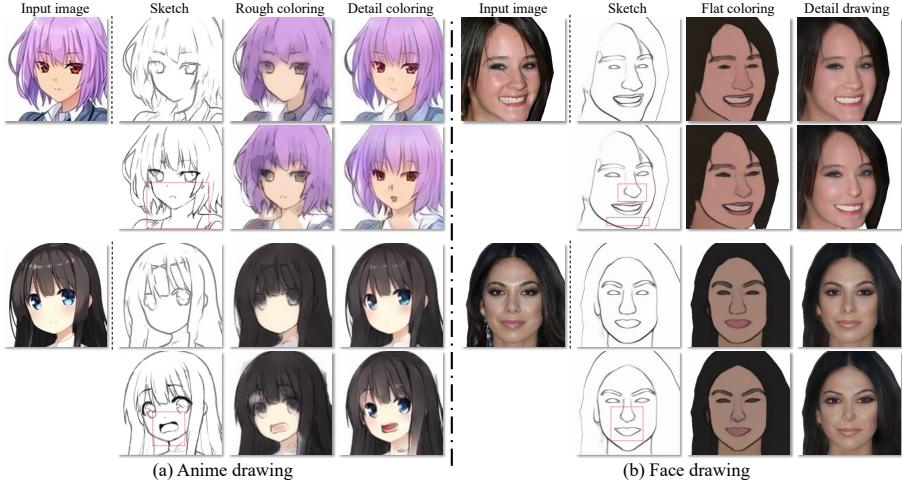


**Fig. 6. Re-sampling latent representation at each stage.** After we use the AdaIN optimization process to reconstruct the input image (1st row), we edit the reconstructed image by re-sampling the latent representations at various stages.

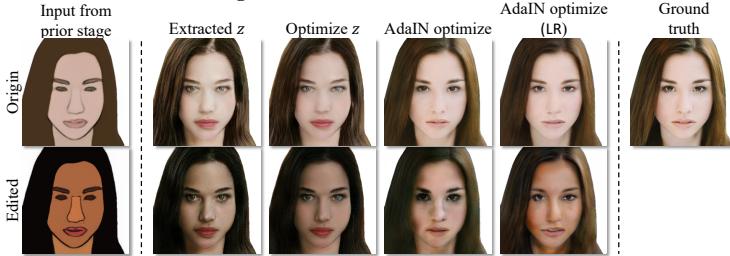
various stages. Our framework is capable of synthesizing the final artwork such that its appearance only changes with respect to the stage with re-sampled latent code. For example, for editing face drawings, re-sampling representations at the flat coloring stage only affects hair color, while maintaining the haircut style and details.

To evaluate the interactivity of our system, we also asked professional artists to edit some example sketches (Figure 7). First, we use the proposed framework to infer the initial sketch from the input artwork image. Given the artwork image and the corresponding sketch, we asked an artist to modify the sketch manually. For the edited sketch (second row), we highlight the edits with the red outlines. This experiment confirms that the proposed framework enables the artists to adjust only some stages of the workflow, controlling only desired aspects of the final synthesized image. Additional artistic edits are shown in Figure 1.

**AdaIN optimization and learning-based regularization.** Figure 8 presents the results of the AdaIN optimization and the proposed learning-based regularization. As shown in the first row, optimizing representations  $z$  fails to refine the reconstructed images due to the limited capacity of the low-dimensional latent representation. In contrast, the AdaIN optimization scheme minimizes the perceptual difference between the input and reconstructed images. We also



**Fig. 7. Results of artistic editing.** Given an input artwork image, we ask the artist to edit the inferred sketch image. The synthesis model then produces the corresponding edited artwork. The first row shows the input artwork and inferred images, and the red outlines indicate the edited regions.



**Fig. 8. Results of different optimization approaches.** We show both the reconstruction and editing results of various optimization approaches at the final stage for the face drawing dataset.

demonstrate how the optimization process influences the editing results in the second row. Although the AdaIN optimization resolves the reconstruction problem, it leads to overfitting and results in unrealistic editing results synthesized by the generation model. By utilizing the proposed learning-based regularization, we address the overfitting problem and improve the quality of the edited images.

