User-Guided Line Art Colourisation with Conditional GAN

Xuancheng Lin
Concord College
004386@concordcollege.org.uk

Abstract

Line art colourisation is a challenging task in computer vision, as it requires colours, textures, and shadings to be derived from abstract lines. Unlike image colourisation, which heavily relies on texture information, line art may not convey textures. In this paper, we propose a semi-automatic learning-based framework for colourising line art based on user-controlled colour hints. Our framework uses a conditional Generative Adversarial Network (cGAN) [38] with a U-net generator [45] and a multi-scale PatchGAN discriminator [23, 51]. To encourage more consistent and higher quality colourisation results, we incorporate the 'Concatenation and Spatial Attention' [57] and 'ConvNeXt' [34] modules throughout the framework. We train our approach solely on synthetic line arts and demonstrate its strong robustness with real-world line art, producing high-quality colourised results. Our proposed framework provides an effective solution for colourising line art and has potential applications in various areas, such as comic book production and animation.

https:github.com/cytrusl/colourisation

1. Introduction

Colourisation plays a fundamental role in the process of art creation. However, creating an ideal colour composition with fine texture and shading details is immensely time-consuming and involves a large portion of redundant work. Automating the entire process, therefore, would be beneficial to the community. With the system, not only are artists allowed to freely explore their colouring ideas to the fullest, but also iteratively polish their illustrations and designs. This also applies to the Anime industry, where illustrators no longer have to participate in the repetitive colourisation of every in-between frame. Though progress has been made in developing such automation, it is still difficult to reconcile automation and the concept of limitless creativity.

Recent studies in user-guided line art colourisation are mainly based on Generative Adversarial networks (GANs) [17, 46]. Earlier frameworks, such as Petalica Paint [43] and Comicolorization [15], suffer from colour artefacts, including watercolour blurring, colour bleeding and colour distortion. To address these issues, more recent frameworks, such as Style2Paint V3 [60] and Yuan and Simo-Serra [57] have introduced new techniques such as a two-stage colouring approach and spatial attention mechanisms. However, there are still issues that remain unsolved, including low-quality results in complicated regions, monotonous art style, and inconsistent colour.

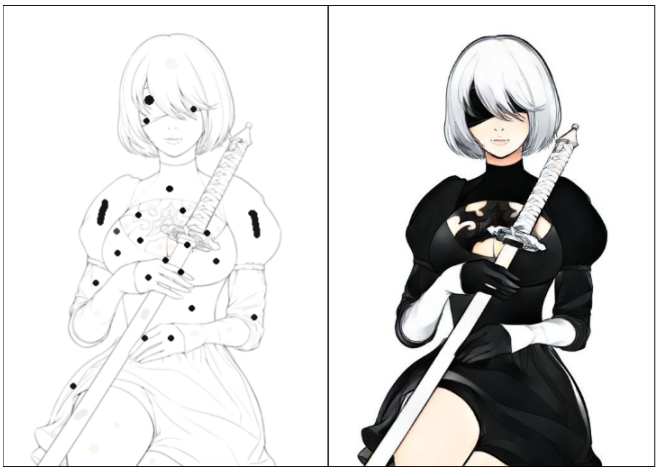

Figure 1. Our proposed framework colourised a line art created by an artist based on guided colour scribbles and learned priors. The line art image is from our collected line art dataset.

In this paper, we propose a deep learning framework capable of generating plausible colourised illustrations from given line arts and colour scribble hints. Our framework is based on the conditional Generative adversarial network (cGAN) [38], consisting of a generator network and a discriminator network; both are Convolutional neural networks (CNNs). Following the majority approaches, our generator has a U-Net [45] architecture, with skip connections between the encoder and decoder to maintain low-level features, such as the line art outline. In addition, we introduce a feature extraction network that directly provides additional spatial features of the line art to the generator aiding the computation of the attention map. We also incorporated modules from the state-of-the-art CNN, ConvNeXt [34], to further enhance the performance of our networks. Inspired by Yuan and Simo-Serra [57], we fuse the

min-max losses with a perceptual loss [24], a feature matching loss [46], and an L1 loss as our criterion in the GAN training phase.

The absence of public illustration/line art pairs significantly increased the difficulty of training a robust high-performance framework to authentic line art styles. Our method is to extract line arts from the illustrations. Figure 6 demonstrates the distinctions between synthetic and authentic line arts. To minimise the framework from overfitting to synthetic line arts, we utilised multiple line art extraction procedures and several data augmentation strategies. These techniques enable us to train our framework with more robustness and compatibility with diversified real-world line arts.

To create a convenient environment for users to experiment with our framework, we developed a GUI software that enables users to colourise line art by scribbling directly on the line art with the desired colour, as shown in Figure 9. Compared to the Petalica Paint [43] family (Tanpopo, Satsuki, and Canna), our model can better follow the user hints and hence provides more precise user control.

By training with our proposed networks, objective function, and dataset in an end-to-end manner, our framework is capable of generating illustration-realistic colourisation from sparse, authentic line art boundaries and colour scribbles. Our contributions are summarised as follows:

- We propose an illustration synthesis framework for user-guided line art colourisation that can produce quality colourised samples.
- Using multiple augmentation strategies and line extraction algorithms, our framework demonstrated strong robustness to authentic line arts.
- We collect a high-quality illustrations dataset for training and a line arts dataset for testing.
- We develop a GUI application allowing users to colourise line art easily.

2. Related Work

Generative Adversarial Networks. Recent studies of GANs [17] have found a great success in a variety of image synthesis tasks, including high-resolution image synthesis [25], photo-realistic super-resolution [55, 52], image in-painting [42], and image-to-image translation [23, 51]. The success of GANs stems from the ability of two competing networks, namely the generator and discriminator, to learn from each other iteratively. The generator network produces synthetic images, while the discriminator network attempts to distinguish them from real images. GANs can be further improved by adding additional information to the generator and discriminator networks, known as conditional GANs (cGANs) [38]. The additional information can be in the form of class labels, matching input images, segmentation maps, or image captions. This approach has been shown to improve the quality of the generated samples significantly. However, it is challenging to train a model that can handle different patterns between the training and testing data, such as synthetic line art and authentic line art. This requires a robust and generalized framework that can handle diverse real-world inputs.

User-guided Colourisation. Early interactive image colourisation frameworks [21, 29] propagate user-specified colours onto images using low-level similarity schemes. These techniques are rigid when dealing with intricate images. Later research extended these methods by using chrominance blending [56], specific schemes for different textures [44], better similarity schemes [37], and global optimisation with all-pair constraints [1, 54]. Learning-based methods such as boosting [31], manifold learning [7], and neural networks [12, 47] has also been proposed to propagate stroke colours with learned prior. Apart from scribble-based methods, some approaches are proposed to colourise images by transferring the reference image's colour theme [15, 19, 30] or colour palette [61, 6]. However, all these methods strongly rely on the additional texture information that exists in the grey-scale images but is unavailable for line arts.

Recent research on user-guided line art colourisation relies heavily on neural network algorithms, with GANs being one of the most prevalent. Early approaches such as Frans [14] and Auto-Painter [32] can generate sensibly coloured results. Style2Paint V1 [58, 28] can colourise synthetic line art based on the colour scheme of the reference image. Petalica Paint [43] developed the first online tool capable of producing plausible results based on synthetic line art and stroke colour hints. However, these methods still suffer from colour inconsistency, colour bleeding, and lack of textures. By incorporating a local feature network, Ci et al. [8] achieved high-quality user-guided line art colouring, as well as greater robustness when encountering real-world line arts. Tag2Pix [26], Yuan and Simo-Serra [57] utilised spatial attention mechanisms and can generate quality details. Style2Paint V3 and V4 [60], which use scribble colours and reference images as user input, introduced a multi-stage training strategy with multiple generators for different purposes. In particular, Style2Paint V4 contains three generator networks, one for generating colours, one for generating gradients, and one for shadings. It is capable of producing images with realistic tones and highlights.

