AnimEditH: AI-Powered Anime Character Editing Web App *

Capstone Design Project Proposal

Chingis Oinar[1][2018315169],
Park Soo Hun[2][2016314364],
Eunmin Kim[3][2018315083], and
Gong He[4][2018313592]

[1] Sungkyunkwan University
2066, Seobu-ro, Jangan-gu, Suwon-si, Gyeonggi-do, Republic of Korea
[2] chingisoinar@gmail.com
[3] soohun96@g.skku.edu
[4] eunmin88@gmail.com
[5] gh970818@hotmail.com

Abstract. With the rise of popularity of anime growing steadily, the demand for unique styles continue. Current anime styled face generation tools are not very user friendly, do not provide an end to end program and do not give much freedom of style to the user. Another problem is that majority of them only **focus on** female anime faces. There are also only female face datasets readily available online. We provide a web app that gives the user much more freedom of creating their own anime styled face. We allow users to free draw their face, colorize to their liking, and use inpainting that allows quick editing at any stage of the process. We created our own dataset of anime styled male faces due to the lack of available datasets. We also have a history functionality that saves all past created images and stores them. After the user is satisfied with their anime styled image, they can download the image for their own personal use. For the machine learning part of our web app, we will use Generative Adversarial Networks (GAN).

Keywords: Anime · GAN · Photo Editor

1 Introduction

Anime is hand-drawn and computer animation that was first produced, with commercial purposes, in 1917, Japan. A distinct art style was developed in the 1960s, inspired by the works of eminent cartoonist Osamu Tezuka. Subsequently, it has been spreading widely, developing a colossal domestic as well as global audience. Moreover, it is now being produced by cartoonists outside of its country of origin. The global anime market size reached a value of approximately *USD*

* Supported by SKKU.

23.56 billion in 2020 and is estimated to grow up to staggering *USD 25.46 billion* in 2021 [7], whereas the revenue forecast is expected to reach *USD 48.03 billion* in 2028 [7]. Thus, the compound annual growth rate constituted to be around 9.5% per year making. Furthermore, Sony Pictures acquired Crunchyroll, an online anime streaming service, for *USD 1.175 billion*, with plans to widespread anime to everyone at home [6]. According to Tony Vinciquerra, Chairman and CEO of Sony Pictures Entertainment Inc., Crunchyroll brings a tremendous value to Sony's current anime businesses, including Funimation as well as company's terrific partners at Aniplex and Sony Music Entertainment [6]. Recently, the Otaku Coin cryptocurrency, which has been running on the Ethereum platform since 2018, has announced its plans to produce an isekai anime with Non-Fungible Tokens (NFTs) [8]. Although the items to be produced possess no inherent attributes, nor do they interact with any broader game rules, the concept has become viral and attracted a huge audience around it, so participants started to produce their artwork and story concepts [8]. The combination of factors above makes Anime a large business area with huge future potentials.

However, it takes tremendous efforts and a huge amount of time to acquire the skills needed to draw decent anime characters. To bridge this gap, there have been many projects and works introduced on this matter. However, the largest part of available datasets online contain data samples of female characters dominantly. Therefore, in order to provide a full control over the process, we need to tackle this issue of the imbalance between female and male characters. In addition to that, most of the current solutions generate the end product only, hence they do not provide a freedom to edit them.
Finally, inspired by the demand and some issues about current solutions our contribution can be summarized as follows:

- We provide more freedom to control anime character editing by allowing users to edit and colorize images as they wish, so they could create their own characters in end to end manner.
- We tackle the problem of gender imbalance in current datasets available online and make our own dataset of male character in addition to them.

2 Motivation & Objective

With the up rise of popularity of anime, we want to provide an AI-powered end to end Anime Styled Face Generation web app to creators or fans of the genre. Since creating anime styled faces and editing is usually a very time consuming task and is quite tedious, we want to develop an app that will shorten time needed and reduce effort required. We want to build it so that users can free draw a face or create a face using tags to generate an anime styled face and make it editable/color-able or completely randomize a design if they choose to do so. Other similar apps only generate anime faces and are not editable afterwards, only downloadable. There aren't many options of how they want it to look like before generating, which makes it difficult to generate a style of the user's liking. Majority of them are randomly generated, so they are not geared towards user

satisfaction. Users may not be content with their randomly generated and may want to change specific features or colors but are unable to. the only way is to use another app where they can edit the generated image. We plan to make an app that allows the user to have complete control over how they want to customize their generated anime styled face using AI models.