### 4.3 Quantitative Evaluation

**Evaluation metrics.** We use the following metrics in the quantitative evaluation.

- Reconstruction error: Given the input artwork  $x_N$  and the reconstructed image  $\hat{x}_N^G$ , we use the  $\ell_1$  distance  $\|\hat{x}_N^G - x_N\|$  to evaluate the reconstruction quality.

**Table 2. Quantitative results of reconstruction.** We use the  $\ell_1$  pixel-wise distance ( $\downarrow$ ) and the FID ( $\downarrow$ ) score to evaluate the reconstruction ability.  $w$  and LR indicates the hyper-parameter for the weight regularization and applying the learned regularization, respectively.

Optimization $w$		Face		Anime		Chair	
		$\ell_1$	FID	$\ell_1$	FID	$\ell_1$	FID
None	-	0.094	39.78	0.127	36.73	0.074	129.2
$z$	0	0.104	40.70	0.126	45.66	0.068	107.0
AdaIN	0	<b>0.040</b>	<b>34.61</b>	<b>0.042</b>	<b>26.56</b>	<b>0.009</b>	<b>46.48</b>
AdaIN	$10^{-3}$	0.043	35.78	0.056	29.14	0.019	53.08
AdaIN	$10^{-2}$	0.053	39.19	0.097	46.31	0.049	83.58
AdaIN	LR	0.045	33.28	0.070	34.16	0.018	49.44

- FID: We use the Fréchet Inception Distance (FID) [17] score to measure the realism of generated images  $\hat{x}_N^G$ . A smaller FID score indicates better visual quality.

**Reconstruction.** As shown in Section 3.2, we conduct the AdaIN optimization for each stage sequentially to reconstruct the testing image at the final stage. We use both the reconstruction error and FID score to evaluate several baseline methods and the AdaIN optimization, and show the results in Table 2. Results on the 2-nd and 3-rd rows demonstrate that the AdaIN optimization is more effective than optimizing the latent representations  $\{z_i\}_{i=1}^{N-1}$ . On the other hand, applying stronger weight decay regularization (i.e.,  $w_i = 10^{-2}$ ) diminishes the reconstruction ability of the AdaIN optimization. By applying the weight decay regularization with learned hyper-parameter  $w$  (i.e., LR), we achieve comparable reconstruction performance in comparison to the optimization without regularization.

**Editing.** In this experiment, we investigate how various optimization methods influence the quality of edited images. For each testing final-stage image, we first use different optimization approaches to refine the reconstructed images. We then conduct the editing by re-sampling the latent representation  $z_i$  at a randomly chosen stage. We adopt the FID score to measure the quality of the edited images and show the results in Table 3. As described in Section 3.2, applying the AdaIN optimization causes overfitting that degrades the quality of the edited images. For instance, applying the AdaIN optimization increases the FID score from 38.68 to 44.28 on the face drawing dataset. One straightforward solution to alleviate this issue is to apply strong weight decay regularizations (i.e.,  $w = 10^{-2}$ ). However, according to the results in 5-th row of Table 2, such strong regularizations reduce the reconstruction effectiveness of the AdaIN optimization. Combining the results in Table 2 and Table 3, we conclude that applying the regularization with the learned hyper-parameter  $w$  not only mitigates overfitting but also maintains the efficacy of the AdaIN optimization. We conduct more analysis of the proposed learning-based regularization in the supplementary materials.

**Table 3. Quantitative results of editing.** We use the FID ( $\downarrow$ ) score to evaluate the quality of the edited images  $\hat{x}_N^G$  synthesized by the proposed framework.  $w$  and LR indicates the hyper-parameter for the weight regularization and applying the learned regularization, respectively.

