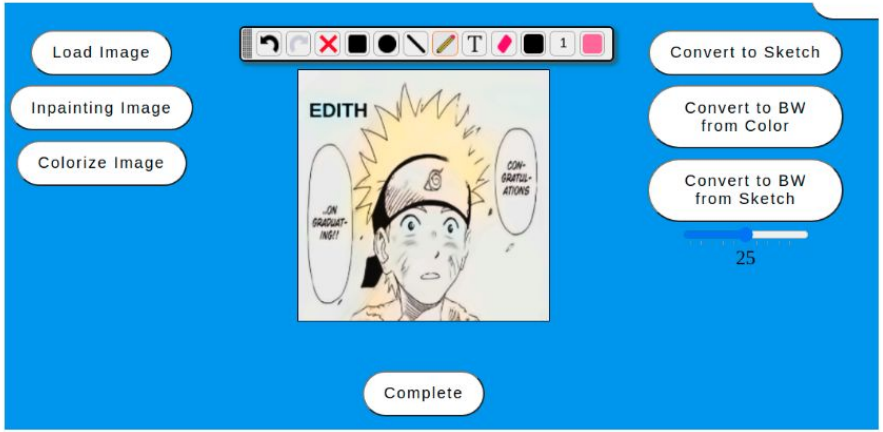
EDITH Final Presentation

Team D
Chingis Oinar
Eunmin Kim
Gong He
Soohun Park

https://github.com/chingisooinar/CDP_EDITH

README.md

AI-Powered Anime Photo Editing Web App



Members

- **Machine Learning**: 칭기스
- **Backend Developer:** 박수헌, 공허
- **Frontend Developer**: 김은민, 공허

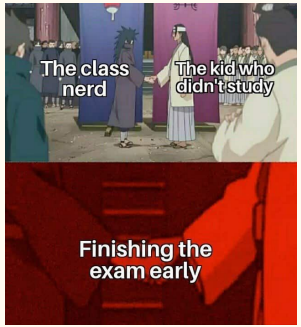
Objective

Create an AI Powered end-to-end anime style photo editor web app

Allow user to have full on control over how they want their final outcome to be

Implement many features that help user edit their photos

- Inpainting
- Colorize
- Convert from Image to Sketch
- Convert from Color to B&W
- Convert from Sketch to B&W



Motivation

Wanted to use GAN for our project

Create a useful editing app for fans of anime-styled images

Possibility of creating NFTs

Project Schedule/Progress

	September	October	November	December
Chingis	Preparing Dataset	Creation & training of models	Creation & training of models	Creation & training of models
Soohun	Preprocessing & Preparing Dataset		Developed APIs	Bug Fixing
Gong He	Start and set up Django project	Frontend layout and backend apis without models	Frontend and backend communication	Bug Fixing
Eunmin	Design of Screen	Frontend styling	Frontend styling and some backend frontend communication	Bug Fixing

Role of Each Member

Chingis: Processing, training GANs, creating/cleaning datasets

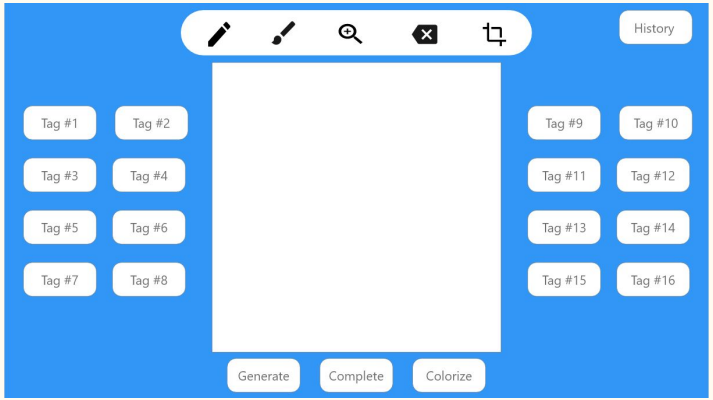
Soohun: Image data collection, Preprocessing & cleaning, creating all APIs

Gong He: Connection between frontend and backend and skeleton of frontend

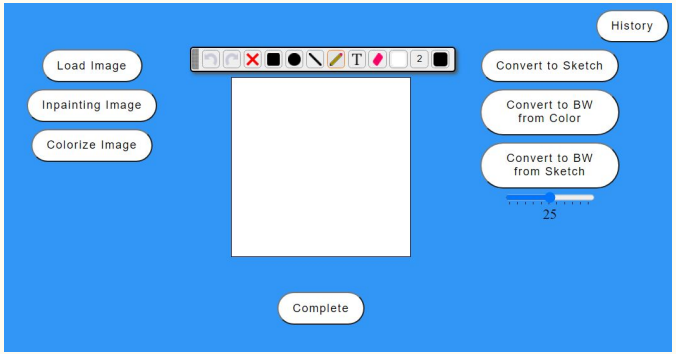
Eunmin: Design of screens, css and layout of frontend