3 Related Work

Generative Adversarial Networks. Generative Adversarial Networks (GANs), proposed by Goodfellow et al., have recently achieved impressive results in the field [3]. The core idea can be summarized as training *generator* and *discriminator* networks in minimax game manner, where the former tries to fool the other network by producing high quality output, whereas the latter network tries to classify each image correctly. Specifically, by incorporating two separate networks, generator and discriminator, GAN learns a loss function due to which it is able to produce highly realistic images. Given a training dataset, generative models synthesize new samples from the same distribution. Thus, Generator's objective is to generate data that is indistinguishable from the real data, whereas the Discriminator takes both real and generated data and tries to distinguish them correctly. Furthermore, considering that GAN learns an objective that adapts to the training data, they have been applied to a wide variety of tasks. Recently, GANs have been employed for Image Restoration, Text-to-Image Translation, Image-to-Image Translation, Face Frontal View Generation, Cartoon Characters Generation, Style Transfer, Face Aging and 3D Object Generation.

Many variants of GANs have been proposed for generating images. Mirza & Osindero et al. introduced *Conditional Generative Adversarial Networks* (CGAN) using which they managed to generate MNIST numbers of a particular class [1]. Later, Reed et al. demonstrated how an encoded text can be used to produce an image of interest [5]. Odena et al. proposed ACGAN where auxiliary classifier to predict the condition input was introduced in addition to the vanilla discriminator [2]. Although the training process is quite simple, optimizing such models is not a trivial task. One of many issues faced in the past is a generation of blurry images. Isola et al. introduced a novel architecture of CGANs and demonstrated that GANs can be efficiently used for Image Colourizing tasks [4], which implies filling in blank or empty spots in images with color relative to surroundings. The unique idea proposed by Isola et al. is in the discriminator, *Patch*GAN, that tries to classify if each $N \times N$ patch in an image is real or fake. Thus, it models the image as a Markov random field, assuming independence between pixels. This architecture helps overcome the issue of blurry images and can be regarded as a form of style loss.

4 Problem Statement & Proposed Solution

Animation is a field that uses computer graphics. All works are done by computer programs, and many technologies have been developed to make this easier. However, if you are not an expert at drawing or editing anime characters or if you are doing it as a hobby, drawing and coloring a new character every time can be a huge burden. If the user just selects a color, the picture is colorized, and the computer inpaints the picture by simply marking the part the user wants to erase, it will be able to bring about an innovative change in the creation of animated characters.
Therefore we propose an animation character editing web app. It will include the processes described below.

4.1 Uploading / Drawing

Users can draw anime character faces on the canvas using drawing tools on the web. If you already have a picture and want to edit it, you can upload it to canvas using the uploading button. At this time, the size of the photo is automatically adjusted to the canvas.

4.2 Inpainting

If there is a part that the user wants to erase from the picture, the user can erase the part he wants to erase with the eraser. However, if you erase it like that, it will just leave a blank white space, so you have to connect the lines by hand and add colors. But you can use the inpainting button to automatically fill in the blank space.

4.3 Colorizing

Coloring is also user selectable. A sketch pad is provided for users to color in the desired color. If you don't want to color it yourself, you can also leave it to the AI model[4] to color it at random. To use the model, use the sketch tool to mark color dots on the desired area and press the Colorize button to automatically color it.

4.4 Editing

Our web has features to help the user's drawing process more easily.There is a "Convert to Sketch" function that detects the edge of the picture and converts the picture into sketch format. There is "Convert to Black&White", which converts drawings to black and white, and "Convert to Black&White from Sketch," which colors the sketch form made of edges in black and white.

