Use of Generative Adversarial Networks (GANs) and LoRA Stable Diffusion for Image Generation

Group 3: Matthew Jamison, James A. Nguyen,

Lawrence Mei

What is a GAN?

GANs are models consisting of two networks:

i. Generator
ii. Discriminator

Role of Generator:
Fool the discriminator

Role of Discriminator:
Correctly identify fake image given one real and one generated input

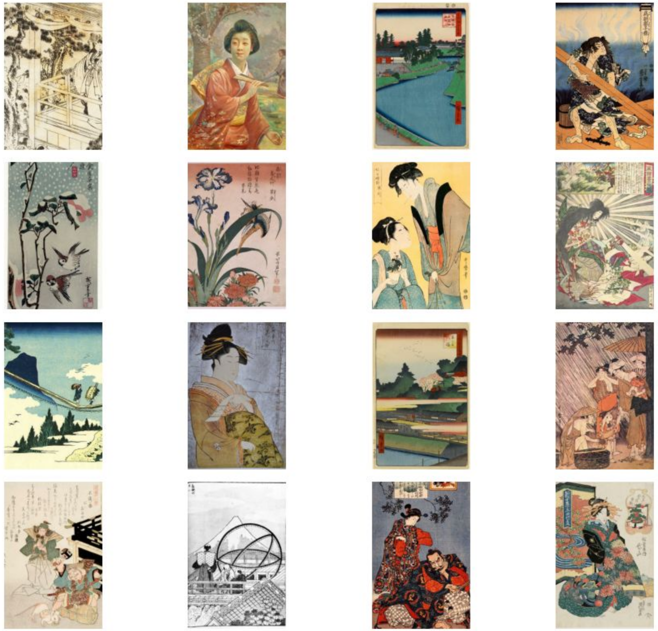
Our Data

2236 -> 1492

Standardized resolution to 480 x 320

Manually isolated images in the style of ukiyo-e edo period style art

Idea: iterate model over random noise until resembles data

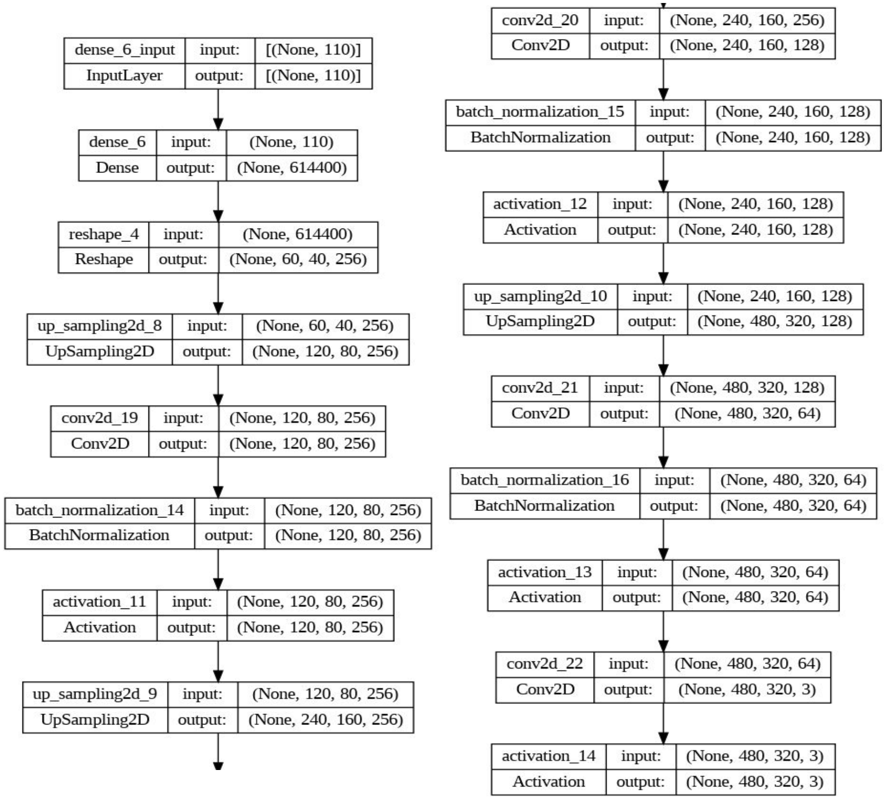


~ 1500 image subset of Edo Period Japanese Art

Our Goal

- Generate images that resemble certain art styles
  - For example: ukiyo-e edo period art, cartoon, caricature, realistic, etc.
- Art restoration
- Create concept art with inspiration from artists

Generator Network Architecture



General Framework:

Upsampling and Batch Normalization in each layer.