Final Design 1

Design

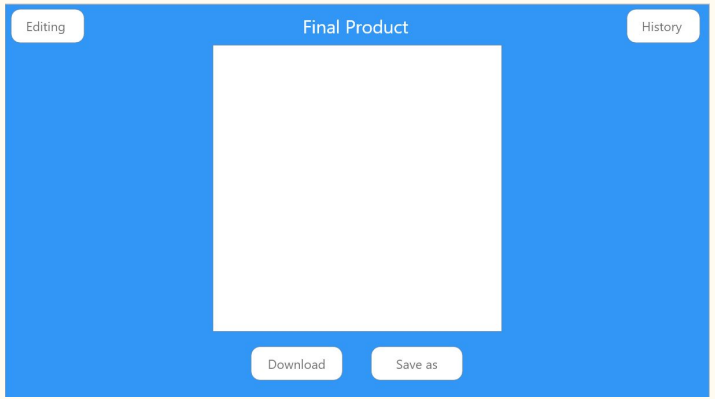


Actual



Final Design 2

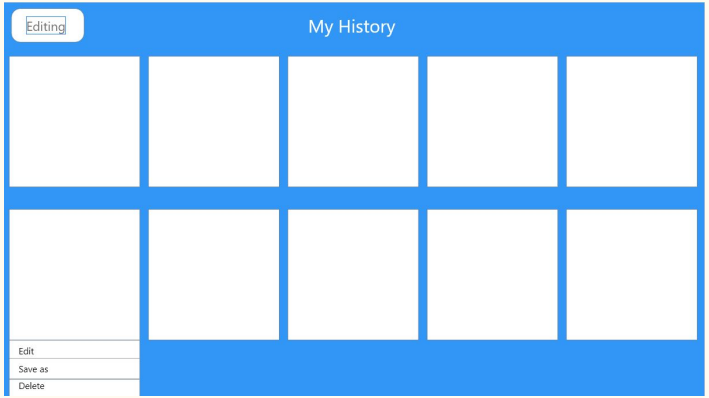
Design



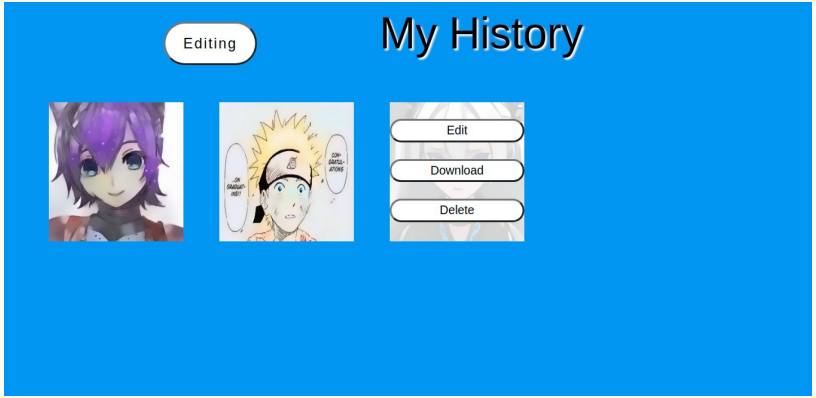
Actual

Final Design 3

Design



Actual



Final Implementation

Frontend: Jquery

Backend: Django

ML/AI: Pytorch

Programming Languages: Python, Javascript, HTML, CSS

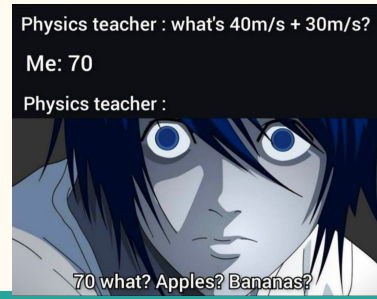
Pages

- Index page
  - Get: Index page with empty canvas.
  - Post: Index page with canvas which loads post image data.
- Result page
  - Get: Jump to index page.
  - Post: Result page with complete image.
- History page
  - Get/Post: History page.

Session

- Use Django session engine.
- Save session data in sqlite database.
- Cookie name: “sessionid” (“Cookie”: “sessionid=xxxxx”)

```
Connection: keep-alive
Cookie: sessionid=v9uicydo0hn4nhvzo3hau6xia9tb3kur
```



History logic

1. New user(no session set) ?

   In index page: Give him a new id(microsecond timestamp) by session

   In history or result page: Jump to index page.

2. Check if user's history folder(named user's id) exists

   If not, make dir

3. History files will be saved in [user id] folder

To all brethren that have to go back to school or back to work

Connect Frontend and Backend

- jQuery Ajax
- FormData

1. When upload a binary image, it will be in FILES field.

   When upload a base64 String image, it will be in POST field.

2. Every api will return base64 String image and frontend will put base64 image to canvas.

APIs

- Inpainting_api
- Colorize_api
- Convert_to_sketch_api
- Convert_to_bw_api
- Convert_edge_to_bw_api

- Complete: upload and store complete result as user's history
- DeleteHistory
- UploadResize

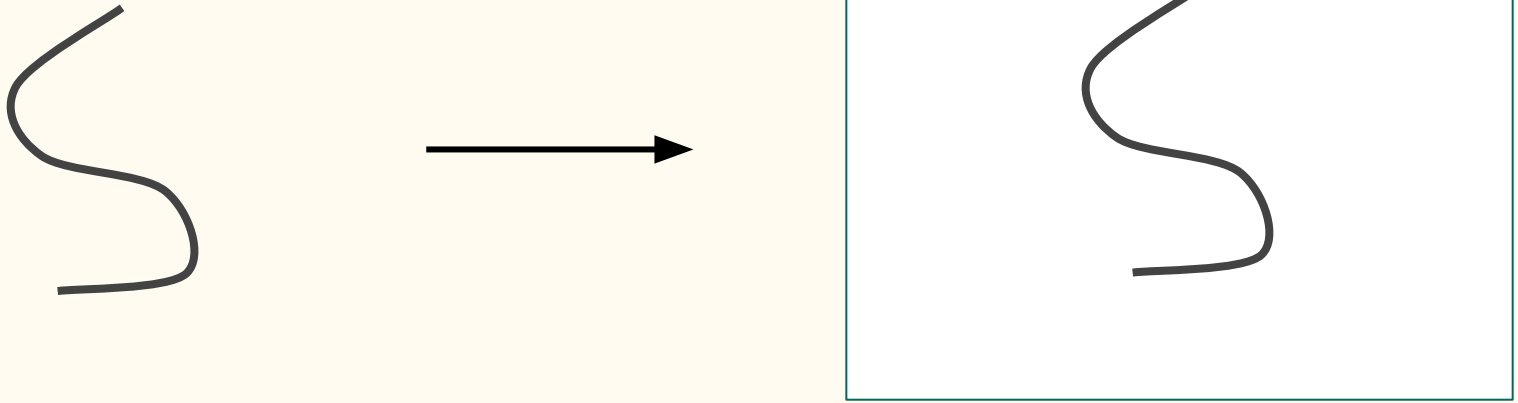
APIs

	Inpainting	Colorize	Convert to Sketch	Convert to BW	Convert edge to BW
Request	Color Image w/ marks	BW Image w/ color dots	Image	Colored Image	Sketch (only edges)
Response	Inpainted Image	Colorized Image	Sketch (Only edges)	BW Image	Image Colorized in BW

In base64 string image format

Prepocessing in APIS

Adding background



```
def transparence2white(img):

    for yh in range(256):
        for xw in range(256):
            color_d=img[xw,yh]
            if(color_d[3]==0):
                img[xw,yh]=[255,255,255,255]
    return img
```