Optimization	$w$	Face	Anime	Chair
None	-	$38.68 \pm 0.44$	<b><math>35.59 \pm 0.12</math></b>	$128.4 \pm 1.50$
AdaIN	0	$44.28 \pm 0.45$	$37.40 \pm 0.36$	$97.90 \pm 1.20$
AdaIN	$10^{-3}$	$41.75 \pm 0.49$	$38.95 \pm 0.59$	<b><math>91.68 \pm 4.23</math></b>
AdaIN	$10^{-2}$	<b><math>38.57 \pm 0.94</math></b>	$38.07 \pm 0.54$	$99.36 \pm 7.23$
AdaIN	LR	$39.40 \pm 0.21$	$35.73 \pm 0.26$	$95.25 \pm 0.73$

#### 4.4 Limitations

The proposed framework has several limitations (see supplemental material for visual examples). First, since the model learns the multi-stage generation from a training dataset, it fails to produce appealing results if the style of the input image is significantly different from images in the training set. Second, the unimodal inference assumption may not be correct. In practice, the mapping from later stages to previous ones can also be multi-modal. For instance, the style of the pencil sketches by various artists may be different. Finally, artists may not follow a well-staged workflow to create artwork in practice. However, our main goal is to provide an example workflow to make the artwork creation and editing more feasible, especially for the users who may not be experts in that type of artwork.

## 5 Conclusions

In this work, we introduce an image generation and editing framework that models the creation stages of an artistic workflow. We also propose a learning-based regularization for the AdaIN optimization to address the reconstruction problem for enabling non-destructive artwork editing. Qualitative results on three different datasets show that the proposed framework 1) generates appealing artwork images via multiple creation stages and 2) synthesizes the editing results made by the artists. Furthermore, the quantitative results validate the effectiveness of the AdaIN optimization and the learning-based regularization.

We believe there are many exciting areas for future research in this direction that could make creating high-quality artwork both more accessible and faster. We would like to study video sequences of artists as they create artwork to automatically learn meaningful workflow stages that better align with the artistic process. This could further enable the design of editing tools that more closely align with the operations artists currently perform to iterate on their designs.

## Acknowledgements

This work is supported in part by the NSF CAREER Grant #1149783.

## References

1. Abdal, R., Qin, Y., Wonka, P.: Image2stylegan: How to embed images into the stylegan latent space? In: ICCV (2019) [4](#), [8](#)
2. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süstrunk, S.: Slic superpixels compared to state-of-the-art superpixel methods. TPAMI **34**(11), 2274–2282 (2012) [19](#)
3. Adobe: Adobe dimension. <https://www.adobe.com/products/dimension.html> (2019) [20](#)
4. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. In: ICML (2017) [3](#)
5. Balaji, Y., Sankaranarayanan, S., Chellappa, R.: Metareg: Towards domain generalization using meta-regularization. In: NIPS (2018) [4](#)
6. Bau, D., Strobelt, H., Peebles, W., Wulff, J., Zhou, B., Zhu, J.Y., Torralba, A.: Semantic photo manipulation with a generative image prior. ACM TOG (Proc. SIGGRAPH) **38**(4), 59 (2019) [2](#), [4](#)
7. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. In: ICLR (2019) [3](#), [6](#), [18](#)
8. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015) [10](#), [20](#)
9. Chang, H., Lu, J., Yu, F., Finkelstein, A.: Pairedcyclegan: Asymmetric style transfer for applying and removing makeup. In: CVPR (2018) [4](#)
10. Chen, W., Hays, J.: Sketchygan: Towards diverse and realistic sketch to image synthesis. In: CVPR (2018) [2](#)
11. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: NIPS (2016) [3](#)
12. Cheng, Y.C., Lee, H.Y., Sun, M., Yang, M.H.: Controllable image synthesis via segvae. In: ECCV (2020) [4](#)
13. Donahue, J., Simonyan, K.: Large scale adversarial representation learning. In: NIPS (2019) [4](#)
14. Ghiasi, G., Lin, T.Y., Le, Q.V.: Dropblock: A regularization method for convolutional networks. In: NIPS (2018) [4](#)
15. Ghosh, A., Zhang, R., Dokania, P.K., Wang, O., Efros, A.A., Torr, P.H., Shechtman, E.: Interactive sketch & fill: Multiclass sketch-to-image translation. In: CVPR (2019) [2](#)
16. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS (2014) [3](#)
17. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: NIPS (2017) [3](#), [13](#), [19](#), [20](#)
18. Huang, H.P., Tseng, H.Y., Lee, H.Y., Huang, J.B.: Semantic view synthesis. In: ECCV (2020) [3](#)
19. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: ICCV (2017) [4](#), [18](#)
20. Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation. In: ECCV (2018) [3](#), [6](#), [18](#)
21. Hung, W.C., Zhang, J., Shen, X., Lin, Z., Lee, J.Y., Yang, M.H.: Learning to blend photos. In: ECCV (2018) [4](#)

