ART GAN

Gregory Wong DSI 22

TABLE OF CONTENTS

01

PROJECT INTRODUCTION

The why and how, mostly the why

ABOUT THE PROJECT

This project was inspired by the Netflix film – Made You Look

It covers the story of how the Knoedler Gallery in New York, with over 165 years of history, sold over $80 million worth of fake art

How to sell fake art?

Scammers

Jose Diaz and Glafira Rosales

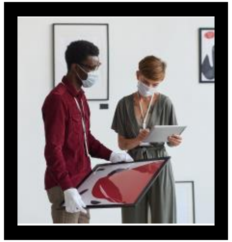
Forger

Pei-Shen Qian

Dealer

Ann Friedman

Appraisal



Collectors

Project Objectives

1. Understand how GANs work and what best practices there are to optimize them

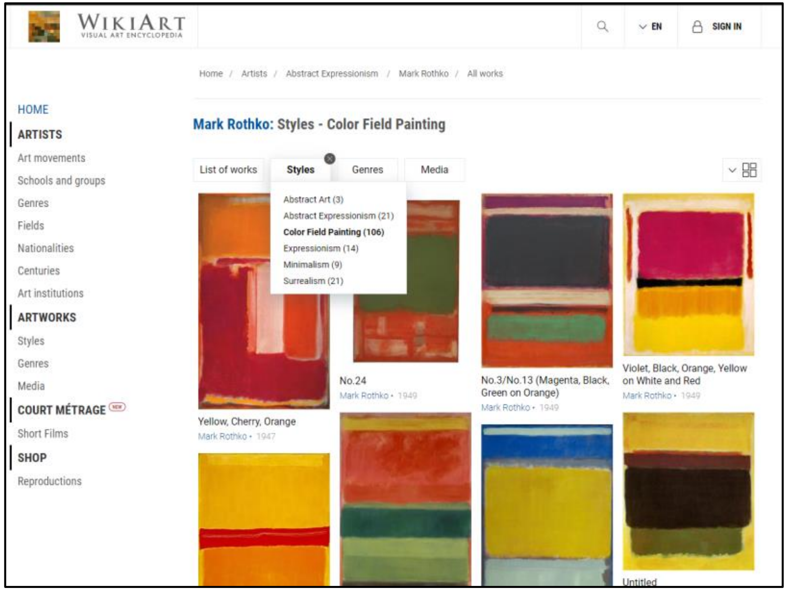
2. Determine if machine learning can help in the creative process

02

DATA GENERATION

Scraping data, cleaning and augmenting

Data Generation



Wikiart was used as the database, specifically color field paintings

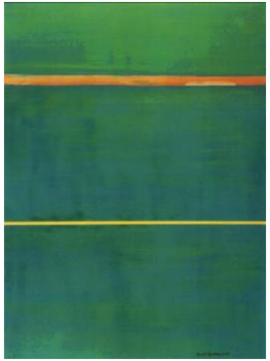
Scraped using Selenium →

	Artist	Num_Works
0	Mark Rothko	106
1	Jack Bush	35
2	Barnet Newman	78
3	Morris Lewis	100
4	Gene Davis	141
5	Ellsworth Kelly	21
6	Frank Stella	47
7	Helen Frankenthaler	65
8	Sam Gilliam	8
9	John Hoyland	32
10	Ray Parker	16
11	Edward Avedisian	6
12	Walter Darby Bannard	9
13	Frank Bowling	2
14	Dan Christensen	15
15	Richard Diebenkorn	17
16	Piero Dorazio	17
17	Friedel Dzubas	23
18	John Ferren	8
19	Sam Francis	74
20	Robert Goodnough	1
21	Paul Jenkins	14
22	Ronnie Landfield	38
23	Pat Lipsky	2
24	Brice Marden	17
25	Robert Motherwell	2
26	Blinky Palermo	4
27	Paul Reed	11
28	Alma Woodsey Thomas	25
29	Larry Zox	6
30	Total	940

30 artists, 940 works

Sample Images

Mark Rothko,
no.24



Barnett Newman,
Dionysius

Gene Davis,
Blue Broad Jump

Morris Louis,
While Series II

Data Cleaning (Round 1)