Despite the contributions based on GANs, diffusion-based generative models [20, 50, 39] have predominated image synthesis tasks over the past two years due to their substantially better outcomes. Furusawa et al. [16] explored line art colourisation with a diffusion-based strategy. Unlike providing user control inputs, the model can generate multiple coloured candidates for users to select. Style2Paint V5 [59], the current state-of-the-art, is based on the Diffusion models. It includes the Dorothy and Alice models. The Alice model accepts input in the form of a rough sketch, and the Dorothy model

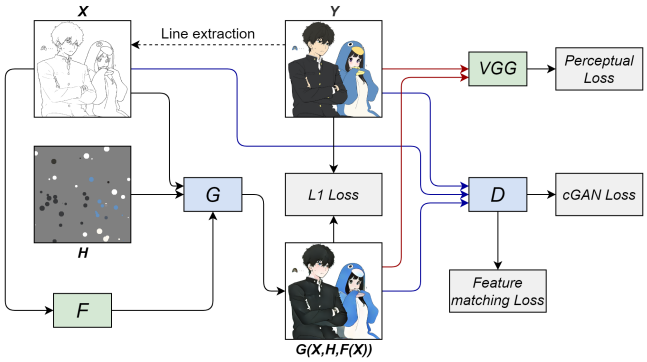

Figure 2. **The workflow of our cGAN-based framework** involves a generator $\mathcal{G}$, a discriminator $\mathcal{D}$, and two feature extractors F ($\mathcal{F}_1$) and VGG ($\mathcal{F}_2$). The arrows indicate the flow of data through the networks to the losses.

allows for digital line art. Both models are capable of generating professional-quality illustrations with highlights, gradients, and shadows on the spot, as well as finely detailed backgrounds, which no previous work could achieve. Nevertheless, the diffusion strategy is not explored in this paper due to the fact that these models are novel and require more in-depth knowledge to comprehend. In addition, the sizes of these frameworks are significantly larger compared to works in GANs, making them difficult for typical consumers to access.

3. Proposed Method

We train a deep learning model to transform line arts $\mathbf{x}$ and colour hints $\mathbf{h}$ into fully-colourised illustrations. In Section 3.1, we introduce our learning-based system. We then define our network architecture in Section 3.2. Finally, we describe the loss functions of our system in Section 3.3.

3.1. Overview workflow

The initial input to our system is a greyscale line art image $\mathbf{x} \in \mathbb{R}^{1\times H\times W}$, which is a sparse, binary image-like tensor synthesised from real illustration Y with line extraction algorithms. The second input is the simulated user hint $\mathbf{h} \in \mathbb{R}^{4\times H\times W}$. It consists of a map of colour scribbles and their matching binary mask. The colour scribble map is an RGB image-like tensor, and the binary mask consists of '1's at locations where scribbles exist and '0's where it does not exist. Both inputs are standardised and normalised with $\mu = (0.5, 0.5, 0.5)$ and $\sigma = (0.5, 0.5, 0.5)$.

Real-world line art often portrays a wide array of themes in a variety of artistic styles. Determining the border between distinct items and extracting semantic information from simple line drawings is essential. In image-to-image translation tasks, the two play a crucial role in producing high-quality outcomes. Following Yuan's and Simo-Serra's approach [57], we utilise a pre-trained model for extracting semantic feature maps from line art. These feature maps include direct semantic and spatial information that is useful for calculating the spatial attention maps in the bottleneck of the generator. Specifically, we utilise the ResNet34 network [18, 3] $\mathcal{F}_1(\mathbf{x}) \in \mathbb{R}^{512\times(H/16)\times(W/16)}$ as the feature, of which is pre-trained on the Danbooru2018 [2] dataset containing 3.33m anime illustrations (including coloured images and line arts).

The output of the system is $\mathcal{G}(\mathbf{x}, \mathbf{h}, \mathcal{F}_1(\mathbf{x}))$, which is an estimate of the RGB channels of the line art. The mapping is learned with a generator $\mathcal{G}$, parameterised by Θ.

Using the network architecture described in Section 3.2 and illustrated in Figure 3. We trained the network to minimise the objective function formulated in Equation 5.

3.2. Network Architecture

Our model is based on a Conditional Generative Adversarial Network (cGAN), consisting of a generator and a discriminator. The forward propagation of the model is illustrated in Figure 2. The overview of our generator and discriminator architectures are shown in Figure 3. The details of each module are described in Section A.

3.2.1. Generator design

Line art colourisation tasks are characterised by the mapping of a high-resolution input grid to a high-resolution output grid. For such problems, the input and output seem different on the surface but have the same structure beneath. Hence, input and output structures are aligned (e.g., line art outlines). With these considerations, our generator network's design builds upon the U-net [45] architecture with skip connections between the encoder and decoder. The skip connections preserve the low-level properties, such as the contours of the line art, that were lost during the downsampling steps. Specifically, we add skip connections between each layer i and layer $n - i$ of the network, where n is the total number of layers. Each skip connection concatenates all channels from layer i with those from layer $n - i$.

In addition to the U-Net, we incorporate a local feature network $\mathcal{F}_1$, which is connected to the bottleneck of the generator network $\mathcal{G}$. The $\mathcal{F}_1$ network consists of a pre-trained ResNet34 [18, 3], a convolution layer, and two ConvNeXt layers. The line art $\mathbf{x}$ is fed into the network to extract the feature map $\mathcal{F}_1(\mathbf{x})$. The output from the ResNet34 is upsampled to match the channel of the feature maps in the bottleneck and processed by convolution layers before being passed to the bottleneck of the generator network.

At the input of the generator network, the line art $\mathbf{x}$ and colour hint $\mathbf{h}$ are concatenated along the channel dimension and transformed into feature maps via a convolution layer. The feature maps are downsampled four times before reaching the bottleneck of the network. The bottleneck has nine Concate-

