








StencilTorch: An Iterative and User-Guided Framework for Anime Lineart Colorization

Yliess Hati^{1,2}(✉) , Vincent Thevenin² , Florent Nolot¹ ,
Francis Rousseaux¹ , and Clement Duhart² 

¹ University of Reims Champagne-Ardenne, Laboratory CReSTIC,
51 100 Reims, France

{florentnolot, francisrousseau}@univ-reims.fr

² Léonard de Vinci Pôle Universitaire, Research Center,
92 916 Paris La Défense, France

{yliess.hati, clement.duhart}@devinci.fr,
vincent.thevenin@edu.devinci.fr

Abstract. Automatic lineart colorization is a challenging task for Computer Vision. Contrary to grayscale images, linearts lack semantic information such as shading and texture, making the task even more difficult. Modern approaches train a Generative Adversarial Network (GAN) to generate illustrations from user inputs such as color hints. While such approaches can generate high-quality outputs in real-time, the user only interacts with the pipeline once at the beginning of the process. This paper presents StencilTorch, an interactive and user-guided framework for anime lineart colorization motivated by digital artist workflows. StencilTorch generates illustrations from a given lineart, color hints, and a mask allowing for iterative workflows where the output of the first pass becomes the input of a second. Our method improves previous work on both objective and subjective evaluations.

1 Introduction

Motivation. Illustration can be summarized as the succession of four well defined tasks: sketching, inking, coloring, and post processing. Referring to Kandinsky's work [19], the colorization process can change the meaning of an entire piece of art by introducing color schemes, shading, and textures. These last three properties of the painting process turn out to be challenging for the Computer Vision task of automatic colorization. Contrary to its grayscale counterpart [8, 14, 45], linearts lack semantic information making the task even more difficult. Materials and 3D shapes can only be inferred from their silhouettes.

Problem. Previous work introduced the use of GAN [9] methods, one of the most widespread neural architectures for image generation. These approaches can generalize and produce perceptively qualitative illustrations. While some work focused on the use of color-based hints [5, 7, 12, 26, 31, 36], others are using

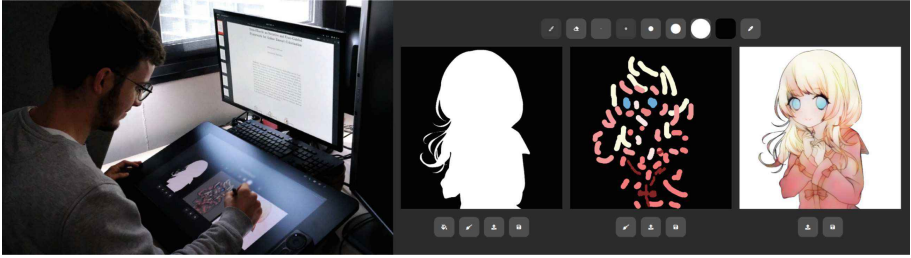


Fig. 1. The figure shows a photo of a user interacting with our model StencilTorch and a screenshot from our Web Application. The left canvas allows the user to provide or draw a mask. Hint maps can be drawn or provided in the middle canvas. Hovering over the hint section reveals a transparent lineart. In the right canvas, the user can upload his lineart and see the resulting illustration. A toolbar with basic digital painting tools is available at the very top, including a pen, an eraser, predefined brush sizes, and a color picker.

style transfer [43] or tags [21] as color cues to condition the output of the model to user intents. In such methods, feature extractors such as Illustration2Vec [35] are used to enforce semantic information on the input.

Previous methods consist of a one-step process where the user is invited to influence the generation process once, in the beginning, using information hints. This type of pipeline is not ideal in the context of creation where the artist wants to iterate and explore the design space of the illustration process. In this paper, we instead formulate the task of automatic colorization as a Human-in-the-loop process where the user collaborates with the machine in an iterative and interactive process to produce the final piece of art.

Solution. We introduce StencilTorch, an iterative and user-guided framework for anime lineart colorization. Our framework is motivated by human workflow and is inspired by previous work done by Ci et al. [5], and follow-up work by Hati et al. [12]. We train a conditional Wasserstein GAN with gradient penalty, c-WGAN-GP for short, to generate illustrations from a given lineart, natural color hints, and a mask describing the region of the image that has to be inpainted.

Our model is trained on images curated online using a similar approach to PaintsTorch [12]. The illustrations are post-processed to extract a synthetic lineart using an Extended Difference of Gaussians (xDoG) [41] and a displacement map is extracted to remove lighting and limit the amount of colors from which the color hints are sampled from.

Findings. We evaluated StencilTorch on our curated test dataset against previous work by Zhang et al. [44], PaintsChainer [31], Ci et al. [5], and PaintsTorch [12]. The models are benchmarked using both objective metrics, Fréchet Inception Distance (FID), Learned Perceptual Image Patch Similarity (LPIPS), and a subjective metric Mean Opinion Score (MOS) obtained by conducting a user study. StencilTorch improves previous work on each of those metrics.