Ellsworth Kelly,
Spectrum V

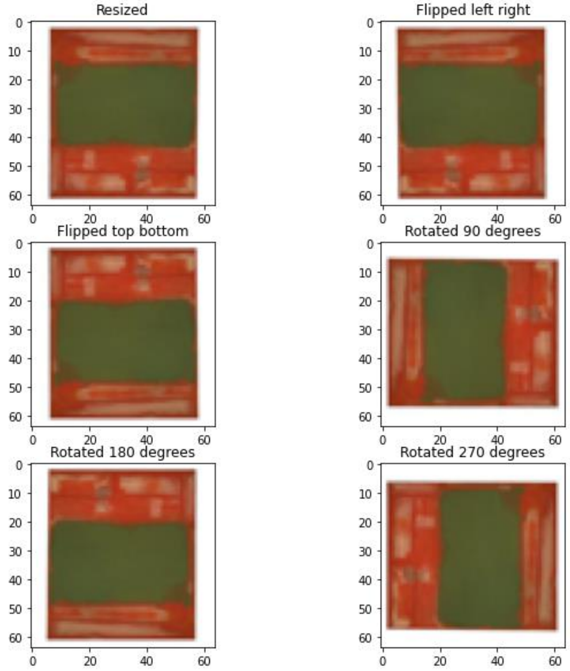
Gene Davis,
Franklin's Footpath

Gene Davis,
Sun Sonata

Image Augmentation

To clean and grow the dataset:

1. Resize images to 64 x 64
2. Rotate images (90, 180 and 270 degrees)
3. Flip images (top-bottom, left-right)



Data Cleaning (Round 2)

Some images were removed as:

- Resizing tall and narrow images only captured a portion of the image
- Some images lacked the characteristics we wanted and would provide little information for training

Data Generation, Augmentation and Cleaning Process

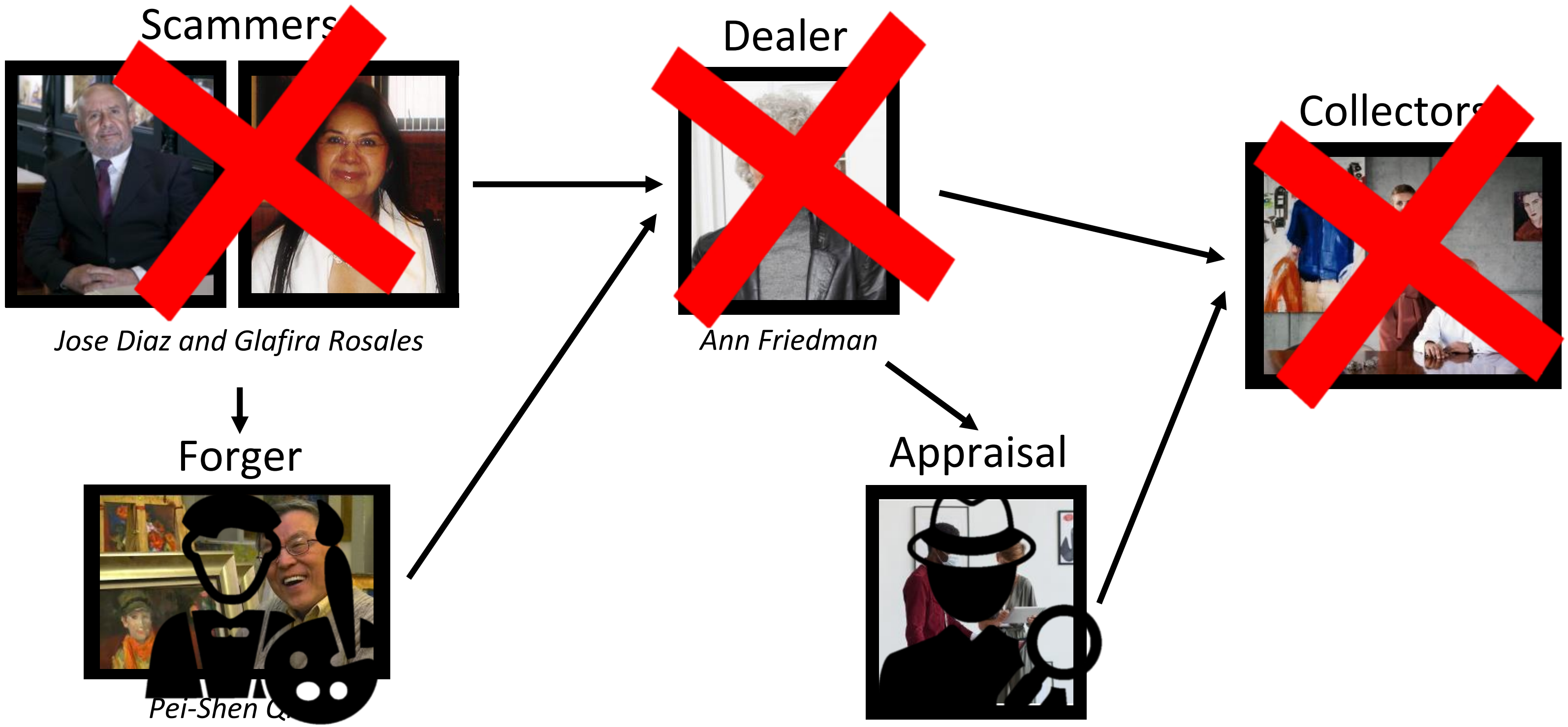
	Scrap Images from Wikiart	Clean images	Resize, augment images	Final Dataset
Number of Images:	940	937	5,622	5,532