22. Iizuka, S., Simo-Serra, E., Ishikawa, H.: 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 (2016) [4](#)
23. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR (2017) [2](#), [3](#), [6](#)
24. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV (2016) [8](#)
25. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. In: ICLR (2018) [3](#)
26. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: CVPR (2019) [3](#)
27. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015) [18](#)
28. Krogh, A., Hertz, J.A.: A simple weight decay can improve generalization. In: NIPS (1992) [4](#), [8](#)
29. Larsen, A.B.L., Sønderby, S.K., Larochelle, H., Winther, O.: Autoencoding beyond pixels using a learned similarity metric. arXiv preprint arXiv:1512.09300 (2015) [4](#)
30. Larsson, G., Maire, M., Shakhnarovich, G.: Learning representations for automatic colorization. In: ECCV (2016) [4](#)
31. Larsson, G., Maire, M., Shakhnarovich, G.: Fractalnet: Ultra-deep neural networks without residuals. In: ICML (2017) [4](#)
32. Lee, C.H., Liu, Z., Wu, L., Luo, P.: Maskgan: Towards diverse and interactive facial image manipulation. In: CVPR (2020) [10](#), [19](#)
33. Lee, H.Y., Tseng, H.Y., Huang, J.B., Singh, M.K., Yang, M.H.: Diverse image-to-image translation via disentangled representations. In: ECCV (2018) [2](#)
34. Lee, H.Y., Tseng, H.Y., Mao, Q., Huang, J.B., Lu, Y.D., Singh, M., Yang, M.H.: Drit++: Diverse image-to-image translation via disentangled representations. IJCV pp. 1–16 (2020) [3](#)
35. Lee, H.Y., Yang, W., Jiang, L., Le, M., Essa, I., Gong, H., Yang, M.H.: Neural design network: Graphic layout generation with constraints. In: ECCV (2020) [4](#)
36. Li, Y., Fang, C., Hertzmann, A., Shechtman, E., Yang, M.H.: Im2pencil: Controllable pencil illustration from photographs. In: CVPR (2019) [19](#), [20](#)
37. Li, Y., Liu, M.Y., Li, X., Yang, M.H., Kautz, J.: A closed-form solution to photo-realistic image stylization. In: ECCV (2018) [2](#), [4](#)
38. Mao, Q., Lee, H.Y., Tseng, H.Y., Ma, S., Yang, M.H.: Mode seeking generative adversarial networks for diverse image synthesis. In: CVPR (2019) [3](#)
39. Nazeri, K., Ng, E., Joseph, T., Qureshi, F., Ebrahimi, M.: Edgeconnect: Generative image inpainting with adversarial edge learning. arXiv preprint arXiv:1901.00212 (2019) [4](#), [10](#), [19](#)
40. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: CVPR (2019) [2](#), [3](#)
41. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: NIPS workshop (2017) [18](#)
42. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: CVPR (2016) [4](#)
43. Portenier, T., Hu, Q., Szabo, A., Bigdeli, S.A., Favaro, P., Zwicker, M.: Faceshop: Deep sketch-based face image editing. ACM TOG (Proc. SIGGRAPH) **37**(4), 99 (2018) [2](#), [4](#)
44. Sangkloy, P., Lu, J., Fang, C., Yu, F., Hays, J.: Scribbler: Controlling deep image synthesis with sketch and color. In: CVPR (2017) [21](#)