ReLU activation function to avoid distortion

Decrease number of neurons with each layer

RMSprop optimizer

Discriminator Network Architecture

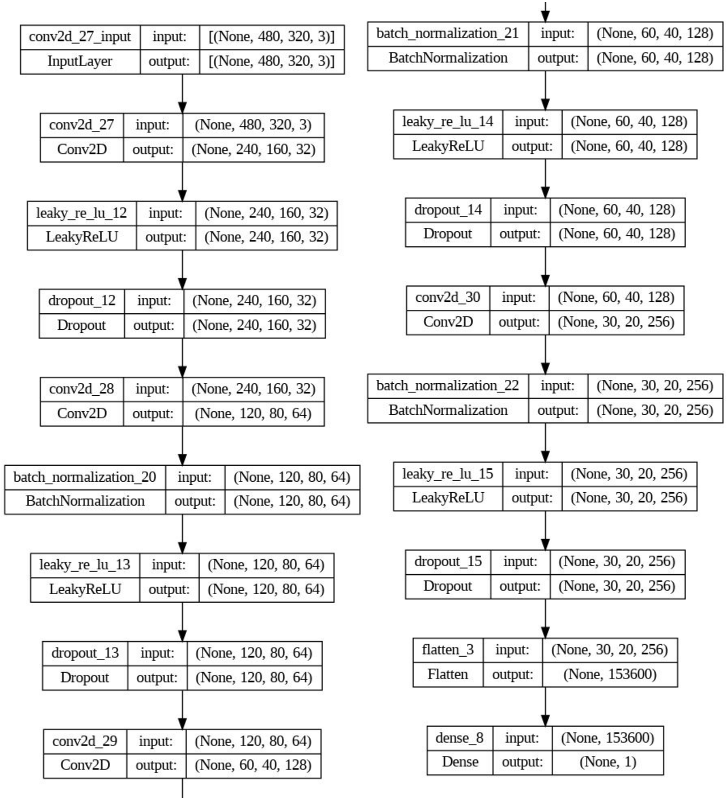
General Framework:

Downsampling and Batch Normalization in each layer.