Fig. 1. Design: Screen 1

Fig. 2. Design: Screen 2

Fig. 3. Design: Screen 3

4.5 History

When sketching or editing using an AI model, it is randomly edited, so you may try to find the best one after several attempts. For this, we provide a history gallery by session. The history gallery allows you to recall or save previously created or modified images.

5 Frontend

5.1 Original Designs

Our original designs were created using Adobe XD. For the first screen, we had an idea for a canvas where users could sketch their drawings to be converted into anime styling. So we added a canvas in the middle as the main feature with editing tools on top. We also had tags on the left and right hand side of the canvas where users could choose features for anime character generation (Eye color, lip color, etc.). At the bottom of the canvas, we added three key buttons: Generate, complete, and colorize. Generate generates an anime styled image using our AI model, complete moves to the second screen, and colorize colorizes the image using an AI model. The second screen is the completion screen. After the user completes their image, they can download or save the file directly into their wanted directory. They can also move the second and third page with the buttons labeled editing and history on the top left and right respectively. The third and final screen was designed so that users could save their image to a local history where they can look at their images again and if necessary, edit them, delete them, or download them. All screens were created with the idea of having anime creation the main feature of our app and everything else is just supplement to that idea. We wanted to make a clean and simple web app that is easy to navigate for people of all ages/skill level to use easily and freely.

5.2 Final Designs

Our final frontend was created in JQuery written in HTML, CSS, JavaScript and Python. The final first screen changed vastly from our original design. First off, we had decided to not do generation AI model due to unstable datasets available. So we moved all AI models and features to the left and right hand side of the canvas. This makes the overall design look much cleaner, instead of having

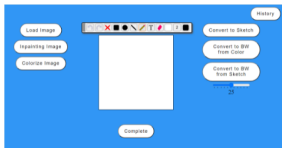

Fig. 4. Final: Screen 1

Fig. 5. Final: Screen 2

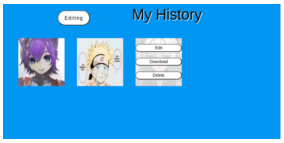

Fig. 6. Final: Screen 3

them all at the bottom of the canvas. Now the only button at the bottom of the canvas is the complete button which moves to the same second screen. We also added more features to the editing tool bar: Undo, Redo, Clear, draw Square, draw ellipse, draw line, text, eraser, choose fill and line color, and line width. The second screen is vastly similar to the original design. We have the final image in the middle of the screen with now only one button at the bottom, download. We got rid of save as because it served a similar functionality to download anyways, so it was a bit redundant to do again. The top left and top right buttons moving to the first and third screen respectively were kept the same. The last and final screen, the history screen's design was also kept pretty similar. We spaced out the images more to clearly differentiate two different images and instead of having a drop down menu, we created the menu to choose delete, download, and edit inside of the image and can only be seen over hover. Overall we made many different changes to the design as we were continuing to develop this web app due to constant changing of our requirements. We kept the overall theme of the web app the same and tried to stay true to this through the end.

6 Backend

We have created several APIs to support the web app and they are as below.

6.1 Request and Response

All the images are passed between frontend and backend in base64 string format.

-	Inpainting	Colorize	Convert Sketch	Convert to B&W	Convert edge toB&W
Request	Color Image w/ marks	B&W Image w/ color dots	Original Image	Colored Image	Sketch (only edges)
Response	Inpainted Image	Colorized Image	Sketch (Only edges)	B&W Image	Image Colorized in B&W

Table 1. API Request & Response formats description

6.2 Preprocessing

All the pictures sent to the APIs after some work on canvas were pictures with a transparent background. However, since this project is related to image processing, even the white background with nothing is also one of the important parts. Therefore, we made and pasted a white background of canvas size of 256 x 256 to the images coming from the frontend and proceeded with the processes chosen.
Augmentation was applied to the image to transform it into a form suitable for application to the model. It includes resizing, toTensor, and normalization.

6.3 Editing

After preprocessing, we feeded the image into an appropriate model that the user selected and the models are descriped in "Machine Learning" parts. If it was editing buttons, we used OpenCV2 library to process the requested functions.