45. Singh, K.K., Ojha, U., Lee, Y.J.: Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In: CVPR (2019) [3](#)
46. Song, S., Zhang, W., Liu, J., Mei, T.: Unsupervised person image generation with semantic parsing transformation. In: CVPR (2019) [2](#)
47. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. JMLR **15**(1), 1929–1958 (2014) [4](#)
48. Tseng, H.Y., Chen, Y.W., Tsai, Y.H., Liu, S., Lin, Y.Y., Yang, M.H.: Regularizing meta-learning via gradient dropout. arXiv preprint arXiv:2004.05859 (2020) [4](#)
49. Tseng, H.Y., Lee, H.Y., Huang, J.B., Yang, M.H.: Cross-domain few-shot classification via learned feature-wise transformation. In: ICLR (2020) [4](#)
50. Tseng, H.Y., Lee, H.Y., Jiang, L., Yang, W., Yang, M.H.: Retrievegan: Image synthesis via differentiable patch retrieval. In: ECCV (2020) [3](#)
51. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional gans. In: CVPR (2018) [2](#)
52. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D.N.: Stackgan++: Realistic image synthesis with stacked generative adversarial networks. TPAMI **41**(8), 1947–1962 (2018) [3](#)
53. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV (2016) [4](#)
54. Zhang, R., Zhu, J.Y., Isola, P., Geng, X., Lin, A.S., Yu, T., Efros, A.A.: Real-time user-guided image colorization with learned deep priors. ACM TOG (Proc. SIGGRAPH) **9**(4) (2017) [2](#), [4](#)
55. Zhu, J.Y., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: ECCV (2016) [2](#), [4](#)
56. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV (2017) [3](#)
57. Zhu, J.Y., Zhang, R., Pathak, D., Darrell, T., Efros, A.A., Wang, O., Shechtman, E.: Toward multimodal image-to-image translation. In: NIPS (2017) [2](#), [3](#), [4](#), [6](#), [18](#)
58. Zhu, J.Y., Zhang, Z., Zhang, C., Wu, J., Torralba, A., Tenenbaum, J., Freeman, B.: Visual object networks: image generation with disentangled 3d representations. In: NIPS (2018) [3](#)

## A Appendix

In this supplementary material, we first present the implementation details for each component of the proposed framework. Second, we complement the experiment details. Third, we visualize the learning-based regularization. Fourth, we show visual examples illustrating the failure cases of the proposed method. Finally, we present more qualitative results to complement the paper.

### A.1 Implementation Details

We implement our framework with PyTorch [41]. The details for each component are described as follows.

**Workflow inference.** The hyper-parameter  $\lambda^1$  in Equation 5 of the paper is assigned to be 10. We use the Adam optimizer [27] with the learning rate of  $2 \times 10^{-4}$  and batch size of 8 for optimizing the model. We first train each network separately with 450,000 iterations, then jointly train all the networks in the workflow inference module with 450,000 iterations.

**Artwork generation.** We set the hyper-parameter  $\lambda^c$  in Equation 3 of the paper to be 1. Similar to the training for the workflow inference module, we use the Adam optimizer [27] with the learning rate of  $2 \times 10^{-4}$  and batch size of 8. We train each network separately with 1,200,000 iterations, then jointly train all the networks in the artwork generation module with 600,000 iterations. We adopt the objectives in the BicycleGAN [57] approach for training the artwork generation module, as described in Equation 3 in the paper. More specifically, the loss  $L_i^{\text{bicycle}}$  in Equation 3 is formulated as

$$L_i^{\text{bicycle}} = L_i^{\text{GAN}} + \lambda^1 L^1 + \lambda^{\text{latent}} L^{\text{latent}} + \lambda^{\text{KL}} L^{\text{KL}}, \quad (9)$$