Processing in APIS

Preprocessing

: applied augmentations for pre processing

```
# DEFINE TRANFORMATIONS FOR PREPROCESSING
_transforms = transforms.Compose(
    [
        transforms.Resize((256, 256)),
        transforms.ToTensor(),
        transforms.Normalize(
            [0.5 for _ in range(3)], [0.5 for _ in range(3)]),
    ]
)
```

....

```
inp = _transforms(image)
```

Postprocessing

: removed noises

```
# post processing
generated_image = (generated_image * 127.5 + 127.5).astype('uint8')
# remove noise
denoised = cv2.bilateralFilter(generated_image, 15, 75, 75) # preserves edges
denoised = cv2.cvtColor(denoised, cv2.COLOR_RGB2GRAY)
denoised = cv2.cvtColor(denoised, cv2.COLOR_GRAY2RGB)
# make edges stronger
if edges_weight > 0:
    edges_contour = (edges / 255.0) - 1
    edges_contour[edges_contour == -1] = 1
    denoised = denoised - edges_weight * edges_contour
    denoised[denoised < 0] = 0
```

```
# post process
img =  (np_image * 127.5 + 127.5).astype('uint8')
# remove nois( salt&pepper noise)
dst = cv2.bilateralFilter(img, 15, 75, 75)
if is_gray:
    dst = cv2.cvtColor(dst, cv2.COLOR_RGB2GRAY)
    dst = cv2.cvtColor(dst, cv2.COLOR_GRAY2RGB)
return dst#, img
```

Dataset

Female Dataset : kaggle(https://www.kaggle.com/scribbless/another-anime-face-dataset)

Male Dataset : Crawling (safebooru.org) → Over 50k images

Problem : Dataset Cleaning

Tried : Pseudo labelling & Metric Learning(mined KNNs)

!!! STILL IT WAS NOT ENOUGH !!!

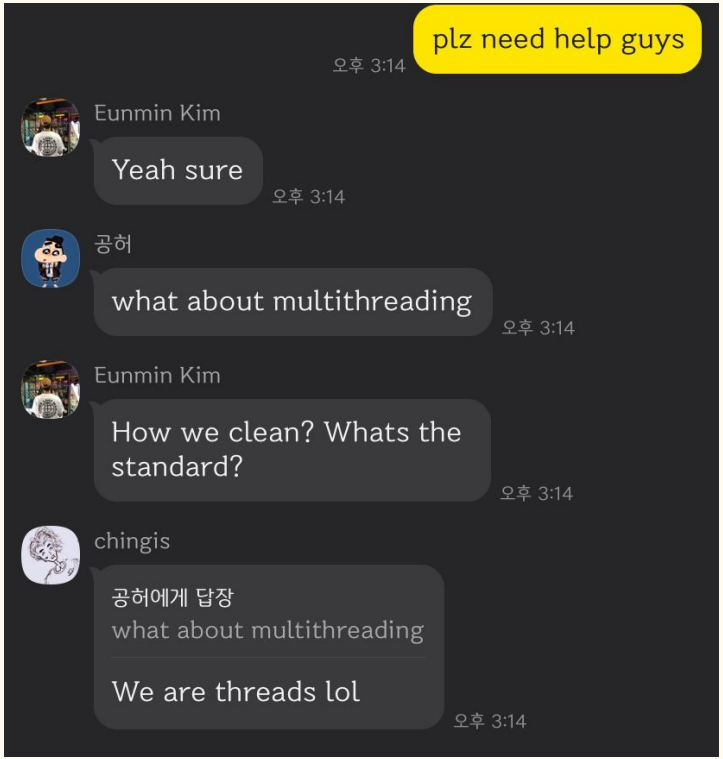
Problems in Dataset

Included

1. Girls
2. Too violent characters(with blood)
3. Slight side views
4. Black & White characters
5. Characters with too unique drawing styles
6. …….

MANUALLY...