Leaky ReLU activation function

Increase number of neurons with each layer

RMSprop optimizer



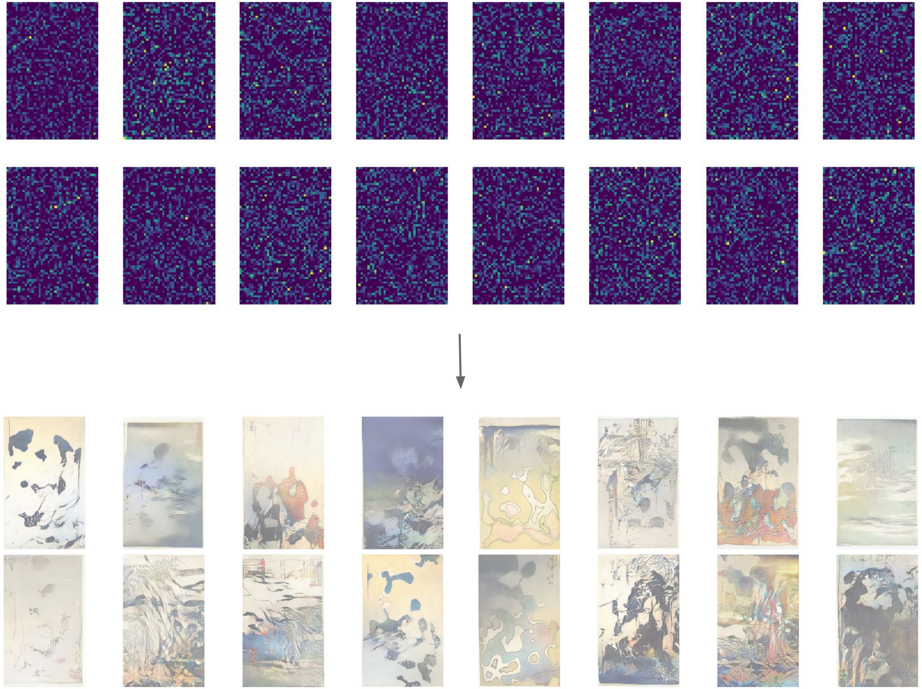
GAN Training Loop

Fundamentally a game theory simulation

Important to balance the learning rates of the models

i. Often involves intentionally nerfing the discriminator to even the playing field

ii. Utilized exponential decay rate

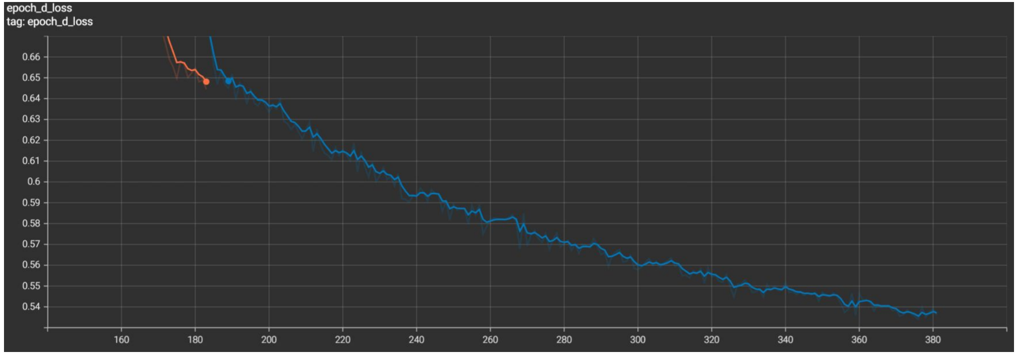


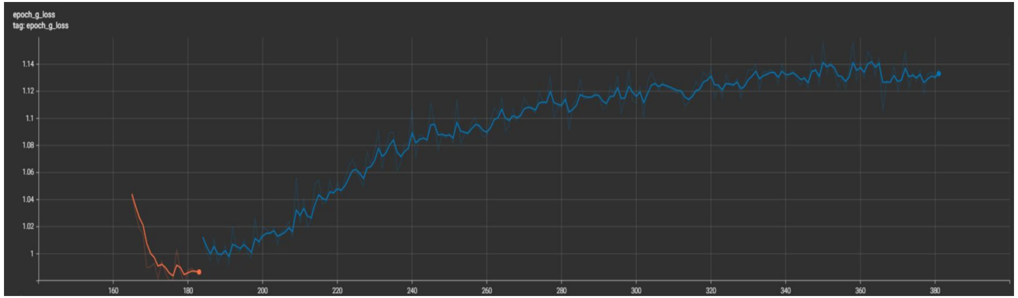
Model Analysis

Only trained on 450 epochs before running out of compute units.

Loss chart suggests that the generator model falls behind after ~ 180 epochs

Could improve with larger dataset and more epochs





Hyperparameter Tuning

Optimizer	RMSprop
Learning Rate	1e-4
Decay Steps	1000
Decay Rate	0.95
Batch Size	16
Epochs	450

RMSprop vs Adam

Equal vs Different Learning Rates

- Early Stage Mode Collapse
- Used Exponential Learning Rate Decay
    - Switch to more adaptive learning rate scheduler

Conservative Batch Size and Epochs due to resource limitations

Model Outputs

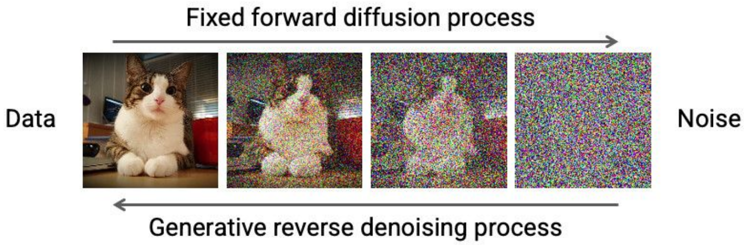
Despite training resource limitations, early training showed signs of