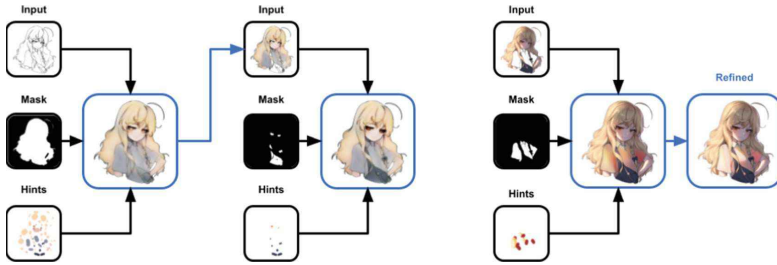


Fig. 2. StencilTorch can paint a lineart given user hints and a mask. The diagram illustrates an iterative workflow. The artist explores a potential colorization for its drawing on the left and explore an alternative colorization on the right after the introduction of shading and lighting in the input image.

Our approach not only enables iterative and collaborative workflows between the human and the computer but also captures the additional painting style introduced by the user, such as shading and illumination.

Contributions.

- A new synthetic dataset curation pipeline for producing qualitative input and output pairs for automatic anime lineart colorization. Our pipeline partially removes texture and shadow information from the illustrations, and uses semantic color segmentation to produce better synthetic hints for training.
- StencilTorch, an interactive and user-guided framework for anime lineart colorization. Given a lineart, color hints and a mask, StencilTorch generates a colored illustration. The colored output can be used as input for the next iteration enabling natural workflows between the human and the machine. Our model introduces a guide network for the generator’s decoder during training and is trained for both illustration generation and inpainting.
- A study showing that StencilTorch improves previous work on our curated dataset both objectively with the FID and LPIPS metrics, and subjectively with an MOS. We also provide an ablation study to justify our design choices.
- An interactive open-source web application of the StencilTorch framework using TensorFlowJS and Web Canvases to simulate digital art programs.

Implication. Previous work focused on improving the perceptual quality of the generation process [5, 12, 31, 45]. Our contribution is orthogonal and can be combined with such improvements enabling seamless and natural digital illustration workflows. Stenciltorch allows the artist to collaborate with the machine, as shown in Fig. 2. We believe that such approaches will allow AI-driven tools to be part of the creative environment and enable fast and broad exploration.

Reproduction. For reproducibility and transparency, we published our implementation and experimentation at www.github.com/yliess86/PaintsTorch2.

2 Background and Related Work

Generative Adversarial Network (GAN). GAN [9] approaches have proven to be one of the best end-to-end methods for generating high-quality images beating previous methods such as autoencoders [16], variational autoencoders [22], and flow networks [23]. GAN driven pipelines are competing with transformer-based architectures [40] and denoising diffusion probabilistic models [17]. They are efficient at inference, enabling closed-to or real-time applications.

Vanilla. The vanilla GAN introduced by Goodfellow et al. [9] consists of a generator \mathcal{G} trained to produce images $\mathcal{G}(z)$ similar to the data distribution fed to a discriminator \mathcal{D} trained to differentiate fake images from true images x . The networks are jointly trained to optimize a Min-Max objective shown in Eq. 1 and can be further split as distinct objectives as shown in Eq. 2.

$$\min_{\mathcal{D}} \max_{\mathcal{G}} \mathbb{E}_x[\log(\mathcal{D}(x))] + \mathbb{E}_z[\log(1 - \mathcal{D}(\mathcal{G}(z)))] , \quad (1)$$

$$\frac{1}{m} \sum_{i=1}^m [\log(\mathcal{D}(x_i)) + \log(1 - \mathcal{D}(\mathcal{G}(z_i)))] , \quad \frac{1}{m} \sum_{i=1}^m [\log(\mathcal{D}(\mathcal{G}(z_i)))] . \quad (2)$$

In practice, this formulation is unstable. The generator often saturates if it does not keep up with the discriminator, which task is most of the time easier to optimize. It also suffers from vanishing gradients and mode collapse, where the generator finds a simple solution that fools the discriminator failing at generating diverse enough outputs. The literature includes techniques to overcome those issues, such as the hinge loss [25], the Wasserstein distance [2], gradient penalty [10], and the use of batch [18] and spectral [30] normalization strategies.