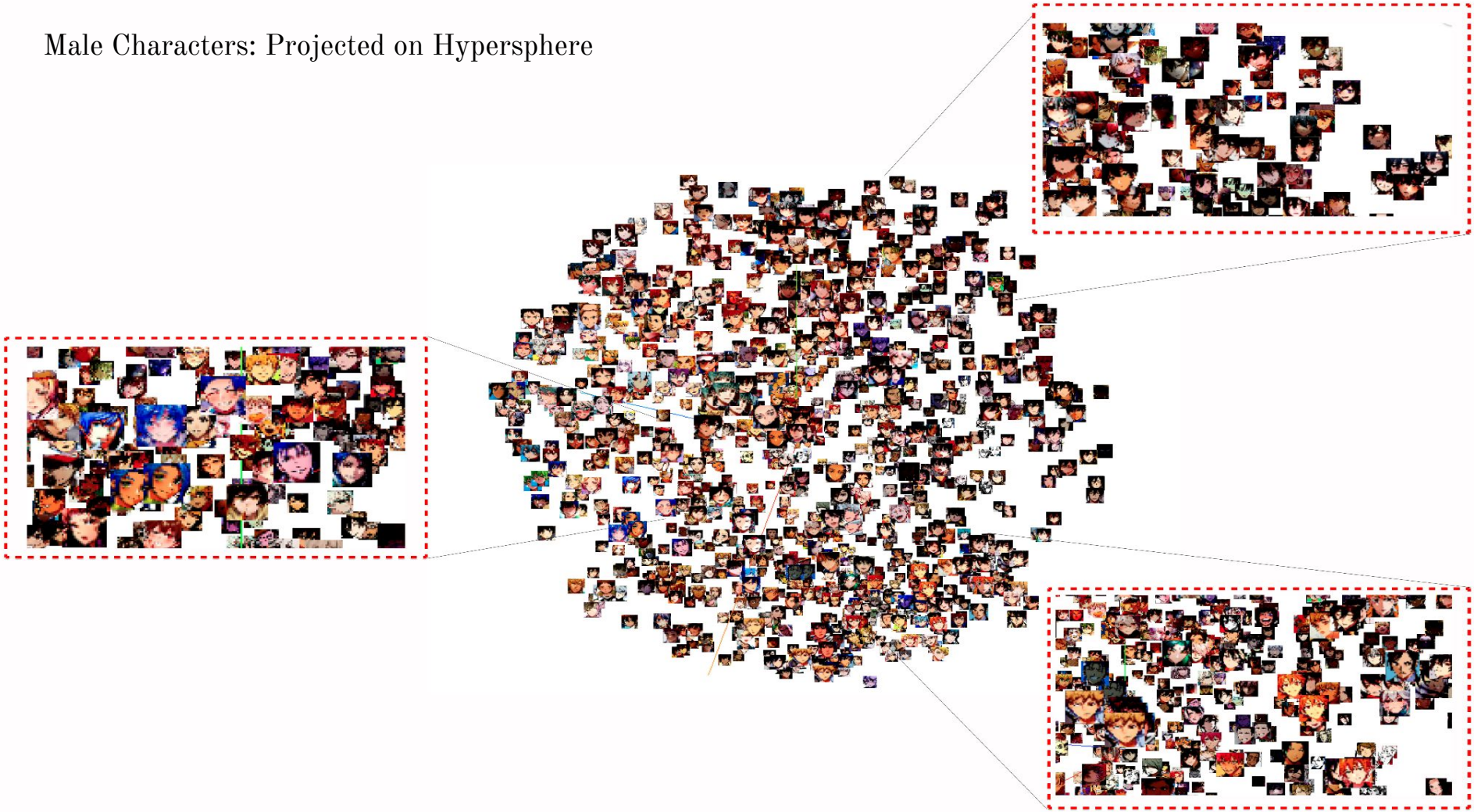
Dataset Cleaning



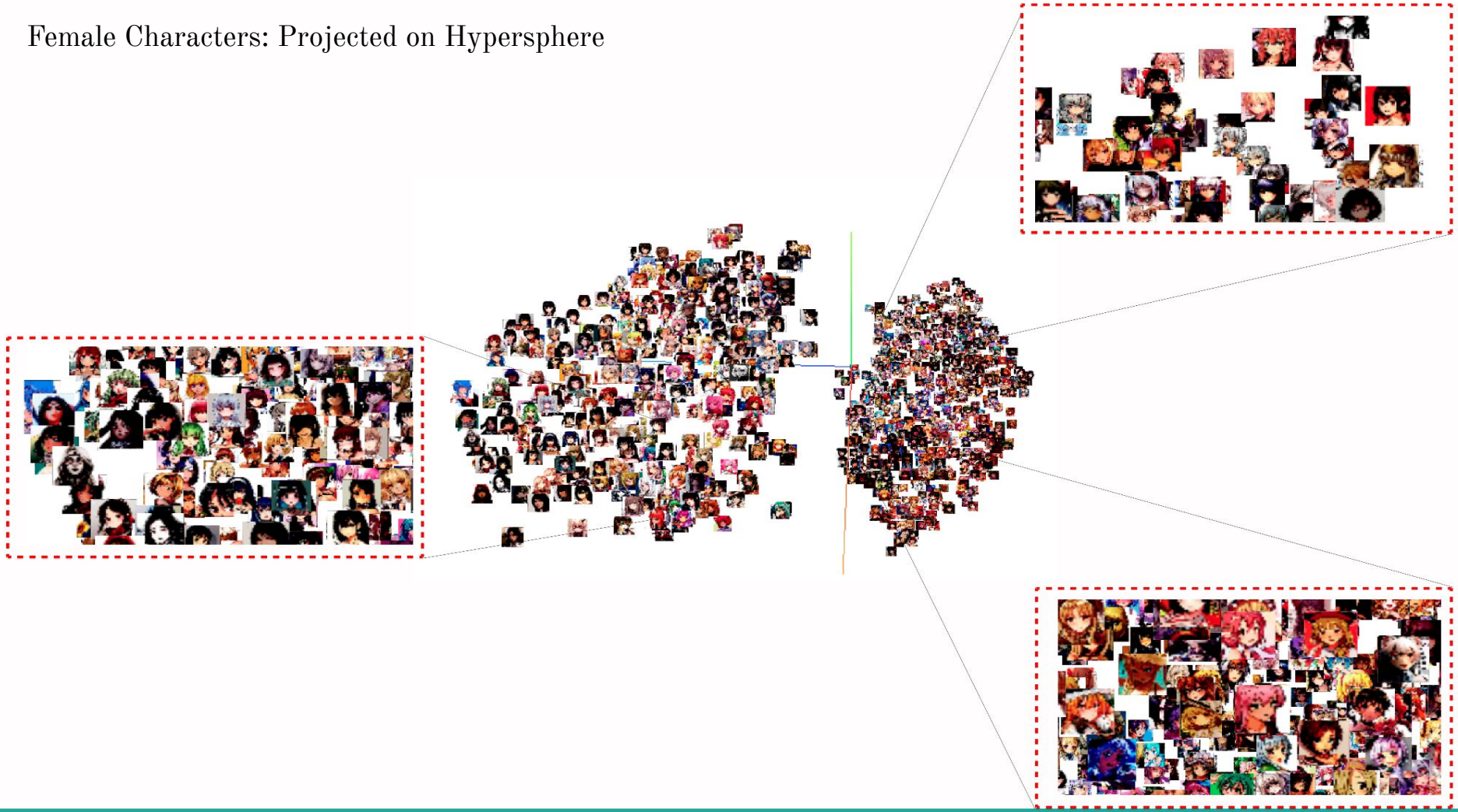
Cleaned dataset manually

Finally a satisfactory dataset was obtained

Male Characters: Projected on Hypersphere

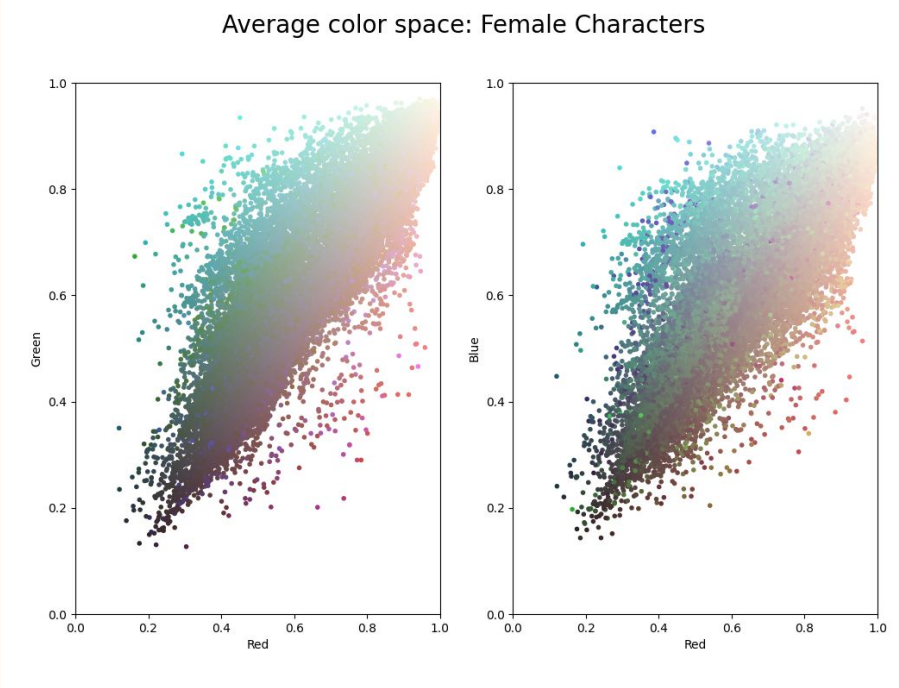


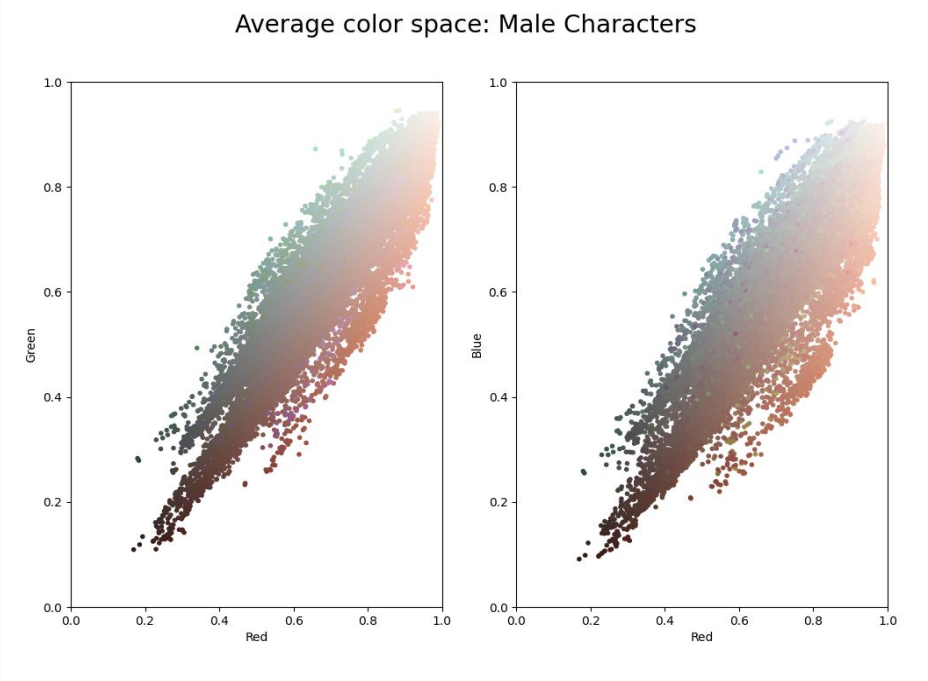
Female Characters: Projected on Hypersphere



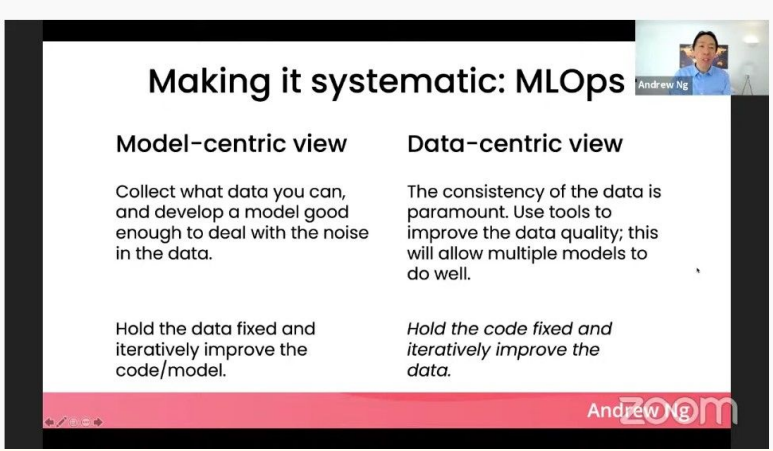
Wider Range of Colors for Colorization and Inpainting

Range of Female characters + Range of Male Characters





ML Issues: Generation



We tried different architectures including DCGAN, DRAGAN, InfoGAN and PROGAN. However, they were not stable.

Possible reasons:

Human Face datasets are highly curated meaning they have somewhat same head positions.

Issues: Generation

Issues: Generation

Due to time limitations and busy schedule, we decided to replace it with a different interesting and useful feature for image editing instead of spending more time on dataset cleaning and data collection.