03

MODELLING

Build a GAN 101

How to sell fake art?



Contextualising how GANs work

Paints a fake painting and collects a real painting

'Here is a real and a fake painting'

Learns how to classify real vs fake

'This is real because…and this is fake because….'

Take in inputs and improves on generating fakes

Paints 5 fakes

'Here are 5 real paintings'

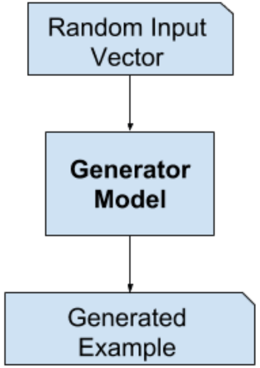
Checks to see if they are real

'2 are real and 3 are fake'

Performance of how the entire GAN

Generator Model

Generator

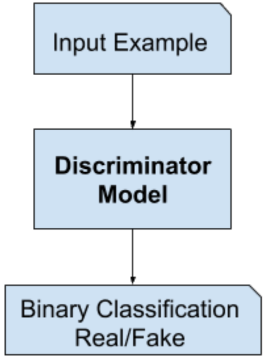



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_11 (Dense)             (None, 4096)              413696
_________________________________________________________________
leaky_re_lu_49 (LeakyReLU)   (None, 4096)              0
_________________________________________________________________
reshape_5 (Reshape)          (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_transpose_20 (Conv2DT (None, 8, 8, 128)         524416
_________________________________________________________________
leaky_re_lu_50 (LeakyReLU)   (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_transpose_21 (Conv2DT (None, 16, 16, 128)       262272
_________________________________________________________________
leaky_re_lu_51 (LeakyReLU)   (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_transpose_22 (Conv2DT (None, 32, 32, 128)       262272
_________________________________________________________________
leaky_re_lu_52 (LeakyReLU)   (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_transpose_23 (Conv2DT (None, 64, 64, 128)       262272
_________________________________________________________________
leaky_re_lu_53 (LeakyReLU)   (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 64, 64, 3)         3459
=================================================================
Total params: 1,728,387
Trainable params: 1,728,387
Non-trainable params: 0
_________________________________________________________________
```

- The latent space recommended is a defined vector of Gaussian-distributed values
- The generator model will draw random points from the latent space randomly and feed them into the generator model during training and this will serve as 'noise'
- Slowly upsamples until ideal output is reached
- Last layer with tanh activation so output is between [-1,1]

Discriminator Model

Discriminator

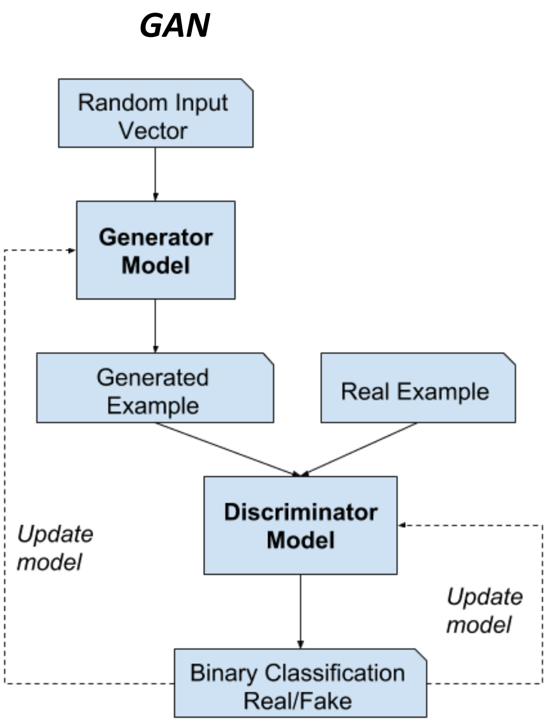


Layer (type)	Output Shape	Param #
conv2d_25 (Conv2D)	(None, 64, 64, 128)	3584
leaky_re_lu_45 (LeakyReLU)	(None, 64, 64, 128)	0
conv2d_26 (Conv2D)	(None, 32, 32, 256)	295168
leaky_re_lu_46 (LeakyReLU)	(None, 32, 32, 256)	0
conv2d_27 (Conv2D)	(None, 16, 16, 256)	590080
leaky_re_lu_47 (LeakyReLU)	(None, 16, 16, 256)	0
conv2d_28 (Conv2D)	(None, 8, 8, 512)	1180160
leaky_re_lu_48 (LeakyReLU)	(None, 8, 8, 512)	0
flatten_5 (Flatten)	(None, 32768)	0
dropout_5 (Dropout)	(None, 32768)	0
dense_10 (Dense)	(None, 1)	32769

Total params: 2,101,761
Trainable params: 2,101,761
Non-trainable params: 0

- The discriminator model will take in our input (64 x 64 images) and return a binary classification
- Starts off with a normal conv layer followed by downsampling
- No pooling layers used, but instead used strided convolutions
- Last layer with sigmoid activation for binary classification

GAN Architecture