Wasserstein Distance. One powerful alternative to the vanilla formulation is the Wasserstein GAN or WGAN for short [2]. It resolves both the mode collapse and the vanishing gradients issues. The output activation of the discriminator is changed from a sigmoid to a linear function. This change turns the discriminator network into a critic rating the quality of the generated output rather than discriminating the fake from the real. The critic and generator objectives are shown in Eq. 3. Gradient clipping is used to satisfy the Lipschitz constraint.

$$\frac{1}{m} \sum_{i=1}^m [\mathcal{D}(x_i)] - [\mathcal{D}(\mathcal{G}(z_i))] , \quad \frac{1}{m} \sum_{i=1}^m [\mathcal{D}(\mathcal{G}(z_i))] . \quad (3)$$

Gradient Penalty. Instead of gradient clipping, Gulrajani et al. [10] enforce a constraint on the critic such that its gradients with respect to the inputs are unit vectors. The critic loss is augmented with an additional term shown in Eq. 4 where \hat{x} is sampled from a linear interpolation between real and fake samples to satisfy the critic’s Lipschitz constraint.

$$\lambda \mathbb{E}_{\hat{x}} [(\|\nabla_{\hat{x}} \mathcal{D}(\hat{x})\|_2 - 1)^2] . \quad (4)$$

User Conditioning. A GAN network can be further conditioned on user inputs such as class labels [29] and transformed into a multi-modal model. The final user is granted fine-grain control over the generated output by conditioning both the generator and the discriminator or critic networks on such input. This approach is called a conditional GAN or c-GAN for short.

Color Hints. User-guided automatic colorization through color hints is one of the leading approaches. It enables fine-grain control on the image generation process by pin-pointing colors in the regions where a specific color is intended. This type of color control has been popularized by Zhang et al. [45] and is used by the majority of the following methods [5, 7, 12, 26, 36, 44].

In their work, Zhang et al. [45] claim that randomly activating pixels from the original illustration during training as color hints is enough to enable fine-grain control on the output. However, Hati et al. [12] demonstrate that this is not enough. The lack of semantic information, shading, and texture is to blame. They propose using simulated strokes to represent the user inputs more faithfully.

Both approaches employ a U-Net [33] architecture with ResNetXt blocks [13] using dilated convolution to increase the receptive field and favor speed, and Pixel Shuffling [37] to limit the generation of artifacts.

Style Transfer. An alternative input style for color hints is the use of style transfer. The user provides a target illustration from which the network has to learn the style properties while conserving the content of the given lineart using VGG16 [38] or VGG19 features as perceptual loss proxies. This approach has been extensively studied in previous work [8, 14, 43]. Style transfer can be combined with color palette conservation to transfer colors from the style image.

Tags. Tag-based automatic colorization has been explored by Kim et al. [21]. They introduce feature attribute vectors to condition the network for lineart colorization. By describing features such as hairstyle, hair color, eye color, and others, the network can transfer the user intent into the painting. They also introduce SECat, a neural network module to help the model focus on details.

3 Proposed Method

Our method, StencilTorch aims at generating colored illustrations from a given lineart, color hints, and a mask to favor iterative and creative workflows where the output of the first pass can become the input of a second. This section discusses the importance of the data curation process, the input generation, our model architecture, the introduction of inpainting, and our training curriculum.

3.1 Dataset Curation

The challenge of anime lineart colorization suffers from a lack of qualitative and publicly available datasets. Finding corresponding pairs of lineart and illustrations in abundance is a challenge. Online scrapping of anime drawings and synthetic lineart extraction are the preferred methods [5, 12, 45].

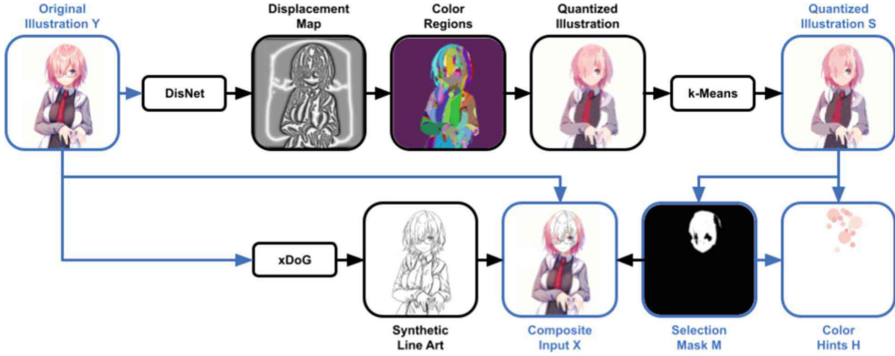


Fig. 3. StencilTorch input generation process diagram. An illustration is sampled from our curated dataset. This illustration is used to generate a synthetic lineart using the xDoG method. A displacement network DisNet is trained to generate displacement maps from colors illustrations used to produce color regions. The color regions are then quantized and reduced via k-means clustering to eliminate most of the lighting, shading, and texture-specific colors. The regions are sampled to produce a mask and color hints. The synthetic lineart, the mask, and the color hints are combined into a single input.

The few public datasets available to the community we are aware of are inconsistent in terms of perceptual quality, the nature of the images (e.g. comics pages, photos), and present illustrations from artists of different backgrounds, levels, and styles. For these reasons, we have curated a custom dataset.

Our dataset contains 21,930 scrapped anime-like illustrations for training, 3,545 for testing, and is manually filtered to ensure a perceptive quality across all samples and remove inappropriate (e.g. gore, mature, and sexual) content. This process is motivated by previous work on PaintsTorch [12] where the authors demonstrate the implications of the dataset quality in the generation process.