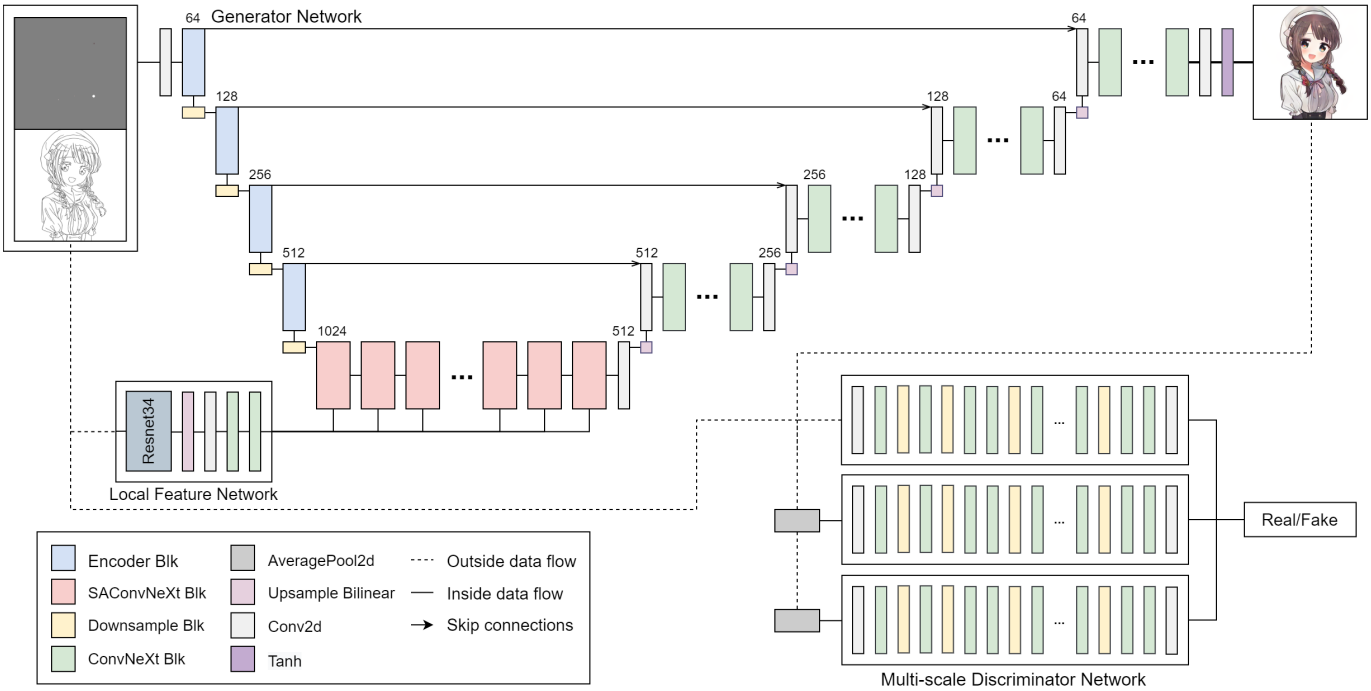

Figure 3. **Overview of the network architectures.** The Generator network processes the hint and the line art to obtain a colourised image. The output then passes through the discriminator with the target image to compute the loss. The number above certain layers indicates the number of channels of the feature maps at these layers. The detailed structures of the modules are displayed in section A.

nation and Spatial Attention [57] blocks, where the feature maps from the previous block are processed and concatenated with the extracted features $\mathcal{F}_1(\mathbf{x})$ of the line art image before computing spatial attention. The spatial attention maps are multiplied with the processed input feature maps and added to the input feature maps. Finally, the output feature maps from the bottleneck are upsampled and processed by multiple ConvNeXt [34] blocks four times to produce the final colourised illustration.

3.2.2. Discriminator design

High-resolution line art colourisation presents a challenge for GAN discriminator design, as it requires a large receptive field to differentiate between real and synthetic images. Traditional CNN discriminator architectures would require a significant number of parameters to achieve a wide receptive field. To address this, we adopt a multi-scale discriminator architecture inspired by Wang et al. [51], which consists of three identical discriminators operating at different image scales. Each discriminator has a PatchGAN [23] architecture that penalises structure at the scale of local patches, with each $N \times N$ region in an image classified as real or fake. By processing input data at different scales, the discriminator with the coarsest scale has the widest receptive field and provides a more global perspective of the image, while the discriminator at the finest scale encourages the generator to produce intricate details. The ultimate output of $\mathcal{D}$ is obtained by running these discriminators convolutionally across the image and averaging their responses.

3.3. Loss functions

Previous studies [8, 57] have demonstrated the benefits of combining the GAN goal with other losses in line art colourisation tasks. While the discriminator's role remains unchanged, the generator is tasked to fool the discriminator and minimise other losses simultaneously. Hence, we utilised the standard conditional min-max losses [38] with the feature matching loss [46], the perceptual loss [24], and the L1 loss. We simplify the following expression from $\mathcal{G}(\mathbf{x}, \mathbf{h}, \mathcal{F}_1(\mathbf{x}))$ to $\mathcal{G}(\mathbf{x}, \mathbf{h})$.

Adversarial Loss. Given a generator $\mathcal{G}$, a discriminator $\mathcal{D}$, and dataset ρ, the conditional GAN loss $\mathcal{L}_{\boldsymbol{cGAN}}$ can be expressed as:

$$\begin{aligned}\mathcal{L}_{\boldsymbol{cGAN}}(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\rho_{data}(x,y)}[\log \mathcal{D}(\mathbf{x}, \mathbf{y})] + \\ \mathbb{E}_{(\mathbf{x},\mathbf{h})\sim\rho_{data}(x,h)}[\log(1 - \mathcal{D}(\mathbf{x}, \mathcal{G}(\mathbf{x}, \mathbf{h})))]\end{aligned} \tag{1}$$

where $\mathbf{x}$ denotes the line art, $\mathbf{y}$ denotes the coloured illustration, and $\mathbf{h}$ denotes the user hints. In particular, we use three different discriminators $\mathcal{D}_i$ to compute at the original resolution, half the resolution, and on a quarter of the original resolution.

Feature Matching Loss. The feature matching loss [46] regularises the generator from over-training on the discriminator. Apart from minimising the cGAN loss, the generator also has to generate images with natural properties that closely resemble the real images. Specifically, we minimise the L1 distance between the real and synthetic images on intermediate layers of the discriminator. The feature matching loss $L_{\boldsymbol{FM}}(\mathcal{G}, \mathcal{D}_i)$ is calculated as:

$$\mathcal{L}_{\boldsymbol{FM}}(\mathcal{G}, \mathcal{D}_i) = \mathbb{E}_{(\mathbf{x},\mathbf{y})} \sum_{j=1}^{T} \frac{1}{N_j} [\| \mathcal{D}_i^{(j)}(\mathbf{x}, \mathbf{y}) - \mathcal{D}_i^{(j)}(\mathbf{x}, \mathcal{G}(\mathbf{x}, \mathbf{h})) \|_1] \tag{2}$$

where T denotes the number of layers in the discriminator $\mathcal{D}_i$, N_j represents the number of elements in the j-th layer of the discriminator, and $\mathcal{D}_i^{(j)}$ denotes the extracted feature maps from the j-th layer of $\mathcal{D}_i$. $\mathcal{D}_i$ is only used for extracting the feature maps and does not try to interfere $L_{\boldsymbol{FM}}$.

Perceptual Loss. Utilising perceptual loss [24] from pre-trained networks can slightly increase the performance of the results and has shown benefits in training line art colourisation models [8, 57]. In this case, a VGG19 [49] network $\mathcal{F}_2$ pre-trained on ImageNet [10] is used for computation:

$$\mathcal{L}_{\boldsymbol{perc}}(\mathcal{G}) = \sum_{k=1}^{N} \frac{1}{M_k} [\| \mathcal{F}_2^{(k)}(\mathbf{y}) - \mathcal{F}_2^{(k)}(\mathcal{G}(\mathbf{x}, \mathbf{h})) \|_1]] \tag{3}$$

where the M_k denotes the number of elements in the k-th layer of VGG19, $\mathcal{F}_2^{(k)}$ is the feature map by the k-th layer.

L1 Loss. Combining the GAN loss with a more traditional loss, such as L1 distance, has shown benefits of encouraging less blurring [23]:

$$\mathcal{L}_{\boldsymbol{L1}}(\mathcal{G}) = \mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{h})} [\| \mathbf{y} - \mathcal{G}(\mathbf{x}, \mathbf{h}) \|_1]] \tag{4}$$

Due to the nature of L1 loss, the generator will attempt to produce images with colours as similar as possible to the target image. As the scribble hints reveal the precise colours at certain regions, the generator will utilise this information more effectively, hence potentially improving the colour accuracy of the output.