ML/DL

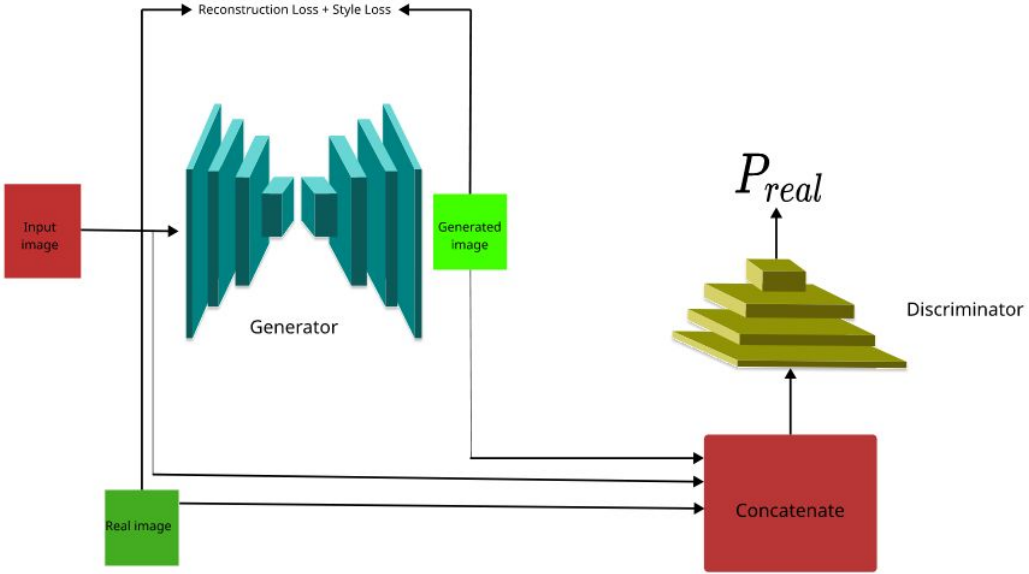
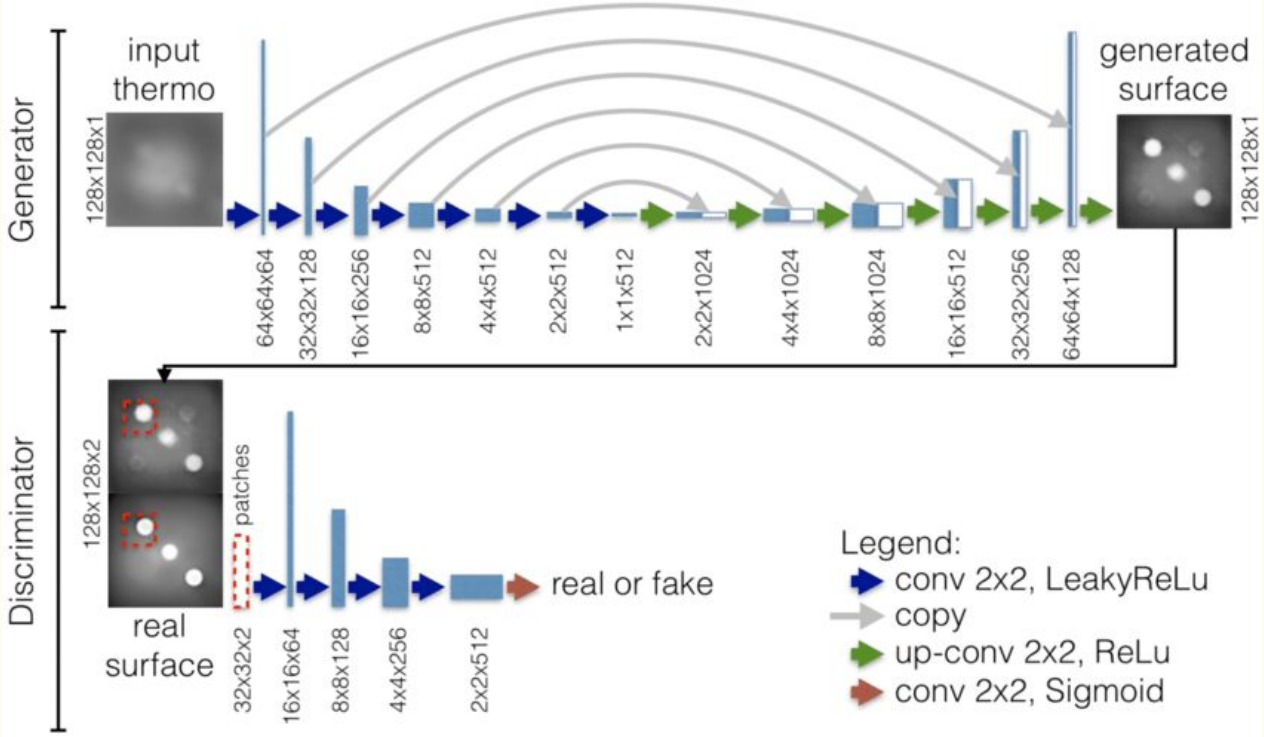


Fig. 2. Training pipeline.

Architecture



Objective

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

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

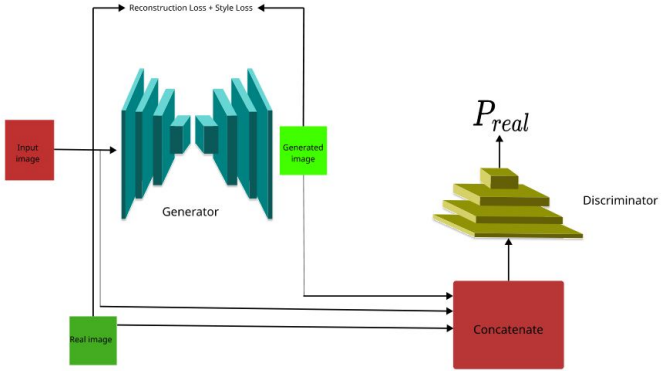


Fig. 2. Training pipeline.

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

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

Hyperparameters

Epochs: 500

Learning rate: 0.0002

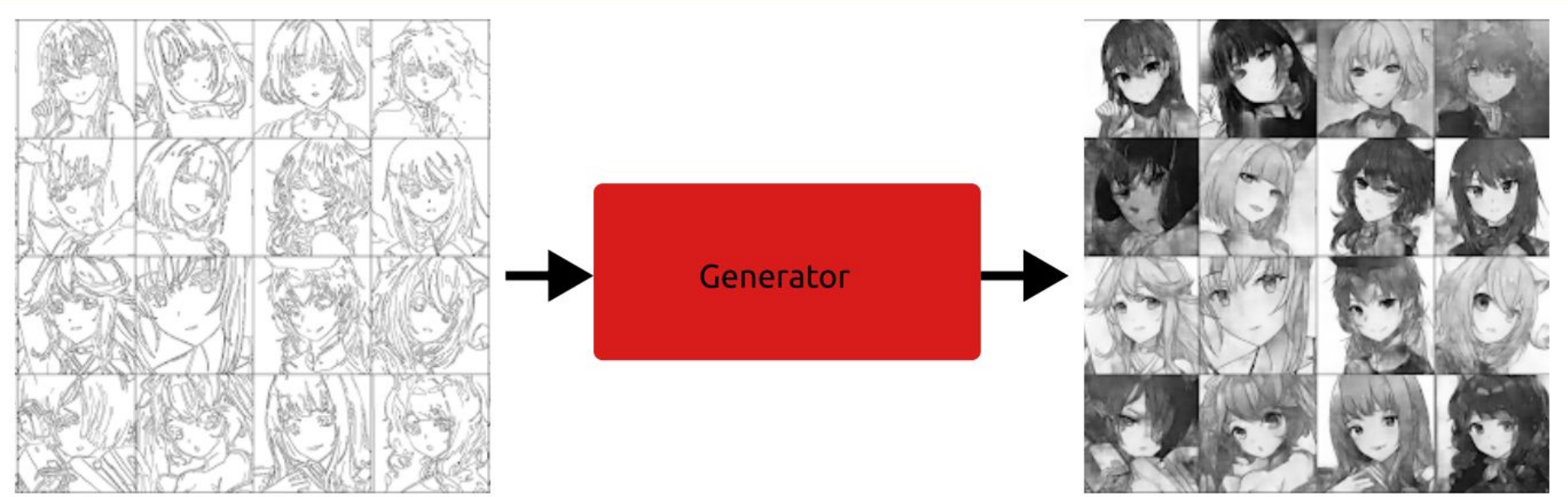
Optimizer: Adam with beta1 = 0.5 and beta2 = 0.999