6.4 Postprocessing

Postprocessing is included to remove noise mixed with the final image. The main noise we found was salt&pepper noises. We used the OpenCV2 library for bilateral filter and made edges stronger using graphical functions.

7 Dataset

7.1 Image Data Processing.

Users can get Sketch images as well. For this task, we can apply common image processing techniques, such as Canny edge detector algorithm, and produce corresponding Edge images, as shown in the Fig. 7. In this example, we applied a Bilateral Filter to reduce a noise before passing it to a Canny edge detector (provided by Opencv[6]), which is followed by a bitwise not operator to invert it. Next, we need to provide color-marked images as input for our colorization model. For this task, we convert RGB images to grayscale after which we randomly add color pixels from the original RGB images. The process is summarized in the Fig.8. Finally, we also need to provide partially erased or 'guided' images for our inpainting model as shown in the Fig.9. The image on top is used for guided inpainting. It is useful when users want to modify an image in the edge-level without spending a lot of time. The second image, however, is a partially erased image that is used to erase minor imperfections. Thus, the model will inpaint these images with appropriate colors or information.

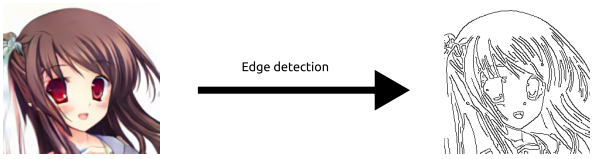

Fig. 7. Image to edge map.

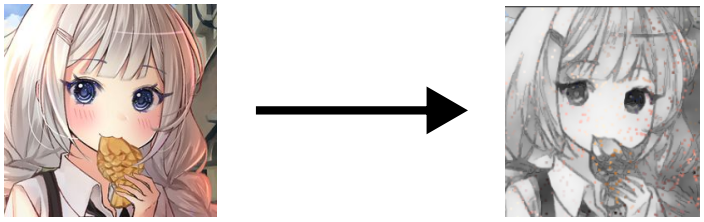

Fig. 8. Image to color-marked image.

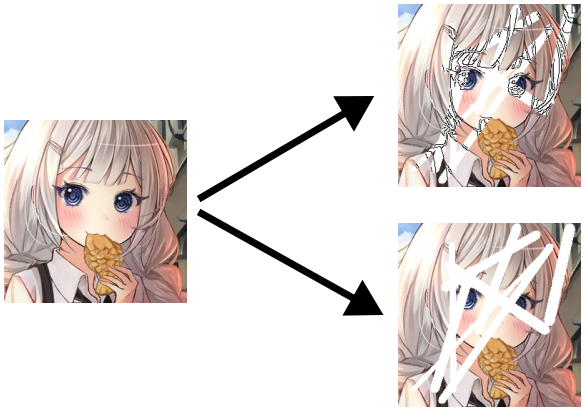

Fig. 9. Image to partially erased image.

7.2 Image Data Collection.

Open Source & Crawling For the female data set, Kaggle [12] data set was used.

[6] https://opencv.org/

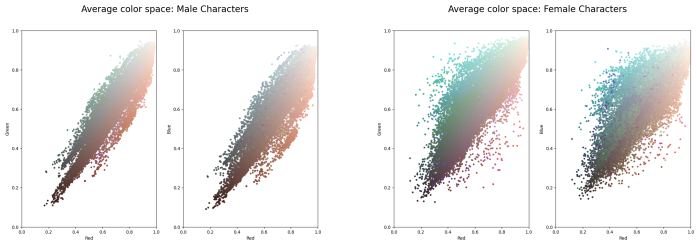

Fig. 10. Average color spaces for female and male characters.