With λ denoting the weight of each loss, our final objective becomes:

$$\mathcal{G}^* = \underset{\mathcal{G}}{argmin} \left(\max_{\mathcal{D}} \sum_{i=3}^{3} \mathcal{L}_{\boldsymbol{cGAN}}(\mathcal{G}, \mathcal{D}_i) \right) + \lambda \sum_{i=3}^{3} \left(\mathcal{L}_{\boldsymbol{FM}}(\mathcal{G}, \mathcal{D}_i) + \mathcal{L}_{\boldsymbol{perc}}(\mathcal{G}) + \mathcal{L}_{\boldsymbol{L1}}(\mathcal{G}) \right) \tag{5}$$

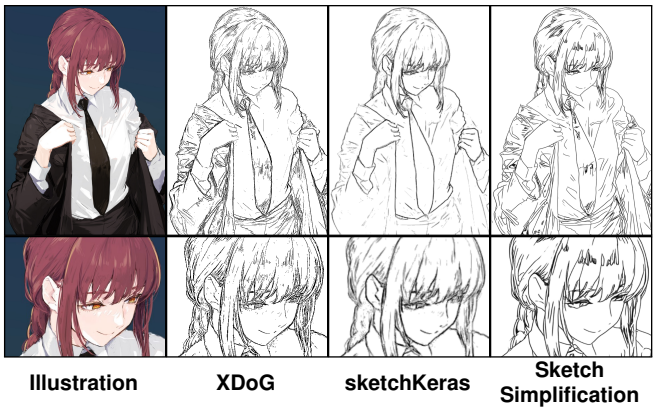

Figure 4. **Example of the results of sketch extraction methods.** Row 1 shows the illustration and its corresponding synthetic line art. Row 2 displays a small region of the image in the first row to show the generated details.

4. Experiment

4.1. Data Processing

Dataset. Danbooru2021 [2] and Nico-Opendata [22] have provided public large-scale Anime illustration datasets. However, both datasets are unsuitable for our task due to the presence of falsely categorised sketches/line arts and obsolete-style illustrations in the datasets. These noises are difficult to eradicate and may hinder the training process. Creating a filtering metric and manually processing the data will necessitate a significant amount of redundant labour. As a result, we gathered our own dataset consisting of $106,933$ illustrations from the internet for training and testing purposes. All the raw illustrations are resized in scale, with the shorter side having a length of 512 pixels.

Line art extraction. The absence of an illustration/line art pair dataset vastly increased the challenge of the model generalisation ability to 'wild' data. Collecting such a dataset is difficult since the pairs require a pixel-wise match. Therefore, our solution is to use algorithms to extract line art from illustrations. There exists a distinction between synthetic and authentic line art, as illustrated in Figure 6. Authentic line arts usually lack eye detail and contain varying line widths. In order to relieve the effect of model over-fitting on the synthetic line art, we used multiple methods of line art extraction as illustrated in Figure 4. XDoG [53] produces images with clear, sharp edges but with large amounts of noise and artefacts. SketchKeras [35] generates images with a pencil sketch style, with varying thicknesses of lines indicating the brightness of the regions on the illustration. Sketch Simplification [48] can produce images closest to digital line art. Since the line thickness generated by Sketch Simplification is dependent on the resolution of the input image, we first process by enlarging the

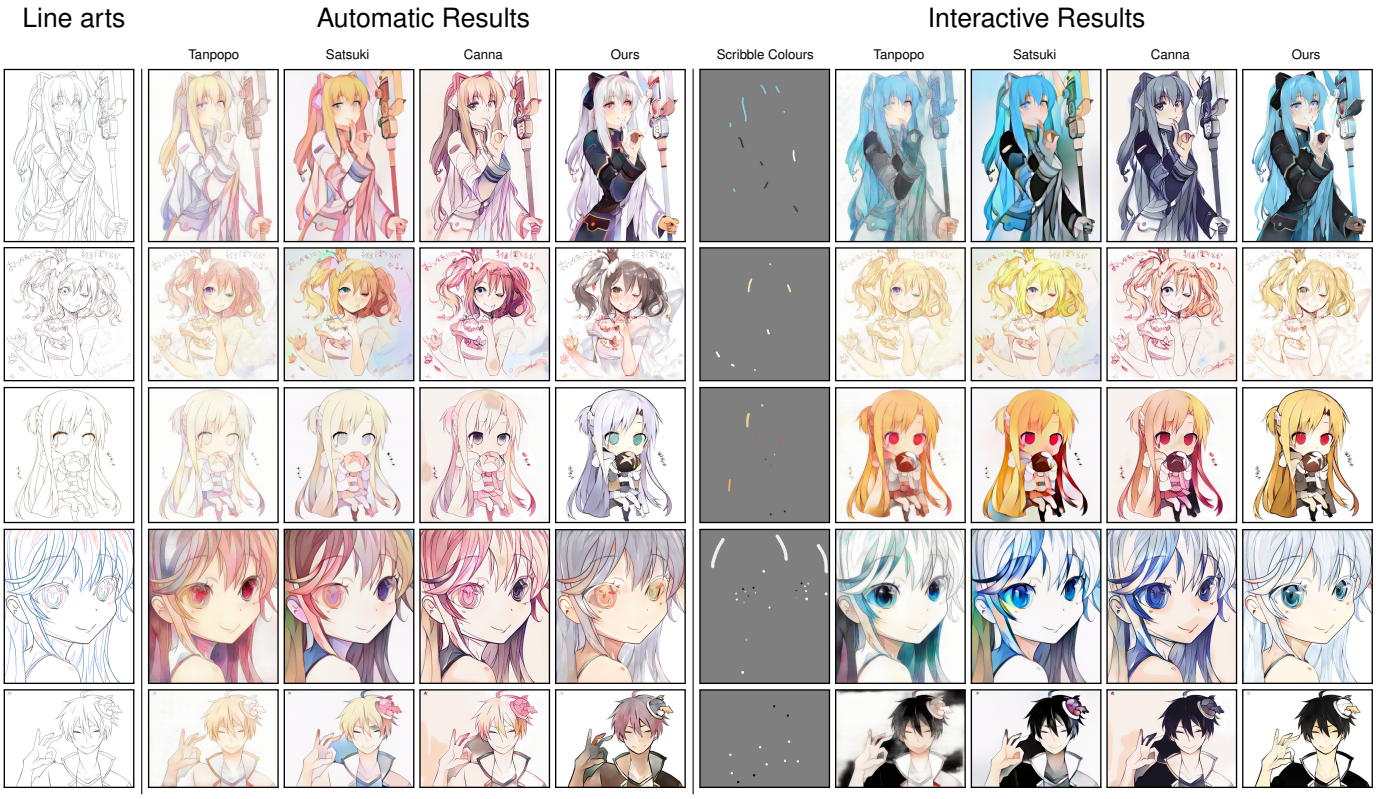

Figure 5. **Visual Comparison.** We compare the results of our proposed framework with previous studies. Column 1 shows the authentic line arts. Columns 2-5 show a comparison of automatic results from the three models of Petalica Paint [43] with ours without user input. Column 6 shows input scribble colours generated using our interactive user interface, best viewed on grey background. Columns 7-10 show the results from Petalica Paint [43] and our model with scribble colour inputs. The automatic results in rows 4-5 show some failures of our model. In row 4, the model cannot colourise the eyes properly. In row 5, the model failed to recognise the hand of the character. However, these problems can be resolved by providing colour hints to the line arts. All images are from our test dataset.

image by a scale factor of 2 with a pre-trained DCSCN [55] network, then extract the line art, followed by resize back to the original size. The type of sketch is randomly chosen during training, and the probability is $p_1 = 0.2$, $p_2 = 0.4$, $p_3 = 0.4$ for XDoG, sketchKeras, or Sketch Simplification to be chosen. The parameters of the XDoG algorithm are set to $\psi = 1 \times 10^9$, $\gamma = 0.95$, $\epsilon = -10$, $\kappa = 4.5$, and $\sigma \in U(0.3, 0.5)$.