We find important to highlight that any dataset [1] used for the challenge of anime-like line art colorization is biased. They are reflections of the anime sub-culture and communities from which they are drawn. The drawings are mostly figures of female characters with visible skin. This may justify the overall salmon watercolor tone attributed to the illustrations produced by current works.

3.2 Input Generation

The lack of lineart and illustration pairs require the use of complex input generation pipelines. Previous work [5, 12] proposed to generate synthetic linearts out of the curated paintings. xDoG [41] is the technique used to extract such information. It produces qualitative and clean lines from complex colored drawings.

The color hints are randomly selected by sampling parts of the illustration or averaging colors from random locations. However, we claim that this approach is

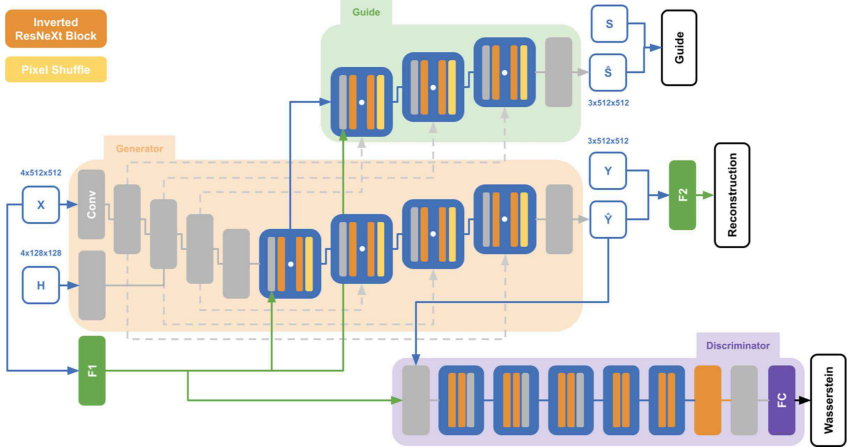


Fig. 4. StencilTorch architecture schematic. The pipeline employs five neural networks: a generator \mathcal{G} responsible for generating fake illustrations from a black and white lineart, color hints, a mask, and a feature vector, a feature extractor \mathcal{F}_1 , a VGG16 content extractor \mathcal{F}_2 , a guide network responsible for guiding the generation process, and a discriminator network \mathcal{D} .

not appropriate for proper colorization. Such processes do not distinguish mid-tone colors from lighting, shading, nor texture, leading to misrepresentations of the user intent. Randomly activating pixels does not account for color bleeding and messy inputs. In this paper, we present a new color hint scheme selection process solving the aforementioned issues and better matching the behavior of our end users. Our pipeline is summarized in the diagram shown in Fig. 3.

The new StencilTorch input pipeline relies on color regions extraction. We train a ResNet [13] model to regress displacement maps from colored illustrations on the DanbooRegion dataset [28]. The displacement map is robust to noise and is used to extract unique color sections. We assign each region to its median color and further process the output by reducing the number of colors to 25 using k-means clustering. We empirically selected the number clusters to apply color quantization in RGB space without sacrificing too much detail. This process limits the shading, lighting, and texture information present in the input illustration. The color hints are finally sampled from this image and drawn to the hint texture as circles of various sizes to simulate a variety of user strokes.

3.3 Model Architecture

The model architecture used in StencilTorch is a c-WGAN-GP inspired by previous work by Ci et al. [5] and Hati et al. [12]. Our adversarial framework consists of five neural networks: a generator, a discriminator, also called a critic, a feature extractor, a style network, and a guide network. A schematic of the entire architecture is shown in Fig. 4.

The Generator is a U-Net [33] autoencoder with inverted ResNeXt blocks [42]. The architecture takes advantage of the dilated depth-wise separable convolutions [4] to augment the network capacity while being computationally efficient. The discriminator network presents similar properties.

The missing semantic information is recovered by a conditioning feature vector extracted by Illustration2Vec [35], a ConvNet \mathcal{F}_1 . The vector is fed to both the generator and the discriminator to condition the generation process.

StencilTorch is train to optimize a weighted mix of losses shown in Eqs. 5 and 6: an adversarial loss, a content preserving loss, a guide loss for the generator, a Wasserstein distance and a gradient penalty for the discriminator.

$$\mathcal{L}_{\mathcal{D}} = \mathcal{L}_w + \mathcal{L}_{gp} , \quad (5)$$

$$\mathcal{L}_{\mathcal{G}} = \lambda_{adv}\mathcal{L}_{adv} + \mathcal{L}_{cont} + \mathcal{L}_{guide} . \quad (6)$$

The adversarial loss \mathcal{L}_{adv} shown in Eq. 7 trains the generator to fool the discriminator. We choose to optimize the hinge loss [25] as it stabilizes training while improving the perceptual quality. The loss is weighted by $\lambda_{adv} = 1e - 4$.