where  $L_i^{\text{GAN}}$  is the hinge version of GAN loss [7],  $L^1$  is the  $\ell_1$  loss between the generated and ground-truth images,  $L^{\text{latent}}$  is the latent regression loss between the predicted and input latent representations, and  $L^{\text{KL}}$  is the KL divergence loss on the latent representations. Following the setting in the BicycleGAN scheme, we respectively assign the hyper-parameters  $\lambda^1$ ,  $\lambda^{\text{latent}}$ , and  $\lambda^{\text{KL}}$  to be 10, 0.5, and 0.01. We use the network architecture proposed in the MUNIT [20] framework (involving AdaIN normalization layers [19]) rather than the U-Net structure in the BicycleGAN framework.

**AdaIN optimization.** In the editing scenario during the testing phase, we conduct the AdaIN optimization from the first to the last stages sequentially to refine the reconstructed image. For each stage, we set the hyper-parameters  $\lambda^p$ ,  $\alpha$ ,  $T$  in Algorithm 1 in the paper to be 10, 0.1 and 150, respectively.

**Learning-based regularization.** We summarize the training of the proposed learning-based regularization in Figure 4 of the paper and Algorithm 2. The regularization function is trained separately for each creation stage. We respectively set the hyper-parameters  $\eta$ ,  $T^{\text{reg}}$ , and  $\lambda^{\text{GAN}}$  to be  $10^{-3}$ , 40000, and 1. We use the Adam optimizer [27] and the batch size of 1 for the training.

---

**Algorithm 2:** Training overview of the learning-based regularization at  $i$ -th stage
 

---

```

1 Require: pre-trained generation model  $\{E_i^G, G_i^G\}$ , learning rate  $\eta$ , iterations
    $T^{\text{reg}}$ , importance factor  $\lambda^{\text{GAN}}$ 
2  $w_i = \mathbf{0.001} \in R^{1 \times 8c}$ 
3 while  $t = \{1, \dots, T^{\text{reg}}\}$  do
4   Sample  $(x_i, x_{i+1})$  and  $x'_i$  from the dataset
5    $z_i^{\text{Ada}} = E_i^G(x_i)$ ,  $\delta_i^{\text{Ada}} = \mathbf{0} \in R^{1 \times 8c}$ 
6   // Get reconstructed image before the AdaIN optimization
7    $\hat{x}_{i+1}^G = G_i^G(x_i, z_i^{\text{Ada}} + \delta_i^{\text{Ada}})$ 
8   // Optimize incremental term with the regularization function
      // (AdaIN optimization)
9    $\tilde{\delta}_i^{\text{Ada}} = \delta_i^{\text{Ada}} - \alpha \left( \nabla_{\delta_i^{\text{Ada}}} L_{\text{Ada}}(\hat{x}_{i+1}^G, x_{i+1}) + w_i \delta_i^{\text{Ada}} \right)$ 
10  // Get the reconstructed image and editing results after the
      // optimization
11   $\tilde{x}_{i+1}^G = G_i^G(x_i, z_i^{\text{Ada}} + \tilde{\delta}_i^{\text{Ada}})$ 
12   $\tilde{x}'_{i+1}^G = G_i^G(x'_i, z_i^{\text{Ada}} + \tilde{\delta}_i^{\text{Ada}})$ 
13  // Update the regularization function based on the reconstruction
      // and editing results after the optimization
14   $L^{\text{L2R}} = L^{\text{Ada}}(\tilde{x}_{i+1}^G, x_{i+1}) + \lambda^{\text{GAN}} L^{\text{GAN}}(\tilde{x}'_{i+1}^G)$ 
15   $w_i = w_i - \eta \nabla_{w_i} L^{\text{L2R}}$ 
16 end
17 Return:  $w_i$ 
  
```

---

## A.2 Experiment Details

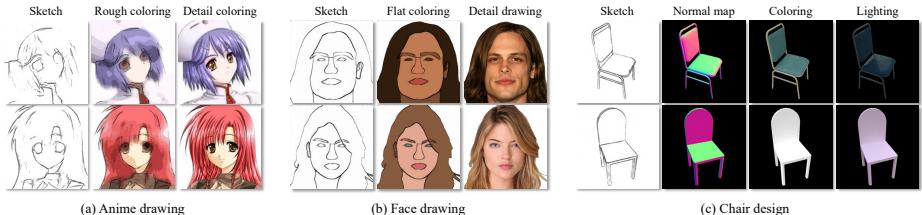
We illustrate how we process each dataset for evaluating the proposed framework. Example training images in each dataset are shown in Figure 9. In addition, we also describe how we compute FID [17] score.