User hint simulation. One of the most intuitive ways for users to control the outcome of colourisation is to 'scribble' some colours on the line art to indicate the preferred colours in a region. In order to mimic the process, a hint simulation algorithm is required during training. We followed the approach by Zhang et al. [61], as randomly sampled points are good enough to simulate point-based inputs. The locations of the sampled pixels are determined by $I = P > |\xi|$ where $P \in \mathbb{R}^{1\times H\times W}$ and $\forall p \in P$, $p \sim U(0, 1)$, and $\xi \sim N(1, 0.0001^2)$. With these positions of interest, we draw disks with random radii $r \sim U(2, 16)$, $r \in \mathbb{Z}^+$ on hint map H with colour values corresponding to the positions on the illustration and a binary mask value of 1 to indicate the location of the disk.

Data augmentation. The training dataset illustrations are randomly cropped to a size of 512×512 pixels and have a probability of $p = 0.5$ for horizontal flipping and $p = 0.1$ for vertical flipping before line art extraction and hint simulation. The synthesized line art may undergo a random dilation or erosion to modify the width of the outlines, with a probability of $p_1 = 0.05$ for dilation and $p_2 = 0.25$ for erosion.

4.2. Training Details

We utilised the Pytorch [41] framework to develop our networks, OpenCV2 [4] and Pillow [9] for image processing, Albumentations [5] for data augmentation, and Tkinter for designing the user interface.

Our model was trained on an NVIDIA RTX 3090 Ti 24GB GPU. To train both the generator and discriminator networks, we utilised the AdamW [36] optimiser with momentum hyperparameters $\beta_1 = 0.5$ and $\beta_2 = 0.99$, and the learning rate set to 0.0002. Due to limited VRAM, we trained the model using mixed precision with both float16 and float32 during training. Additionally, we integrated DropPath [27] modules to enhance

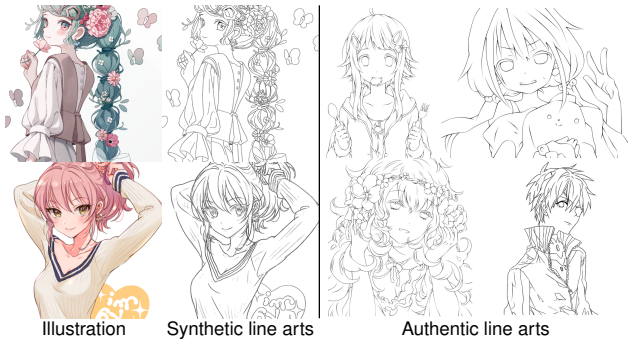

Figure 6. **Synthetic and Authentic line art.** Sample images from our dataset with matching synthetic line arts. The first synthetic line art is generated with Sketch Simplification [48] and the second line art is generated with sketchKeras [35].

training efficiency and slightly improve model performance. The drop path rate increases linearly with the model's depth, with a maximum drop path rate of 0.2. The loss weights for each component were set to $\lambda_{cGAN} = 0.1$, $\lambda_{FM} = 1.0$, $\lambda_{perc} = 10^6$, $\lambda_{L1} = 1.0$; various combinations of λs were tested, and this is our final configuration. The generator and discriminator parameters are updated once per iteration, and all models were trained from scratch.

Figure 10 demonstrates that the line art extraction algorithms are ineffective when the background lacks distinct borders. The resulting line arts contain unnatural patterns that do not provide any valuable content to the model. We trained our model using these illustrations for the initial 900k iterations, which did not produce aesthetically pleasing backgrounds as anticipated. Since the background introduces additional complexity and is not the main focus of our research, we filtered out all non-white background images, resulting in $17,519$ white background images, and trained the model for an additional 200k iterations.

4.3. User Interface

Our system can colourise a 512x512 line drawing in under 100ms, providing real-time performance that offers instantaneous visual feedback following incremental changes in line art colorization applications.

To create a convenient painting environment, we have developed a user interface that includes two canvases, an adjustable pen size scroll, and a colour picker, as shown in Figure 9. The first canvas is designed for line drawings and scribbling hints, while the second canvas displays the colourised results. We used the tkinter package to create the drawing canvas, where a circle is drawn around the user's left-click coordinate, and as the cursor is moved, a new disc is drawn at the updated position. Unfortunately, we were unable to develop an eraser tool as we could not detect if the cursor was overlaid over scribbles, so we implemented a "clear" tool instead. This tool can remove all the color scribbles from the line drawings.

4.4. Visual Comparison

The illustrations generated by the models of Petalica Paint [43] displayed a lack of gradients and highlights, and often suffered from color bleeding, as can be seen in Figure 5. On the other hand, our approach generates images with superior perceptual quality, featuring reduced color bleeding and more realistic textures throughout the entire image. Unfortunately, we were unable to compare our method with more recent approaches, as many of them do not provide an easily accessible webpage or software for generating outputs.

4.5. Discussions and Limitations

Dataset. Although we used multiple strategies to improve the robustness of our model, the model still occasionally fails to capture the authentic data distribution due to the difference between synthetic and authentic line art, shown in Figure 6. Hence, a clean, high-quality dataset containing pairs of illustrations and sketches will undisputedly enhance the performance of our model. Nevertheless, collecting such a dataset is extremely difficult because the pairs must be pixel-perfect matches, which requires significant effort to filter out noisy images. Another approach is to collect illustration raw files with separable layers (e.g., Photoshop files), but it remains challenging because there are rare publicly available files on the internet.

In addition, our dataset is unbalanced, with more female characters than male characters and a more frequent appearance of certain skin colours. This might lead the model to over-fit to a particular style. If we could obtain a more diversified dataset, the model would be able to better generalise the colourisation patterns and produce better outcomes.

Experiment. Despite utilising a high-performance consumer-level GPU to train our framework, we encountered several limitations due to its limited VRAM. As a result, the scalability of our model, training duration, batch size, and dataset size were all restricted, preventing our model from reaching its full potential. Furthermore, due to limited access to codes and time constraints, we were unable to conduct a quantitative comparison with other works by training each model from scratch.

Network Architecture. In our study, we employed a discriminator architecture inspired by PatchGAN, but with ConVNeXt modules instead of the original PatchGAN discriminator. The ConVNeXt modules feature convolutional layers with larger kernel sizes, which provide a broader perceptual field. Consequently, all the critics in each discriminator have a complete view of the image, eliminating the need to penalize the image at the patch scale. Future research should explore reducing the

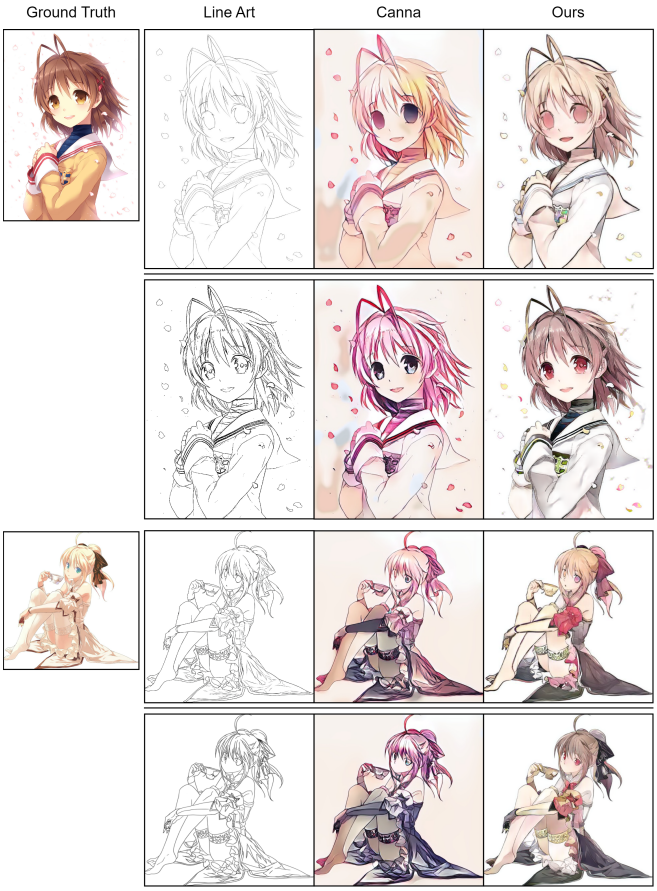

Figure 7. **Comparison on automatic colourisation results from synthetic and authentic line arts**. The first rows display authentic line art colouring results, whereas the second rows show synthetic line art colouring results. Results from synthetic line arts produce more perceptually gratifying results since the framework is specifically trained to colourise line art with these kinds of patterns. Despite this, our method is still capable of creating realistic images with real-world line art by utilising all of the strategies discussed.

perceptual field of the convolutional layers to evaluate if this new architecture can genuinely improve the model's performance. Additionally, some studies suggest that giving line art as input to the discriminator can lead to overfitting to synthetic line art, as authentic line art has a distinct pattern.