```
# define gan
def define_gan(g_model, d_model):
    # make weights in discriminator not trainable
    d_model.trainable = False

    # connect the 2 models
    model = Sequential()

    # add generator
    model.add(g_model)

    # add discriminator
    model.add(d_model)

    # compile
    opt = Adam(learning_rate=adamlr, beta_1 = beta_1)
    model.compile(loss='binary_crossentropy', optimizer = opt)
    return model
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_22 (Sequential)   (None, 64, 64, 3)         1728387
_________________________________________________________________
sequential_21 (Sequential)   (None, 1)                 2101761
=================================================================
Total params: 3,830,148
Trainable params: 1,728,387
Non-trainable params: 2,101,761
_________________________________________________________________
```

- Training for discriminator can be done independently so weights are frozen in the combined model
- Stacks the generator and discriminator model on top of each other
- Loss is still binary crossentropy as it is a classification problem
- Not a CNN per se, but just an easy way to compile models

GAN Training

Outline of Training

- Sample half batch of real images → train discriminator
- Generate half batch of fake images → train discriminator
- Generate fake images, label as real → Feed into combined model and train (only generator is trained)
- Save plots and models every 10 epochs
- Train for 150 epochs

Extensions

1. Include batch normalization in generator
2. Include batch normalization in both generator and discriminator
3. Train model on curated dataset

Model Extentions

1. Batch Normalisation in Generator

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 4096)	413696
leaky_re_lu_1 (LeakyReLU)	(None, 4096)	0
reshape_1 (Reshape)	(None, 4, 4, 256)	0
conv2d_transpose_1 (Conv2DTr	(None, 8, 8, 128)	524416