$$\mathcal{L}_{adv} = \mathbb{E}_{\hat{Y}}[\max(1 - \mathcal{D}(\hat{Y}, \mathcal{F}_1(X)), 0)] . \quad (7)$$

The content loss \mathcal{L}_{cont} shown in Eq. 8 ensures the preservation of the original content present in the linart. The content information is provided by a VGG16 model we refer to as \mathcal{F}_2 pretrained on ImageNet [34].

$$\mathcal{L}_{cont} = \frac{1}{chw} \|\mathcal{F}_2(\hat{Y}) - \mathcal{F}_2(Y)\|_2 . \quad (8)$$

The guide loss \mathcal{L}_{guide} shown in Eq. 9 helps the decoder disentangling feature maps to recover flat colorization in early stages. The loss is inspired by previous work by Zhang et al. [43]. The guide network is used during training, removed at inference time, and has the same architecture as the generator decoder. It produces an output \hat{S} we optimize to be close to the quantized illustration S obtained in the input generation process.

$$\mathcal{L}_{guide} = \frac{1}{chw} \|\hat{S} - S\|_2 . \quad (9)$$

The Wasserstein loss \mathcal{L}_w shown in Eq. 10 is responsible for training the discriminator. For the adversarial loss, we optimize its hinge formulation.

$$\mathcal{L}_w = \mathbb{E}_{\hat{Y}}[\max(1 + \mathcal{D}(\hat{Y}, \mathcal{F}_1(X)), 0)] + \mathbb{E}_Y[\max(1 - \mathcal{D}(Y, \mathcal{F}_1(X)), 0)] . \quad (10)$$

StencilTorch optimizes an additional gradient penalty term \mathcal{L}_{gp} shown in Eq. 11. As proposed by Karras et al. [20] the gradient penalty is augmented with a drifting term to further enforce stabilization during the training of the discriminator. The gradient penalty is weighted by the hyperparameter $\lambda_{gp} = 10$ and a drifting weight $\epsilon_{drift} = 1e - 3$.

$$\mathcal{L}_{gp} = \lambda_{gp}\mathbb{E}_{\hat{Y}}[(\|\lambda_{\hat{Y}}\mathcal{D}(\hat{Y}, \mathcal{F}_1(X))\|_2 - 1)^2] + \epsilon_{drift}\mathbb{E}_Y[\mathcal{D}(Y, \mathcal{F}_1(X))^2] . \quad (11)$$

3.4 Mask Inpainting

Selection masks are part of the tools used by digital artists to select regions of the illustration and avoid bleeding artifacts. Inspired by this kind of workflow, we introduce the use of selection masks to limit the GAN generation process to specific areas of the input. This process allows for partial inpainting of the illustration. It is responsible for our iterative workflow where the output of a first pass can be used as the input of a second pass, with or without intermediate modification. As shown in Fig. 2, it allows the generator to pick insights from the artist’s style and naturally fits the standard digital artist workflow.

During training, masks are generated by sampling and merging the color regions computed in the input generation process. The selection mask is first used for the synthetic lineart and its corresponding colored illustration compositing. The lineart is drawn where the mask is white, the illustration where it is black. The composite input is stacked with the mask and is used instead of the black-and-white lineart to feed the generator. The mask is finally reused for inpainting when computing the final generated output illustration and acts as attention weights during the entire process.

3.5 Curriculum Learning

Curriculum learning [3,6] is a type of pipeline in which a model is trained on tasks that gradually increase in difficulty. It has been shown to increase models’ performances in convolutional neural networks [11].

In StencilTorch, we apply curriculum learning by progressively increasing the number of regions used for inpainting during training. In early stages, the surrounding pixels are used to provide context for inpainting. At the end of training, the model is forced to paint the entire black-and-white lineart. This feature is implemented by sampling the proportion of regions to inpaint $p = \max(\epsilon, u)$, $u \sim \mathcal{U}(0, 1)$ where ϵ is linearly annealed over time $\epsilon = 0.9 \rightarrow 0$.

4 Implementation

StencilTorch is trained using the PyTorch library for fast prototyping and iterations. Pytorch is, however, not suited for building client-side Web Applications. More actions have to be taken.

Model Export. We exported the PyTorch model as an ONNX model using the TorchScript intermediate representation. Some layers must be adapted to account for supported operations between the two frameworks. The ONNX model is then transpiled into a TensorFlow model that is further processed to be exported as a TensorFlowJS instance. This final instance of the model can be used in a client-side Web App.

Web Application. We also implement a Web Application that provides similar tooling to digital painting software, such as a brush, an eraser, a color picker, a color wheel with different brush sizes, and a canvas. A screenshot of the Web App is provided in Fig. 1. The user is invited to draw or import a lineart, a mask, and color hints. The inputs are processed by the TensorFlowJS model and outputted on a result canvas in real-time. The user can download every input and output for reproduction or integration with their favorite digital art tool.

5 Evaluation Setup

Data. We evaluate and train our method on our custom dataset containing illustrations scrapped from the web and filtered manually, 21,930 for training, and 3,545 for test. The images are resized to 512 on their smallest side and randomly cropped to 512×512 during training, and center cropped at test time.