Another avenue to explore is a multi-generator framework like Style2Paint V3 [60]. With multiple generators for different purposes, the task for each generator would be simpler, potentially improving the coloured outcome. However, this approach would increase the model's scale, making higher hardware requirements necessary unless each network's scope is reduced.

Due to insufficient foundational knowledge and time constraints, we were unable to explore contemporary state-of-the-art approaches such as denoising diffusion probabilistic models [20, 50], vector quantization [40, 13], or vision transformer

Figure 8. **The evolution of our model.** Version 1 (left) lacked colour accuracy and realism, while Version 2 (right) has successfully addressed these issues. The colours in Version 2 are not only more accurate but also richer and more nuanced, reflecting improvements made in the loss functions and network architectures. These changes have resolved inaccuracies in the colourisation process, resulting in significantly improved performance in Version 2.

architectures [11, 33] to optimize our framework. In the future, we aim to focus more on designing the network architecture and proposing new ideas to the deep learning community.

Results. Our approach illustrated plausible coloured results in Figures 5 and 12. Yet, there are several issues with our framework. Figure 11 depicts the limitations of our framework. Firstly, our system cannot colourise rough draughts properly. This is mostly because line art extraction algorithms produce a neat, continuous line drawing, whereas rough drawings contain several imprecise strokes. In addition, our framework cannot fill large areas with constant colour. We are unable to identify the root of the second issue; however, a multi-generator system might mitigate this effect since an additional generator would smooth out the colours. Lastly, our model is incapable of handling intricate drawings; when there is an excessive quantity of content on the line art, especially with a complicated background, our model cannot recognise the regions correctly.

5. Conclusion

In this paper, we present a learning-based image synthesis framework that can efficiently and intelligently colour line drawings in an interactive manner. Our proposed network architectures are optimised to achieve colour consistency and produce high-quality illustrations. We investigated a potential application for our model and showed that it is intuitive to use with no learning curve. Despite the model being trained entirely on synthetic data, extensive results demonstrated the generalisability and reliability of our approach.

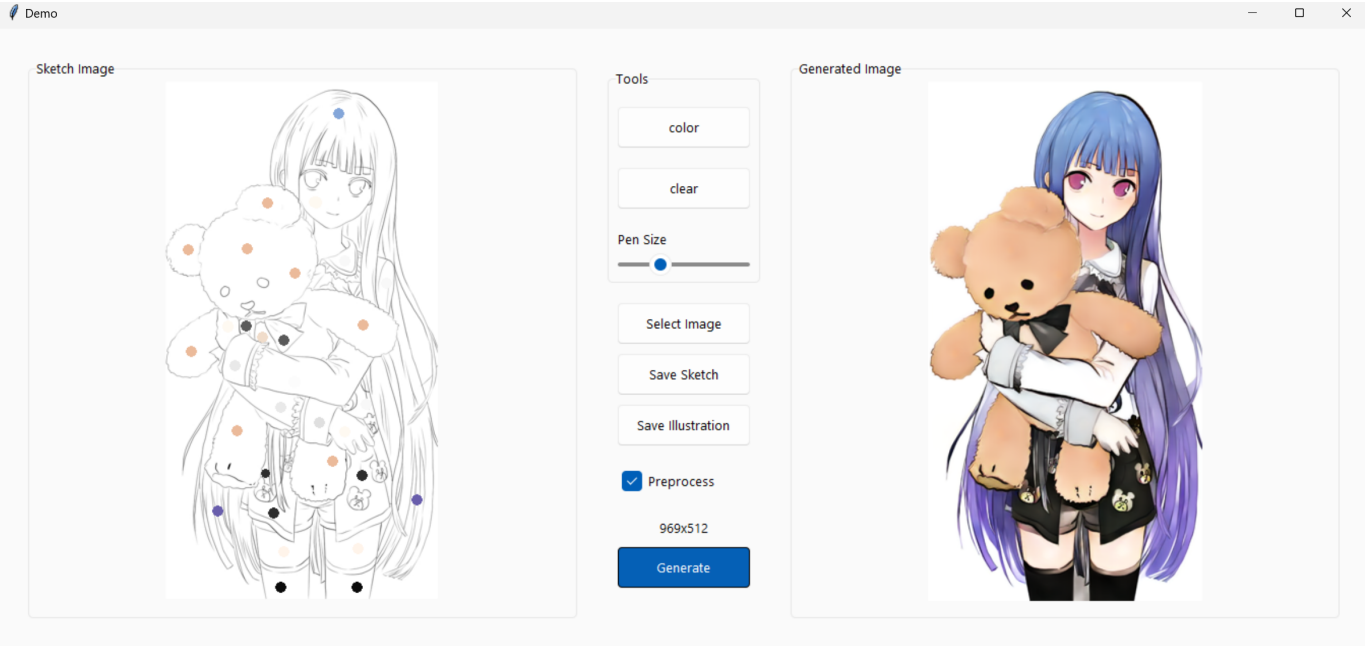

Figure 9. **GUI software environment.** The sketch/line art image and user hints are presented on the left. In the centre is a toolbar with a colour picker, a clear tool, and buttons for selecting line art, saving the line art, and saving the colourised illustration. Following that, there is a ticker for resizing the input to our framework's default image size. The coloured output is presented on the right. The user may simply add colours to line art by selecting the required line art, picking desired colours, scribbling on the line art, and clicking the generate button to construct the artwork.

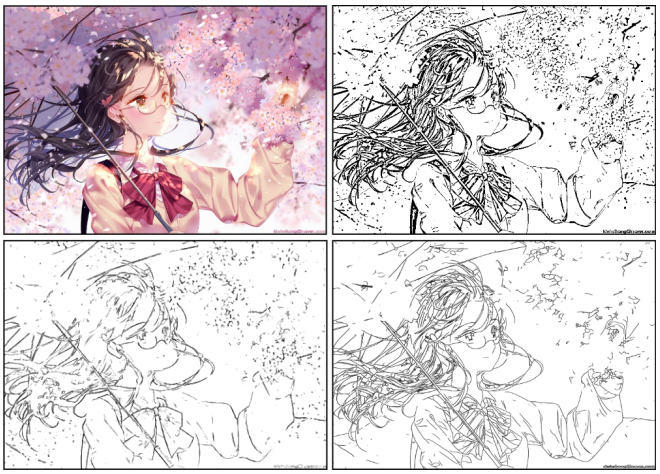

Figure 10. **Sketch extraction failure case.** The algorithms cannot perform optimally when the illustration has a background with vague boundaries. From top left to bottom right: the illustration, XDoG [53], sketchKeras [35], and Sketch Simplification [48].