**Face drawing dataset.** We collect the photo-realistic face images from the CelebAMask-HQ dataset [32]. We prepare three design stages for the face drawing dataset: sketch, flat coloring, and detail drawing. We use the ground-truth attribute segmentation mask to remove the background of the cropped RGB images in the CelebAMask-HQ dataset as the final-stage images. For the flat coloring, we assign pixels with the median color computed from the corresponding region according to the ground-truth attribute segmentation mask. Finally, we use the pencil sketch [36] model to generate simple sketch images from the flat coloring images.

**Anime drawing dataset.** We construct the dataset from the anime images in the EdgeConnect [39] dataset. Three stages are used in this dataset: sketch, rough coloring, detail coloring. For rough coloring, we first apply the SLIC [2]

**Table 4. FID scores of real images.** We show the FID ( $\downarrow$ ) scores of the real images in the test set to supplement the results in Table 2 and Table 3 of the paper.

Datasets	Face Anime Chair		
Real images	12.8	16.5	25.3



**Fig. 9. Training examples in each dataset.** For each dataset, we show the example training images at each creation stage.

super-pixel approach to cluster the pixels in each anime image. For each cluster, We then compute the median color and assign to the pixels in that cluster. Finally, we adopt the median filter to smooth the rough coloring images. As for the sketch, we use the pencil sketch [36] scheme to extract the sketch image from the original anime image.

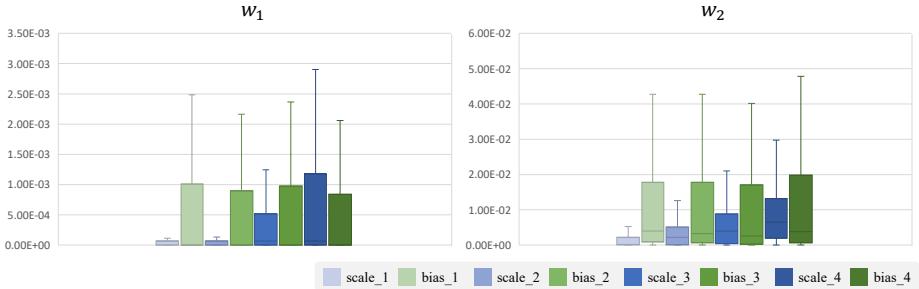
**Chair design.** We render the chair models in the ShapeNet dataset [8] via the photo-realistic renderer [3] for building the dataset. There are four stages presented in this dataset: sketch, normal map, coloring, and lighting. We sample two different camera viewpoints for each chair model. For each viewpoint, we randomly sample from 300 spherical environment maps of diverse indoor and outdoor scenes to render the last-stage image. For the coloring image, we use a default white lighting environment for the rendering. We configure the rendering tools to produce the corresponding depth map for each viewpoint and infer the normal map image from the depth map. Finally, we extract the sketch image from the normal map image using the pencil sketch model [36].

**FID scores.** We use the official implementation to compute the FID [17] scores.<sup>2</sup> For all experiments, we use the generated images from the whole test set as well as the real images in the training set. Since we need to re-sample the latent representations for the editing experiments presented in Table 3 in the paper, we conduct 5 trials for each experiment and report the average results. We show the FID scores of real images in the test set in Table 4. The scores reported in this table can be considered as the lower-bound scores for each task.