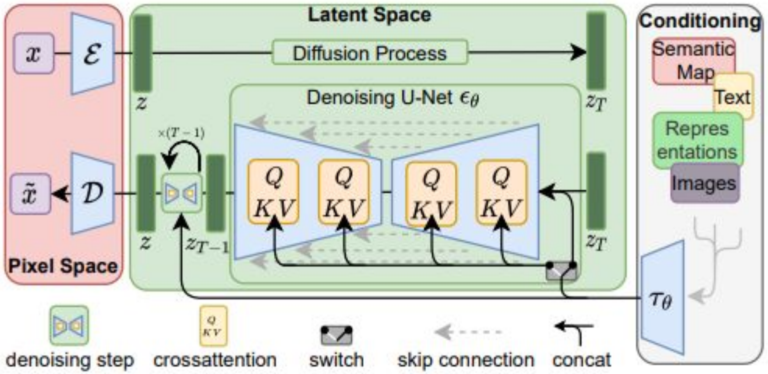
i. shared qualities with the dataset e.g. color similarities

ii. as well as simple feature extraction e.g. red tag commonly found in the top right corner

What is Stable Diffusion?

- We chose SDXL 1.0
- It is a latent diffusion model trained on 2.3 billion images
- Architecture:
    - Encoder/Decoder
    - Diffusion Process
    - Language Transformer
    - U-Net
    - Cross-Attention

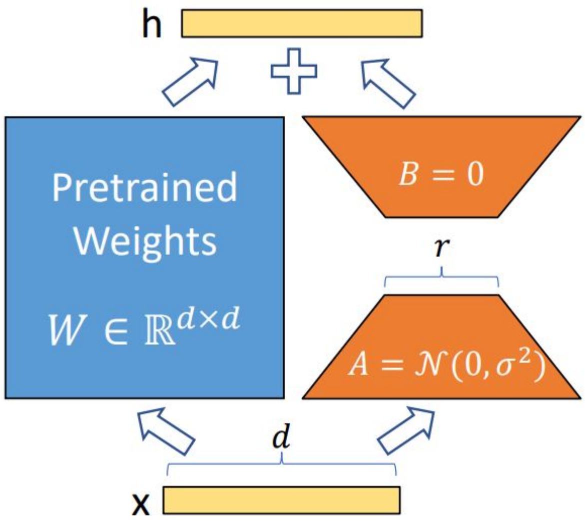




What is LoRA?

Low-Rank Adaptation

- Pretrained model weights are frozen
- Fine-tuning weights created through low-rank decomposition to matrices A and B
- Way less training time!



Data Preparation

- Resized images to 600x400 (GAN was 480x320)
- BLIP model to provide captions for images with some manual tweaking
- Rented GPU time on an RTX A5000 due to memory implications

a cat sitting on a window ledge looking out at a landscape

LoRA Hyperparameter Tuning

Optimizer	Adafactor
Learning Rate	1e-3
Network Rank	128
Batch Size	20
Epochs	12

-Adafactor chosen to save on memory

-Faster learning rate due to LoRA training times

-128 is considered “good enough” for SDXL LoRA training

-12 epochs to compare images

Epoch Comparison

Epoch 7

Prompt: Samurai on a hill with his sword drawn, highly detailed, clouds and ocean

Comparison to Base Model

Base Model <lora:jpn_art_style-000001:1> <lora:jpn_art_style-000006:1> <lora:jpn_art_style-000007:1>

Prompt: Lawson’s store with Mt. Fuji in the background

More Results

Future Works and Improvements

- Wasserstein loss
  - Relationship between y_true and y_false
- Indicators such as frechet inception distance (FID) and inceptions score to evaluate generator quality
- More training data, more epochs
- Higher network rank for LoRA training

Thank you.