Figure 11. **Colourisation failure cases.** Column 1 shows the result for rough draught input. Column 2 shows the result when we try to fill the background with a consistent colour. Column 3 shows the result with complicated input. Results in Columns 1 and 3 show some colour distortions with the automatic results. The colour consistency can be improved by providing user hints.

Figure 12. **More examples of our cGAN-based framework.** Given the line art and colour hints (left), our framework can wisely generate the colourised image (right).

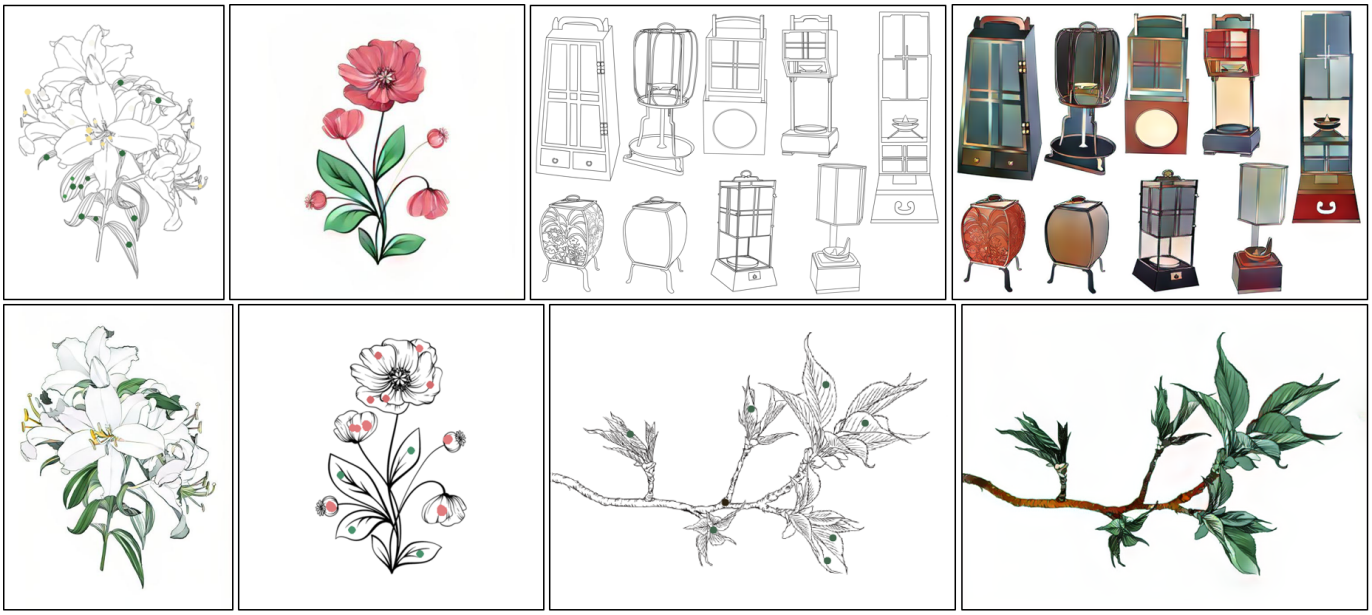

Figure 13. Despite being trained on the Anime dataset, our model can colourise non-anime images rather effectively.

References

[1] Xiaobo An and Fabio Pellacini. Appprop: All-pairs appearance-space edit propagation. *ACM Trans. Graph.*. Vol. 27. No. 3, pp. 1–9. 2008. 1.

[2] Anonymous, Danbooru community, and Gwern Branwen. Danbooru2021: A large-scale crowdsourced and tagged anime illustration dataset. https://www.gwern.net/danbooru2021. 2022. 2, 4.

[3] Matthew Baas. Danbooru2018 pretrained resnet models for pytorch. https://rf5.github.io/2019/07/08/danbuuro-pretrained.html. 2019. 2.

[4] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 2000. 5.

[5] Alexander V. Buslaev, Alex Parinov, Eugene Khvedchenya, Vladimir I. Iglovikov, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *arXiv preprint arXiv:1809.06839*. 2018. 5.

[6] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. Palette-based photo recoloring. *ACM Transactions on Graphics (Proc. SIGGRAPH)*. Vol. 34. No. 4. 2015. 1.

[7] Xiaowu Chen, Dongqing Zou, Qinping Zhao, and Ping Tan. Manifold preserving edit propagation. *ACM Trans. Graph.*. Vol. 31. No. 6. 2012. 1.

[8] Yuanzheng Ci, Xinzhu Ma, Zhihui Wang, Haojie Li, and Zhongxuan Luo. User-guided deep anime line art colorization with conditional adversarial networks. In *Proceedings of the 26th ACM International Conference on Multimedia*. Ser. MM '18, pp. 1536–1544. 2018. 1, 3, 4.

[9] Alex Clark. Pillow (pil fork) documentation. 2015. 5.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. 2009. 4.

[11] Alexey Dosovitskiy *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*. 2021. 7.

[12] Yuki Endo, Satoshi Iizuka, Yoshihiro Kanamori, and Jun Mitani. Deepprop: Extracting deep features from a single image for edit propagation. *Computer Graphics Forum*. Vol. 35. No. 2, pp. 189–201. 2016. 1.

[13] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *arXiv preprint arXiv:2012.09841*. 2020. 7.

[14] Kevin Frans. Outline colorization through tandem adversarial networks. *arXiv preprint arXiv:1704.08834*. 2017. 1.

[15] Chie Furusawa, Kazuyuki Hiroshiba, Keisuke Ogaki, and Yuri Odagiri. Comicolorization: Semi-automatic manga colorization. In *SIGGRAPH Asia 2017 Technical Briefs*. Ser. SA '17. 2017. 0, 1.

[16] Chie Furusawa, Shinya Kitaoka, Michael Li, and Yuri Odagiri. Generative probabilistic image colorization. *arXiv preprint arXiv:2109.14518*. 2021. 1.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. Vol. 27. 2014. 0, 1.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. 2016. 2.

[19] Paulina Hensman and Kiyoharu Aizawa. Cgan-based manga colorization using a single training image. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 03, pp. 72–77. 2017. 1.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*. Vol. 33, pp. 6840–6851. 2020. 1, 7.

[21] Yi-Chin Huang, Yi-Shin Tung, Jun-Cheng Chen, Sung-Wen Wang, and Ja-Ling Wu. An adaptive edge detection based colorization algorithm and its applications. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*. Ser. MULTIMEDIA '05, pp. 351–354. 2005. 1.

[22] Hikaru Ikuta, Keisuke Ogaki, and Yuri Odagiri. Blending texture features from multiple reference images for style transfer. In *SIGGRAPH ASIA 2016 Technical Briefs*. Ser. SA '16. 2016. 4.

[23] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017. 0, 1, 3, 4.

[24] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision – ECCV 2016*, pp. 694–711. 2016. 1, 3, 4.

[25] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*. 2018. 1.

[26] Hyunsu Kim, Ho Jhoo, Eunhyeok Park, and Sungjoo Yoo. Tag2pix: Line art colorization using text tag with secat and changing loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9055–9064. 2019. 1.

[27] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *International Conference on Learning Representations*. 2017. 5.

[28] Junsoo Lee, Eungyeup Kim, Yunsung Lee, Dongjun Kim, Jaehyuk Chang, and Jaegul Choo. Reference-based sketch image colorization using augmented-self reference and dense semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. 1.

[29] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*. Ser. SIGGRAPH '04, pp. 689–694. 2004. 1.

[30] Xujie Li, Hanli Zhao, Guizhi Nie, and Hui Huang. Image recoloring using geodesic distance based color harmonization. *Computational Visual Media*. Vol. 1. 2015. 1.