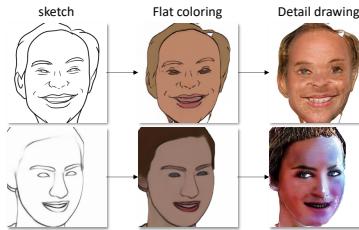
### A.3 Additional Experimental Results

**Visualizing learning-based regularization.** To understand our learning-based regularization function, we visualize the learned hyper-parameter  $w_i$  of

<sup>2</sup> <https://github.com/bioinf-jku/TTUR>



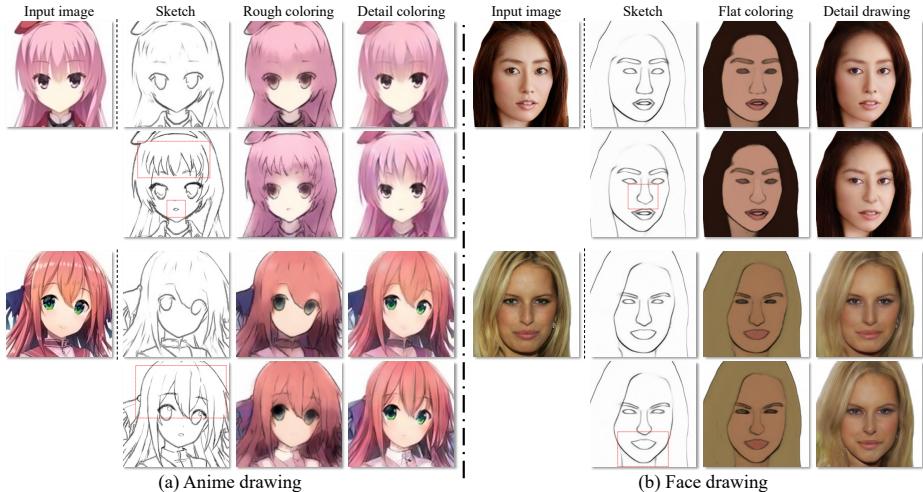
**Fig. 10. Visualization of the proposed learning-based regularization.** We show the quartile visualization of the hyper-parameter  $w_i$  for our learning-based regularization approach trained on the face drawing dataset. The learned function tends to have stronger regularization on the bias terms.



**Fig. 11. Failure cases.** Our framework fails to generate appealing results (*top*) if the style of the input sketch image is significantly different from that of training images, and (*bottom*) if we use extreme latent representations which are out of the prior distributions we sampled from during the training phase.

the weight decay regularization described in Section 3.3 in the paper and Algorithm 2. The value of the hyper-parameter indicates the strength of the regularization for the AdaIN optimization process. Figure 10 shows the visualization of the hyper-parameters trained on the face drawing dataset. In general, The regularization on the bias terms is stronger than that on the scaling terms. Since the goal is to minimize the appearance distance between the reconstructed and original input image, the AdaIN optimization tends to complement such discrepancy with the bias terms. However, as shown in Figure 8 in the paper, such optimization may lead the bias terms to extreme values and make the generation model sensitive to the change (i.e., editing) of the input image. The proposed learning-based regularization mitigates the problem by applying stronger regularization on the bias terms, thus encourage the optimization process to modify the scaling terms. Quantitative results shown in Section 4.3 in the paper validate that the proposed learning-based regularization improves the quality of the editing results.

**Failure cases.** We observe several failure cases of the proposed framework, which are presented in Figure 11. First, if the style of the input image is significantly different from the training data, the artwork generation module fails to produce appealing results. Similar to the Scribbler [44] approach, we argue



**Fig. 12. Results of artistic editing.** Given an input artwork image, we ask the artist to edit the inferred sketch image. The synthesis model then produces the corresponding edited artwork. The first row shows the input artwork and inferred images, and the red outlines indicate the edited regions.

that such a problem may be alleviated by diversifying the style of the training images. Second, during the creation process, the generation module synthesizes results with artifacts if we use extreme latent representations that are out of the prior distributions we sampled from during the training phase.

**Qualitative results.** We show more editing results conducted by the artists in Figure 12.