Fig. 11. Male dataset on hypersphere

However, the male dataset could not be found as open source. So I started crawling on a site called safebooru[13]. "Male_focus" was used as a keyword to search in the website, and a total of 50k images were crawled. Faces were recognized and cropped and saved using an open source AI model[14].
We collected a huge amount of data, but it wasn't enough to be the perfect dataset we wanted. Therefore, Dataset cleaning was carried out. We tried to use AI in this part of the work as well. We tried the pseudo-labelling and method of extracting kNN using metric learning, but neither did good performance. Therefore, with the firstly extracted kNN dataset we manually cleaned by our four team members. Our completed dataset is displayed in Fig. 11, and Fig. 12. The Fig.10 shows that the average color space of male characters mostly consists of darker colors than that of female characters. Thus, we can obtain a much wider color range by combining both of them, which might be useful for our coloriza-

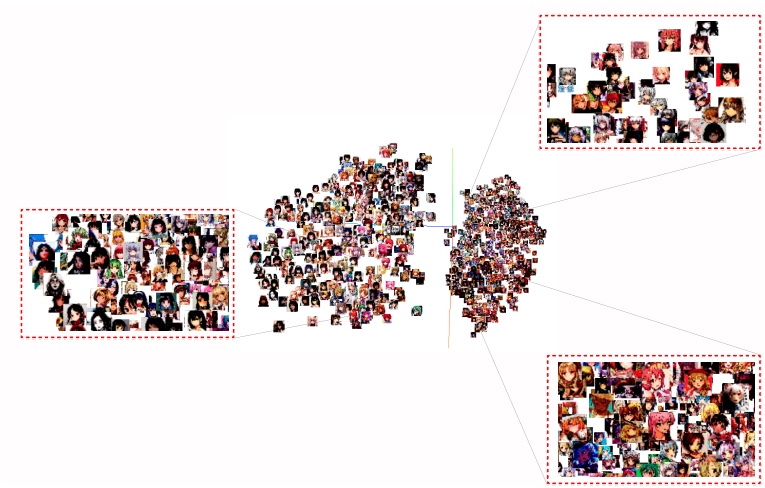

Fig. 12. Female dataset on hypersphere

tion and inpainting models.

8 Machine Learning

8.1 Generative Adversarial Network

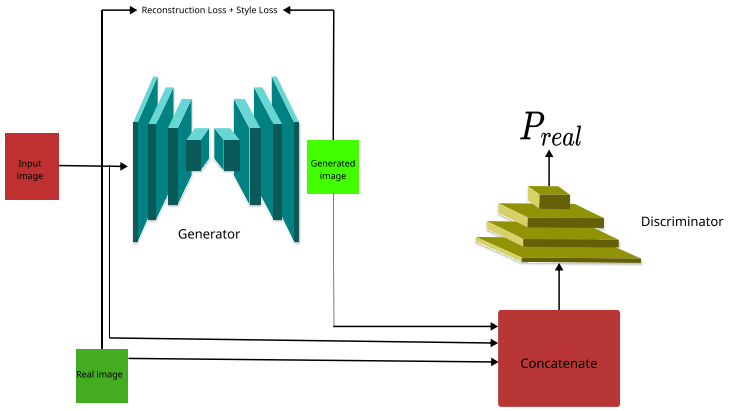

Fig. 13. Training pipeline.

Architecture. The problems we are tackling are related to image translation. U-net is a simple yet stable architecture. Moreover, it can be used for various image translation tasks. Therefore, we could use a U-net-like architecture for this model, as is done by Isola et al. [4], which uses a U-net with a PatchGAN discriminator, which penalizes textures of the output that are too different from those in ground truths. Overall, the output of PatchGAN shows how real a generated image looks giving probabilities patch-wise. However, we made little modifications. We came up with the framework shown in Fig. 13 which includes a reconstruction loss and style transfer loss. The style loss encourages some details, such as texture or color information, to be transferred to generated images. The style-transfer loss can be written as follows

$$G_{ij} = \sum_k F_{ik} F_{jk}, \tag{1}$$

$$L_{style} = \frac{1}{4N^2M^2} \times (G_{ij} - A_i j)^2 \tag{2}$$

where G_{ij} is the gram-matrix for the generated image, A_{ij} is the gram-matrix for the ground truth maps image, N is the amount of channels, M is the product of the height and width of the images and F is an image. In the original style transfer paper, this style loss is calculated by adding up the activation of all the layers. However, we suffice with taking the style loss of the generated images directly. Finally, the reconstruction loss, pixel-wise loss, and PatchGAN discriminator loss are defined as follows:

$$L_{pixel-wise} = \begin{cases} 0.5 \times \frac{(I-P)^2}{beta}, & \text{if } |I-P| < beta \\ |I-P| - 0.5 \times beta, & \text{otherwise} \end{cases}, \tag{3}$$

$$L_{PatchGAN} = (I-P)^2 \tag{4}$$

where I is a real image and P is a predicted image, whereas $beta$ is a hyperparameter that is commonly set to 1.0.

Sketch to BW. The very first stage of drawing an anime character is sketching it. Therefore, one of the tasks we are tackling is to convert sketch images into Black&White images, after which users can colorize them. Thus, the input to this model is sketch maps, as described in **Image Data Processing**, whereas the output is a corresponding Black&White image. The process is summarized in Fig. 14. This feature is especially useful providing a user needs to change a character in the edge level but preserve the overall style of the character.

Colorization. As we obtain a B& W image our next Image-to-Image translation task becomes colorization, meaning we want to translate our B&W image into a color image. However, it is important to provide a full control of this process by allowing users to leave some color marks using which a model can colorize

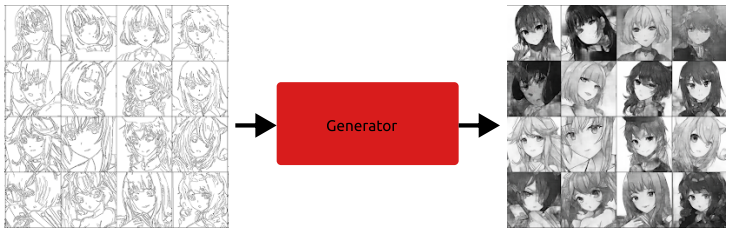

Fig. 14. Sketch map to B&W image model.

the given image. Therefore, the input to this model is a B&W image that has color marks (dots) on specific areas, whereas the desired output is the image with the colorized areas. The process is demonstrated in Fig.15. We believe this is the most convenient way that puts no restrictions on color ranges or areas to colorize.

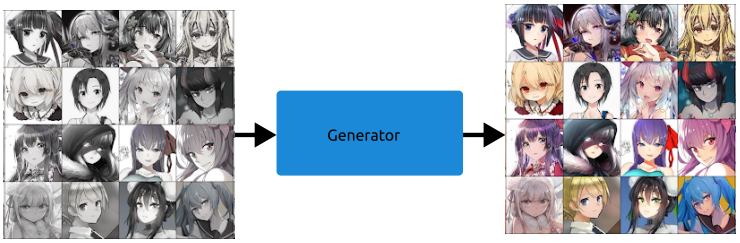

Fig. 15. Marked image to color image conversion.

Inpainting. The last but not the least Image-to-Image translation task we are tackling is image inpainting. This feature becomes desirable providing a user needs to make quick and small adjustments, especially in the edge level, on Color or B&W images without going back to the Edge map. The small modifications usually include erasing certain parts or changing edge information, for example a character's mouth. Therefore, the input to this model is partially erased or modified images, whereas the output is inpainted images. The process is shown in the Fig. 16.

Hyperparameter settings. All the models were trained for 500 epochs with Adam optimizer. The learning rate was set to 0.0002, whereas $beta1$ and $beta2$ were set to 0.5 and 0.999, respectively. The batch size was set to 64 and the images were resized to 256 by 256.

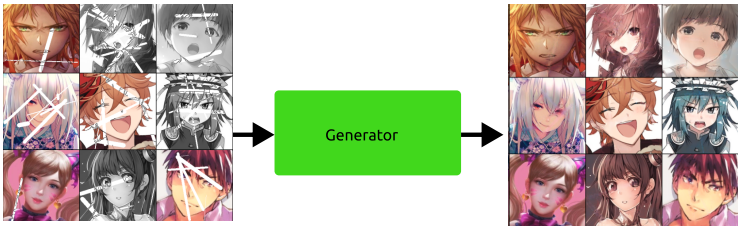

Fig. 16. Inpainting of partially erased or modified image.