[31] Yuanzhen Li, Edward Adelson, and Aseem Agarwala. Scribbleboost: Adding classification to edge-aware interpolation of

local image and video adjustments. *Computer Graphics Forum*. Vol. 27. No. 4, pp. 1255–1264. 2008. 1.

[32] Yifan Liu, Zengchang Qin, Tao Wan, and Zhenbo Luo. Auto-painter: Cartoon image generation from sketch by using conditional wasserstein generative adversarial networks. *Neurocomput.*. Vol. 311. No. C, pp. 78–87. 2018. 1.

[33] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021. 7.

[34] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022. 0, 3.

[35] lllyasviel. Sketchkeras. https://github.com/lllyasviel/sketchKeras. 2017. 4, 6, 8.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*. 2019. 5.

[37] Qing Luan, Fang Wen, Daniel Cohen-Or, Lin Liang, Ying-Qing Xu, and Heung-Yeung Shum. Natural Image Colorization. In *Rendering Techniques*. 2007. 1.

[38] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*. Vol. abs/1411.1784. 2014. 0, 1, 3.

[39] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*. 2021. 1.

[40] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu koray. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*. Vol. 30. 2017. 7.

[41] Adam Paszke *et al.* Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. 2019, pp. 8024–8035. 5.

[42] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *Computer Vision and Pattern Recognition (CVPR)*. 2016. 1.

[43] pixiv. Petalica paint. https://petalica.com. 2017. 0, 1, 5, 6.

[44] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM Trans. Graph.*. Vol. 25. No. 3, pp. 1214–1220. 2006. 1.

[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241. 2015. 0, 2.

[46] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. Vol. 29. 2016. 0, 1, 3, 4.

[47] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6836–6845. 2017. 1.

[48] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Mastering sketching: Adversarial augmentation for structured prediction. *ACM Trans. Graph.*. Vol. 37. No. 1. 2018. 4, 6, 8.

[49] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. 2014. 4.

[50] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Ser. Proceedings of Machine Learning Research, pp. 2256–2265. 2015. 1, 7.

[51] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018. 0, 1, 3.

[52] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *The European Conference on Computer Vision Workshops (ECCVW)*. 2018. 1.

[53] Holger Winnemoeller, Jan Kyprianidis, and Sven Olsen. Xdog: An extended difference-of-gaussians compendium including advanced image stylization. *Computers & Graphics*. Vol. 36, pp. 740–753. 2012. 4, 8.

[54] Kun Xu, Yong Li, Tao Ju, Shi-Min Hu, and Tian-Qiang Liu. Efficient affinity-based edit propagation using k-d tree. *ACM Trans. Graph.*. Vol. 28. No. 5, pp. 1–6. 2009. 1.

[55] Jin Yamanaka, Shigesumi Kuwashima, and Takio Kurita. Fast and accurate image super resolution by deep cnn with skip connection and network in network. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Ser. NIPS'17, pp. 217–225. 2017. 1, 5.

[56] Liron Yatziv and Guillermo Sapiro. Fast image and video colorization using chrominance blending. *IEEE Transactions on Image Processing*. Vol. 15. No. 5, pp. 1120–1129. 2006. 1.

[57] Mingcheng Yuan and Edgar Simo-Serra. Line art colorization with concatenated spatial attention. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3941–3945. 2021. 0–4.

[58] Lvmin Zhang, Yi Ji, Xin Lin, and Chunping Liu. Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier gan. In *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 506–511. 2017. 1.

[59] Lvmin Zhang, Chengze Li, Edgar Simo-Serra, Yi Ji, Tien-Tsin Wong, and Chunping Liu. User-guided line art flat filling with split filling mechanism. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021. 1.

[60] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. Two-stage sketch colorization. *ACM Trans. Graph.*. Vol. 37. No. 6. 2018. 0, 1, 7.

[61] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*. Vol. 9. No. 4. 2017. 1, 5.

A. Network Details

In this section, we describe the details of each block function displayed in Figure 3.

A.1. ConvNeXt Block & Encoder Block

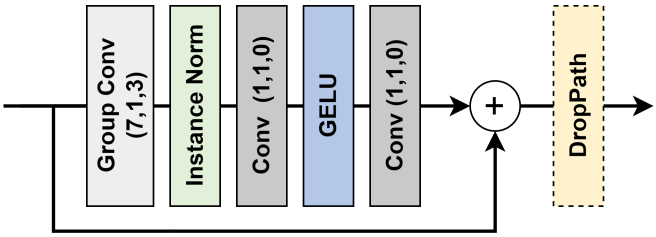

Figure A.1. ConvNeXt block [5] — Kernel size, stride, and padding of the convolution shown in the parenthesis respectively.

The input x is processed through a grouped convolutional [3] layer, then normalised each channel via instance normalisation [6] and passes through two 1×1 convolution layers with a GELU [2] activation function in between. The DropPath [4] layer has a chance to activate, which disables the whole block and the input will skip this block directly to the next.

The encoder block function is similar to the ConvNeXt block. However, the group convolutional layer is changed into a standard convolutional layer to reduce the number of parameters with the same kernel size, stride, and padding settings.

A.2. Concatenation and Spatial Attention Block

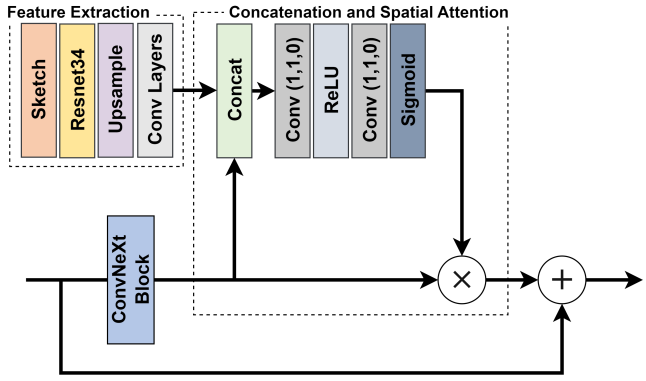

Figure A.2. Concatenation and Spatial Attention Block [7].

The input is first processed by a ConvNeXt [5] block. Afterwards, it is concatenated with the features extracted by the local feature network and convolved with two convolutional layers with a ReLU [1] and a Sigmoid activation function to obtain the attention map. This attention map is multiplied element-wise with the processed input and added to the input. The attention map is a matrix of values between $[0, 1]$, in a trained network, the important features in the input will have a corresponding value closer to 1 in the attention map, and insignificant features will have a value closer to 0. This mechanism enables the network to 'focus' on the important features, thus generating a higher-quality image.

A.3. Downsample block

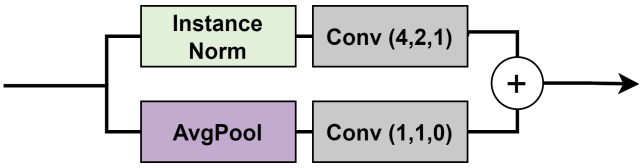

Figure A.3. Downsample block.

Inspired by ConvNeXt, we separated the downsample function from the ConvNeXt block. The input is first processed by an instance normalisation layer followed by a convolutional layer to downsample the input features and map the features to a higher dimension. The outputs from the convolutional layer are added to the input processed by an average pooling layer and a convolutional layer to map the features to a higher dimension.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). 2018. 12.

[2] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. 2016. 12.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. Ser. NIPS'12, pp. 1097–1105. 2012. 12.

[4] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *International Conference on Learning Representations*. 2017. 12.

[5] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022. 12.

[6] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. 2016. 12.

[7] Mingcheng Yuan and Edgar Simo-Serra. Line art colorization with concatenated spatial attention. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3941–3945. 2021. 12.