Metrics. To measure the perceptual quality of the generated images, the GAN literature reports evaluations of both objective and subjective metrics. Art is a subjective matter, human evaluation is required.

Concerning the objective metrics, we measure two neural feature-based scores in our test set, the FID [15], and the LPIPS [44]. Those metrics uses an ImageNet [34] pretrained neural network, respectively InceptionNet [39], and AlexNet [24], to measure similarities between a pair of images, a fake one, the generated image, and a real one, the target, in feature space.

To assess subjective preferences, we conducted a user study to evaluate an MOS. The study consists of showing colorized linearts to the user whose task is to rate the quality of the illustration on a scale from 1 (bad) to 5 (excellent). Our study consists of 20 images for each model. Our study population comprises 46 individuals aged from 16 to 30, with 26% women and 35% experienced in drawing or colorization.

Baseline. We evaluate StencilTorch against previous works: PaintsChainer [31], Ci et al. [5], Zhang et al. [45], and PaintsTorch [12].

Training. The models are trained end-to-end using the AdamW optimizer [27] with a learning rate $\alpha = 1e - 4$ and beta parameters $\beta_1 = 0.5$ and $\beta_2 = 0.9$. They are trained for 40 epochs using a batch size of 32 on each of the four GPUs during 24 hours straight.

Measurements. All experiments and measurements are realized on a DGX1-station from NVidia equipped with an Intel Xeon E5-2698 20-Core Processor, 512 Go of DDR4 RAM, and four V100 GPUs with 32 Go of VRAM each.

Table 1. Benchmark and ablation of StencilTorch against previous works [5, 12, 31, 45]. The table reports both FID and LPIPS evaluations. BN stands for Batch Normalization, G for Guide network, and C for Curriculum Learning. In average StencilTorch improves previous work. The evaluation use different amount of color hints, No Hints, regular Hints, and Full Hints.

Model	No Hint	Hints	Full Hints	Mean
FID ↓				
Zhang et al.	134.06	274.87	242.58	245.33
PaintsChainer	54.97	99.63	112.16	93.02
Ci et al.	52.48	96.22	106.73	85.14
PaintsTorch	51.54	95.71	98.37	81.87
StencilTorch	51.16	94.40	106.05	81.63
StencilTorch + BN	70.50	118.82	106.33	96.78
StencilTorch + G	85.00	91.60	93.80	89.98
StencilTorch + G + C	103.12	159.27	153.42	136.03
LPIPS ↓				
Zhang et al.	0.28	0.46	0.26	0.37
PaintsChainer	0.28	0.71	0.60	0.54
Ci et al.	0.23	0.62	0.59	0.48
PaintsTorch	0.18	0.59	0.56	0.44
StencilTorch	0.16	0.51	0.58	0.40
StencilTorch + BN	0.19	0.50	0.56	0.41
StencilTorch + G	0.31	0.50	0.58	0.46
StencilTorch + G + C	0.21	0.30	0.55	0.36

Table 2. Mean Opinion Scores, and the 95% confidence t -test p -values for the mean comparing StencilTorch to previous works [5, 12, 31, 45] on our curated test set. StencilTorch improves previous contributions.

Model	MOS ↑	STD ↓	p -value ↓
PaintsChainer	1.79	0.51	$6.04e^{-23}$
Ci et al.	2.18	0.56	$7.72e^{-18}$
Zhang et al.	2.83	0.67	$9.84e^{-08}$
PaintsTorch	3.05	0.42	$9.15e^{-09}$
StencilTorch	3.71	0.28	

6 Results

We quantitatively in Table 1, 2, and qualitatively in Fig. 2, 5, and 7 show that StencilTorch outperforms previous works [5, 12, 31, 45] in FID, LPIPS and MOS.



Fig. 5. Mosaic of inputs and output pairs generated with one StencilTorch pass. The top image represent the colorization, bottom right the lineart, bottom center the inpainting mask and bottom right the color hints.

Benchmark. StencilTorch is benchmarked against previous contributions in FID, LPIPS and MOS evaluations on our curated test set. One challenge for such evaluation is the generation of consistent color hints for every lineart of the test set. We propose to evaluate every approach using no hints, regular hint sampling, and full hints, meaning dense color sampling from the color region extracted during the input generation. On average, our approach improves previous work by Zhang et al. [45], PaintsChainer [31], Ci et al. [5], and PaintsTorch [12].

Ablation. The ablation study evaluates the impact of our design choices. We report the FID and the LPIPS on our test set. Our curriculum strategy helps the generator handle color hints on every of the three hint quantities we evaluated.

Limitations. Although our approach allows the generation of illustrations given a black-and-white lineart, a mask, and a hint map, the generated output often lacks shadows and textures compared to previous work by Hati et al. [12]. Such phenomena can be observed in Fig. 7. The flat coloring produced by our method is certainly due to the guide network involved in the entire decoding process during training. It seems, however, that the problem fades away when our model is used for inpainting as it was intended (see Fig. 2).