Batch size = 64

Image size = 256

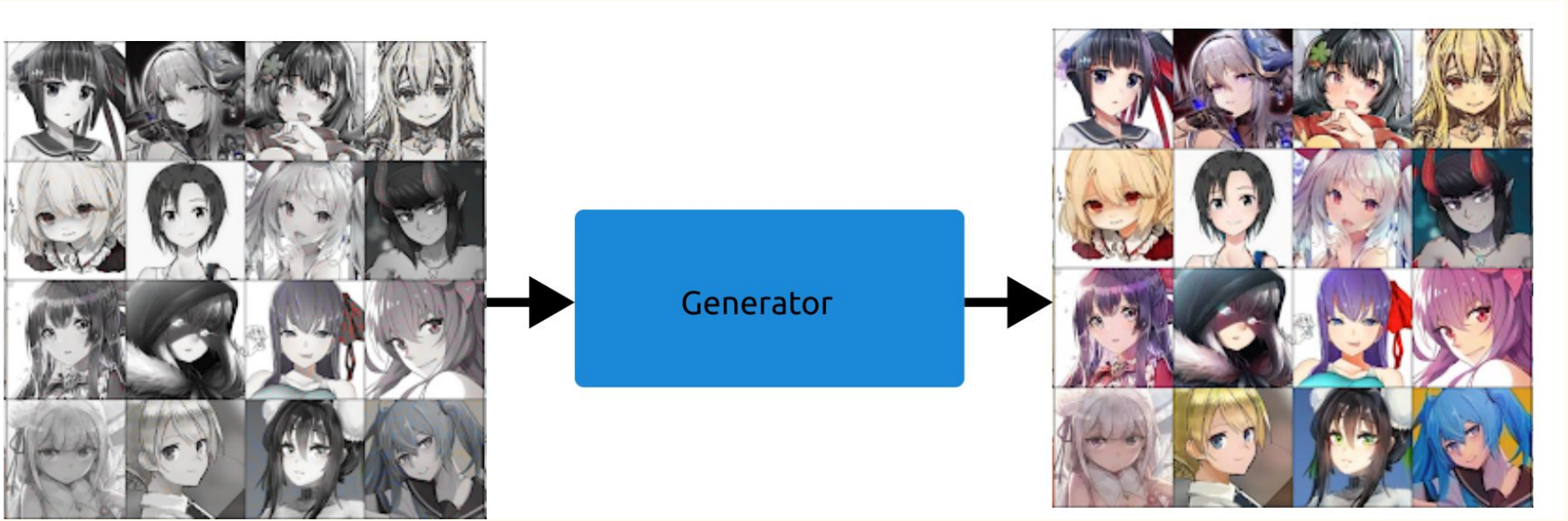
Sketch to BW

Sketch to BW



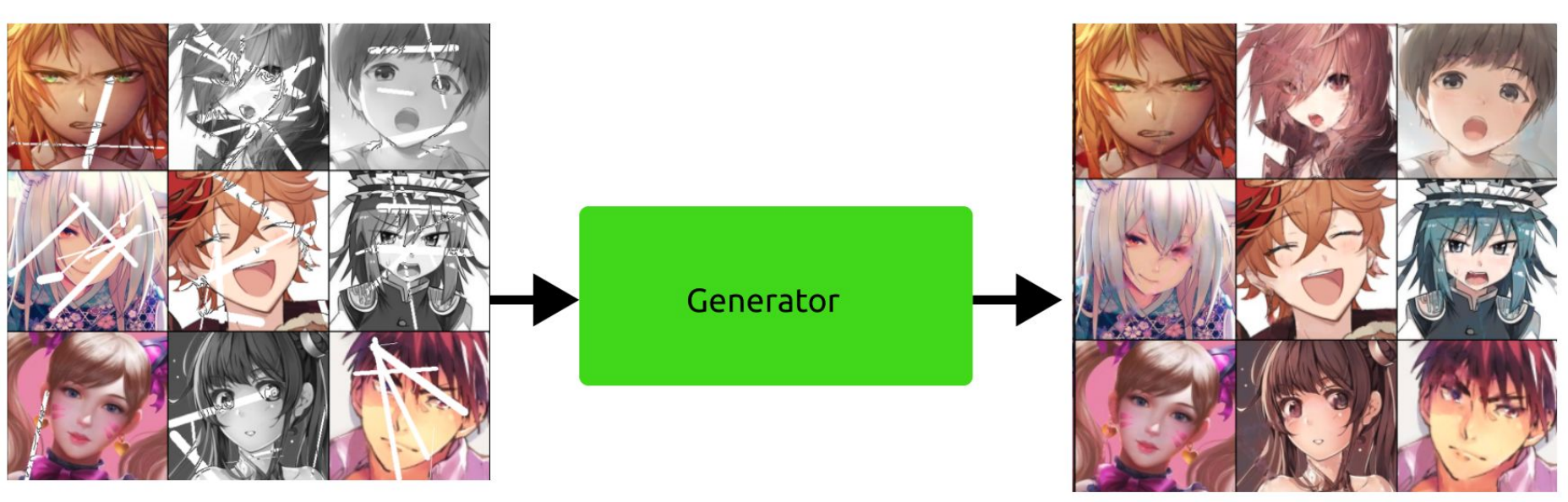
Colorization

Colorization



Inpainting

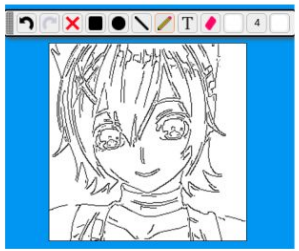
Inpainting



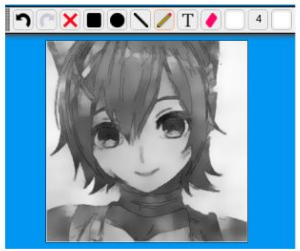
Use Case Scenarios

Reference

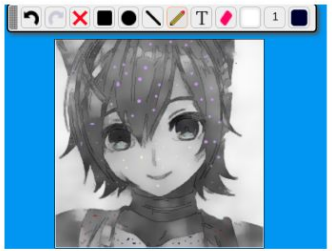
Sketch level



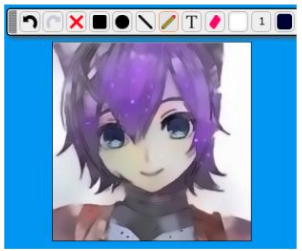
BW Image



Color Marked



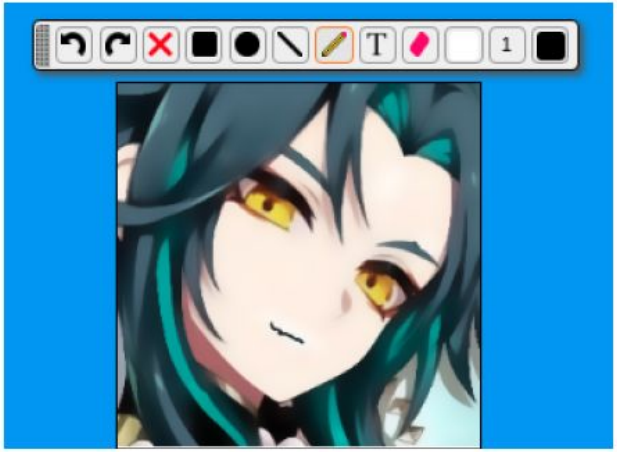
Colorized Image



Use Case: Inpainting with Color Images

Reference

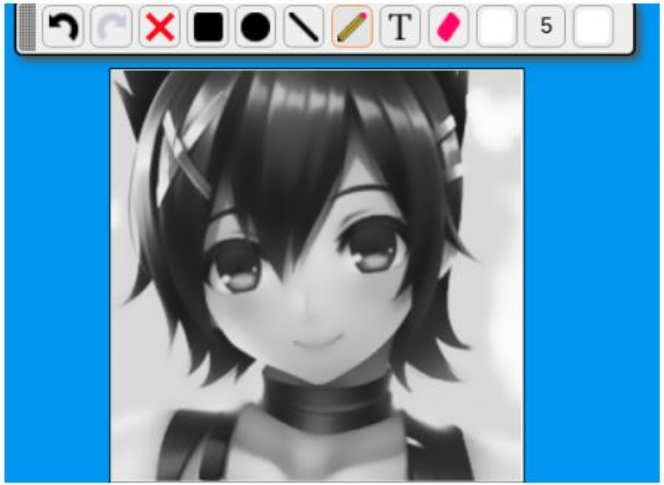
Inpainted



Use Case: Inpainting with BW Images

Reference

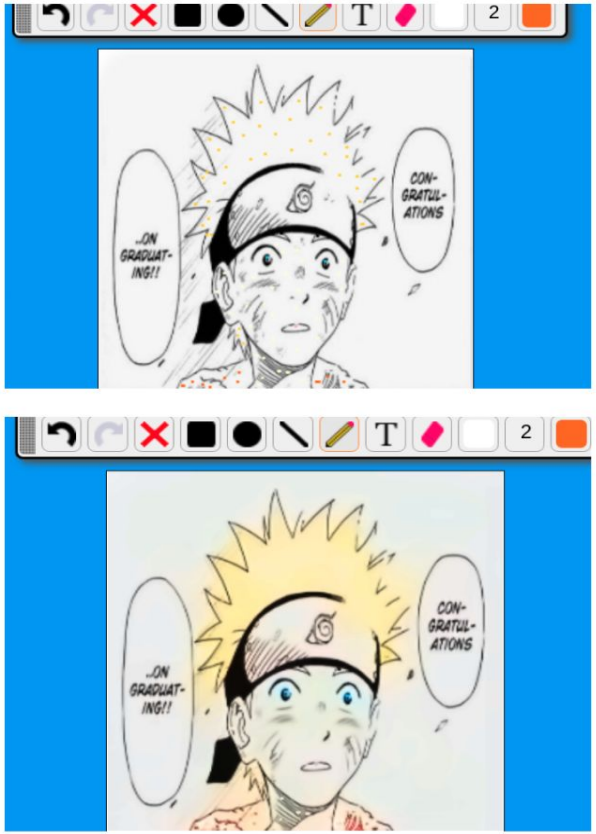
Inpainted



Use case: Manga Editing

Reference