9 Evaluation & Use Case Scenarios

9.1 Common usage

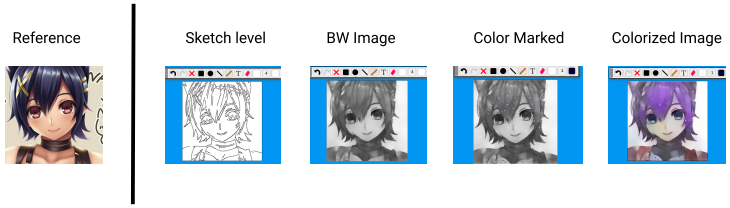

Fig. 17. Work flow.

The Fig.17 shows how users can edit an imported image. First, a user can obtain a sketch image (edge map) and modify it as desired, as we can see we erased the background. The sketch image is converted to B&W image, using the appropriate model, after which it can be marked with color dots. We provide a huge range of colors that can be used to colorize a character. As the image is marked, it can be colorized using our Colorization model. Next, sometimes we may wish to quickly edit an imported image. Thereby, our Inpainting model can handle the adjustments by inpainting erased or modified regions with apropriate colors. We believe it is a very convenient and quick way to make small adjustments on images as we see in Fig.18.

9.2 Editing real drawings

The Fig.19 shows that the users can import their drawings and edit them on our web app. It brings a fresh and new perspectives on their old drawings. Moreover, if a user wants to draw directly on our web app, we provide all necessary tools and features to do that. We demonstrate it on our demo video uploaded on

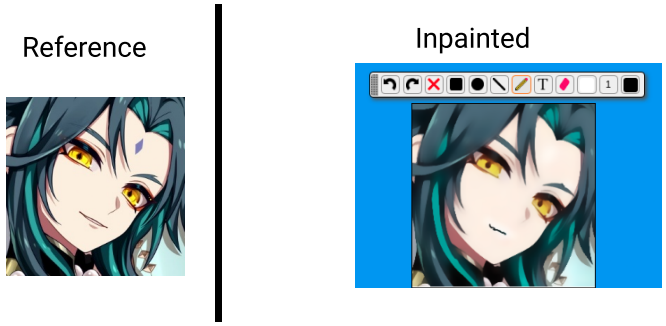

Fig. 18. Image inpainting.

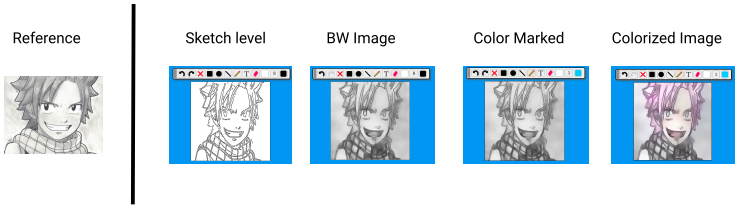

Fig. 19. Editing real drawings.

YouTube[7]. Additionally, we show how Inpainting model can be used to quickly handle imperfections.

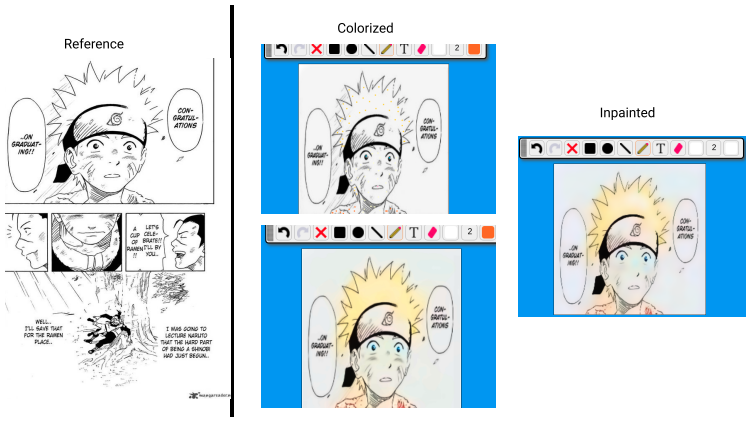

Fig. 20. Editing mangas.