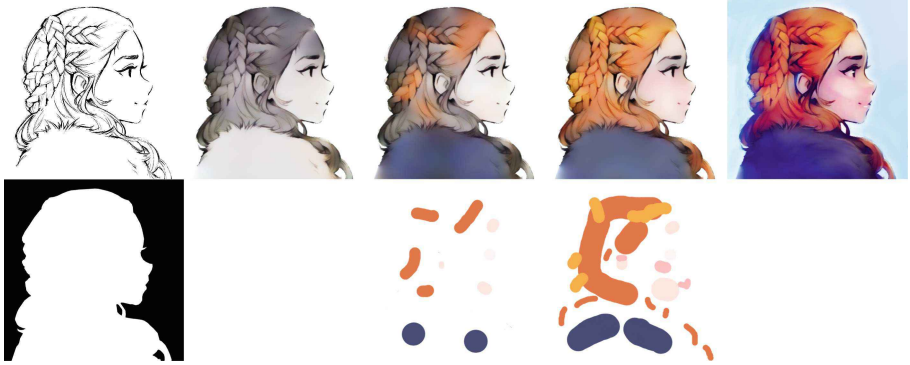


Fig. 6. Impact of the hint concentration on StencilTorch output. The lineart and mask are shown in the first column. Column 2 to 4 demonstrate different concentration, from no hint to a sufficient amount. Top rows represent the outputs, and bottom rows the hints. The last illustration is manually refined by an artist.



Fig. 7. Comparison of StencilTorch, PaintsTorch [12], and PaintsChainer [31] from left to right given the same lineart and hint map. The illustration generated by StencilTorch is the result of a single pass without any user in the loop. StencilTorch and PaintsTorch provide cleaner outputs. The colors of StencilTorch appear flat but do not present artifacts in comparison to PaintsTorch. The output of our model can be refined in collaboration with the user using inpainting mode.

7 Conclusion

Our work StencilTorch addresses the need for AI-driven tools that naturally integrate into the artist workflow and allow fast prototyping and iteration in collaboration with the machine. While current approaches have focused on improving the generation of user-guided anime lineart colorization, we explored the use of inpainting masks. This reformulation of the colorization process allows natural workflows to emerge. The output of a first pass is a potential input for a second. Our study demonstrates that our approach beats previous work on subjective metrics, FID, LPIPS, and objective metrics, MOS.

Future Work. The use of curriculum learning to progressively inpaint the lineart does not seem to generate good quality illustrations when no hints are provided as shown in Fig. 6. Recent advances in the domain of Denoising Diffusion Probabilistic Model (DDPM) such as DALL-E 2 [32] demonstrate unprecedented performance in image generation while providing natural conditioning. In future work, we want to explore the use of such a technique for automatic anime lineart colorization in the continuity of our quest for human-computer collaboration.

References

1. Anonymous, community, D., Branwen, G.: Danbooru 2020: A large-scale crowd-sourced and tagged anime illustration dataset, January 2021. <https://www.gwern.net/Danbooru2020>
2. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 214–223. PMLR, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. <https://proceedings.mlr.press/v70/arjovsky17a.html>
3. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 41–48. ICML 2009, Association for Computing Machinery, New York, NY, USA (2009). <https://doi.org/10.1145/1553374.1553380>
4. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1800–1807 (2017). <https://doi.org/10.1109/CVPR.2017.195>
5. Ci, Y., Ma, X., Wang, Z., Li, H., Luo, Z.: User-guided deep anime line art colorization with conditional adversarial networks. In: Proceedings of the 26th ACM International Conference on Multimedia, pp. 1536–1544. MM 2018, Association for Computing Machinery, New York, NY, USA (2018). <https://doi.org/10.1145/3240508.3240661>
6. Elman, J.L.: Learning and development in neural networks: the importance of starting small. *Cognition* **48**(1), 71–99 (1993)
7. Frans, K.: Outline colorization through tandem adversarial networks. CoRR abs/1704.08834 (2017). [arxiv:1704.08834](https://arxiv.org/abs/1704.08834)
8. Furusawa, C., Hiroshiba, K., Ogaki, K., Odagiri, Y.: Comicolorization: semi-automatic manga colorization. In: SIGGRAPH Asia 2017 Technical Briefs, SA 2017, Association for Computing Machinery, New York, NY, USA (2017). <https://doi.org/10.1145/3145749.3149430>