Colorized

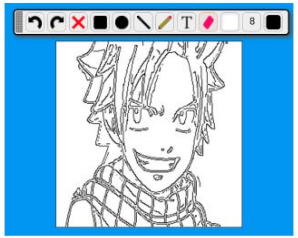


Inpainted

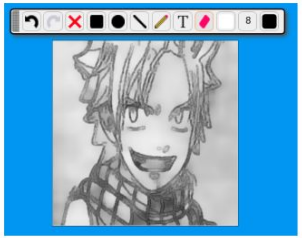
Use case: Importing real drawings

Reference

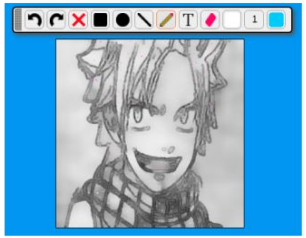
Sketch level



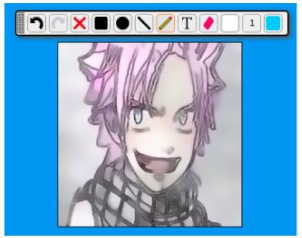
BW Image



Color Marked



Colorized Image



Limitations

- Constraints/Assumptions when user sketches:
  - Is a human (Not an animal or inanimate object)
  - Only face + neck, not full body
  - Includes desired key features (Nose, eyes, lips, etc)
- Deployment:
  - We have 3 AI models each being at least 260 MB. We did not find any services that would allow us to deploy them for free.

Demo


History
Load Image
Inpainting Image
Colorize Image
Convert to Sketch
Convert to BW from Sketch
35
Complete