[7] https://www.youtube.com/watch?v=BAOXr-gBciI

9.3 Manga editing

Finally, the Fig.20 shows that our web app can also be used to edit Mangas. We believe our web app provides all necessary tools and features to make this process convenient and user-friendly.

10 Milestone & Team Cooperation

Table 2 depicts a plan of our work. The roles are divided and shown under the *Person* column, whereas the latter column reveals all the subtasks described above. Finally, we are following Scrum Methodology, which is an agile develop-

Month	Subtask	Person
Oct.	Image Data Collection	Chingis, Soohun
	Processing	Chingis, Soohun
	Creation and Design of Splash Screen	Eunmin
	GenerateSketch API	Gone He
Nov.	Train Generation Model	Chingis
	Colorization Model	Chingis
	Creation and Design of Screens	Eunmin, Gong He
	Adding Editing Tools	Eunmin, Gong He
	Colorize API	Soohun
	History API	Soohun
Dec.	Testing	All
	Bug Fixing	All

Table 2. An approximate plan of our team.

ment methodology based on an iterative and incremental processes. Thus, we also had weekly meetings to discuss the progress as well as other issues, including subtasks, ideas and difficulties, via Google Meet[8].

11 Limitations

There are some limitations that should be mentioned.
User related limitations:

- We assume users work with anime characters only.
- We assume imported images contain only the head region (not full body).

Technical limitations:

- We have 3 AI models each being at least 250 MB. Therefore, we did not fine a service that would allow us to deploy them for free.

[8] https://meet.google.com/

– We wanted to include a model that would conditionally generate anime characters. We tried multiple architectures, including DRAGAN, infoGAN, CGAN and PROGAN. However, the problem is probably within the dataset. We believe that the issue is that our images have different head positions and styles in general, whereas human face datasets are highly curated, especially in terms of head positions. However, we simply did not have enough time to work on our dataset even more.

12 Conclusion

Thus, with the development of technology, anime is still growing massively as a business area, and there is a huge demand and potential. There are many related projects but they are limited in terms of control provided to users. Also, current datasets have an imbalance problem in terms of female and male characters, making models biased towards female character generation. Inspired by the observations, our team, EDITH, wants to provide a full control of anime character editing process by allowing users to sketch and create their own characters in end-to-end manner. We created a web app with a user-friendly interface, where users can freely edit images, and provide a set of useful GAN-based features.

References

1. Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.
2. Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.
3. 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*, pages 2672–2680, 2014.
4. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv preprint arXiv:1611.07004*, 2018.
5. Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.
6. Mark Serrels, CNET Crunchyroll and Funimation merger: What does it mean and what happens next?, https://www.cnet.com/news/crunchyroll-and-funimation-merger-what-does-it-mean-and-what-happens-next/. Last accessed 29 Sep 2021
7. Report Overview, https://www.grandviewresearch.com/industry-analysis/anime-market. Last accessed 29 Sep 2021
8. Otaku Coin Cryptocurrency Wants to Create an Isekai Anime with NFTs https://www.animenewsnetwork.com/interest/2021-09-10/otaku-coin-cryptocurrency-wants-to-create-an-isekai-anime-with-nfts/.177170. Last accessed 29 Sep 2021
9. Make Girls Moe Homepage, https://make.girls.moe/. Last accessed 29 Sep 2021
10. Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. How to train your dragan. arXiv preprint arXiv:1705.07215, 2017.

11. Masaki Saito and Yusuke Matsui. Illustration2vec: a semantic vector representation of illustrations. In SIGGRAPH Asia 2015 Technical Briefs, page 5. ACM, 2015.
12. Kaggle https://www.kaggle.com/scribbless/another-anime-face-dataset
13. Safebooru https://safebooru.org/
14. Github, Anime-face-detector https://github.com/qhgz2013/anime-face-detector