9. Goodfellow, I.J., et al.: Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems, vol. 2, pp. 2672–2680. NIPS 2014, MIT Press, Cambridge, MA, USA (2014). <https://dl.acm.org/doi/10.5555/2969033.2969125>
10. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of Wasserstein GANs. In: Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017), <https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccc52936e27cbd0ff683d6-Paper.pdf>
11. Hacoheh, G., Weinshall, D.: On the power of curriculum learning in training deep networks. In: Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 2535–2544. PMLR (2019). <https://proceedings.mlr.press/v97/hacohen19a.html>
12. Hati, Y., Jouet, G., Rousseaux, F., Duhart, C.: PaintsTorch: a user-guided anime line art colorization tool with double generator conditional adversarial network. In: European Conference on Visual Media Production. CVMP 2019, Association for Computing Machinery, New York, NY, USA (2019). <https://doi.org/10.1145/3359998.3369401>
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016
14. Hensman, P., Aizawa, K.: CGAN-based manga colorization using a single training image. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 3, pp. 72–77. IEEE Computer Society, Los Alamitos, CA, USA, Nov 2017. <https://doi.org/10.1109/ICDAR.2017.295>, <https://doi.ieeecomputersociety.org/10.1109/ICDAR.2017.295>
15. 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: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 6629–6640. NIPS’17, Curran Associates Inc., Red Hook, NY, USA (2017)
16. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science* **313**(5786), 504–507 (2006)
17. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. *Adv. Neural Inf. Process. Syst.* **33**, 6840–6851 (2020)
18. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37, pp. 448–456. PMLR, Lille, France, 07–09 July 2015. <https://proceedings.mlr.press/v37/ioffe15.html>
19. Kandinsky, W., Sadleir, M.: Concerning the Spiritual in Art. Dover Publications, New York (1977). (oCLC: 3042682)
20. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. *CoRR abs/1710.10196* (2017). [arxiv.org:1710.10196](https://arxiv.org/1710.10196)
21. Kim, H., Jhoo, H.Y., Park, E., Yoo, S.: Tag2pix: line art colorization using text tag with Secat and changing loss. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9055–9064 (2019). <https://doi.org/10.1109/ICCV.2019.00915>

22. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings (2014). [arxiv.org:1312.6114](https://arxiv.org/abs/1312.6114)
23. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1×1 convolutions. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. *Commun. ACM* **60**(6), 84–90 (2017)
25. Lim, J.H., Ye, J.C.: Geometric GAN (2017)
26. Liu, Y., Qin, Z., Wan, T., Luo, Z.: Auto-painter: cartoon image generation from sketch by using conditional Wasserstein generative adversarial networks. *Neurocomputing* **311**, 78–87 (2018)
27. Loshchilov, I., Hutter, F.: Fixing weight decay regularization in adam. CoRR abs/1711.05101 (2017). [arxiv.org:1711.05101](https://arxiv.org/abs/1711.05101)
28. Zhang, L., Ji, Y., Liu, C.: DanbooRegion: an illustration region dataset. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12358, pp. 137–154. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58601-0_9
29. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint [arXiv:1411.1784](https://arxiv.org/abs/1411.1784) (2014)
30. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. CoRR abs/1802.05957 (2018). [arxiv.org:1802.05957](https://arxiv.org/abs/1802.05957)
31. Pixiv: Pelica Paint. https://petalica-paint.pixiv.dev/index_en.html (2017)
32. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip LATENTS. arXiv preprint [arXiv:2204.06125](https://arxiv.org/abs/2204.06125) (2022)
33. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
34. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. *Int. J. Comput. Vision (IJCV)* **115**(3), 211–252 (2015). <https://doi.org/10.1007/s11263-015-0816-y>
35. Saito, M., Matsui, Y.: Illustration2vec: a semantic vector representation of illustrations. In: SIGGRAPH Asia 2015 Technical Briefs, pp. 5:1–5:4. SA 2015, ACM, New York, NY, USA (2015). <https://doi.org/10.1145/2820903.2820907>
36. Sangkloy, P., Lu, J., Fang, C., Yu, F., Hays, J.: Scribbler: controlling deep image synthesis with sketch and color. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 6836–6845. IEEE Computer Society (2017). <https://doi.org/10.1109/CVPR.2017.723>
37. Shi, W., et al.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1874–1883 (2016)
38. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015). [arxiv.org:1409.1556](https://arxiv.org/abs/1409.1556)
39. Szegedy, C., et al.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)

40. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc. (2017). <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>
41. Winnemöller, H., Kyprianidis, J.E., Olsen, S.C.: XDOG: an extended difference-of-Gaussians compendium including advanced image stylization. *Comput. Graph.* **36**(6), 740–753 (2012). <https://doi.org/10.1016/j.cag.2012.03.004>, www.sciencedirect.com/science/article/pii/S009784931200043X, 2011 Joint Symposium on Computational Aesthetics (CAe), Non-Photorealistic Animation and Rendering (NPAR), and Sketch-Based Interfaces and Modeling (SBIM)
42. Xie, S., Girshick, R.B., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5987–5995 (2017)
43. Zhang, L., Ji, Y., Lin, X., Liu, C.: 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). <https://doi.org/10.1109/ACPR.2017.61>
44. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
45. Zhang, R., et al.: Real-time user-guided image colorization with learned deep priors. *ACM Trans. Graph.* **36**(4), 1–11 (2017). <https://doi.org/10.1145/3072959.3073703>