batch_normalization (BatchNo	(None, 8, 8, 128)	512
leaky_re_lu_2 (LeakyReLU)	(None, 8, 8, 128)	0
conv2d_transpose_2 (Conv2DTr	(None, 16, 16, 128)	262272
batch_normalization_1 (Batch	(None, 16, 16, 128)	512
leaky_re_lu_3 (LeakyReLU)	(None, 16, 16, 128)	0
conv2d_transpose_3 (Conv2DTr	(None, 32, 32, 128)	262272
batch_normalization_2 (Batch	(None, 32, 32, 128)	512
leaky_re_lu_4 (LeakyReLU)	(None, 32, 32, 128)	0
conv2d_transpose_4 (Conv2DTr	(None, 64, 64, 128)	262272
batch_normalization_3 (Batch	(None, 64, 64, 128)	512
leaky_re_lu_5 (LeakyReLU)	(None, 64, 64, 128)	0
conv2d (Conv2D)	(None, 64, 64, 3)	3459

Total params: 1,730,435
Trainable params: 1,729,411
Non-trainable params: 1,024

2. Batch Normalisation in Discriminator

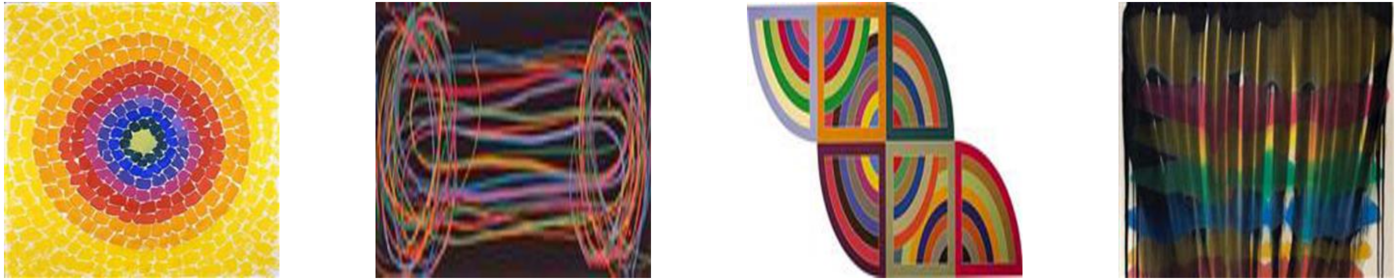
Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 64, 64, 128)	3584
leaky_re_lu_10 (LeakyReLU)	(None, 64, 64, 128)	0
conv2d_6 (Conv2D)	(None, 32, 32, 256)	295168
batch_normalization_7 (Batch	(None, 32, 32, 256)	1024
leaky_re_lu_11 (LeakyReLU)	(None, 32, 32, 256)	0
conv2d_7 (Conv2D)	(None, 16, 16, 256)	590080
batch_normalization_8 (Batch	(None, 16, 16, 256)	1024
leaky_re_lu_12 (LeakyReLU)	(None, 16, 16, 256)	0
conv2d_8 (Conv2D)	(None, 8, 8, 512)	1180160
batch_normalization_9 (Batch	(None, 8, 8, 512)	2048
leaky_re_lu_13 (LeakyReLU)	(None, 8, 8, 512)	0
flatten_1 (Flatten)	(None, 32768)	0
dropout_1 (Dropout)	(None, 32768)	0
dense_3 (Dense)	(None, 1)	32769

Total params: 2,105,857
Trainable params: 2,103,809
Non-trainable params: 2,048

Model Extentions

3.Curated dataset

REMOVE:



KEEP:

1,656 images remained

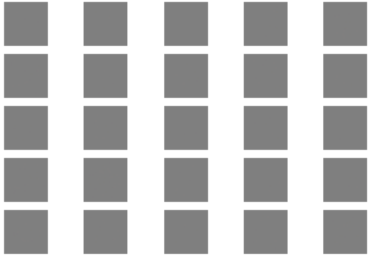
04

EVALUATION & CONCLUSION

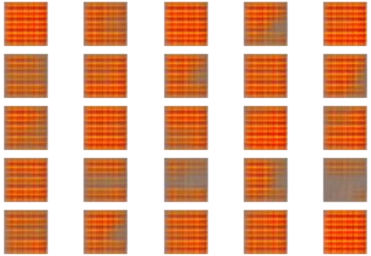
🤔🤔🤔🤔🤔🤔🤔🤔

Model 1 (Baseline Model)

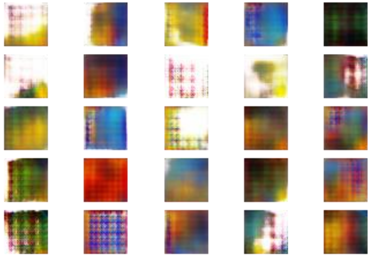
Epoch 0



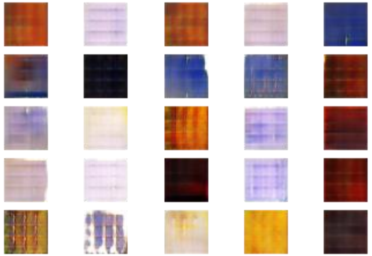
Epoch 10



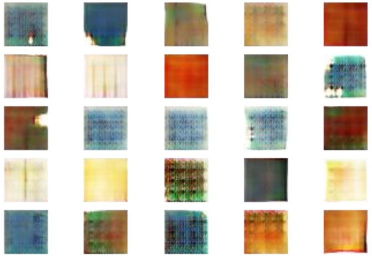
Epoch 50



Epoch 100

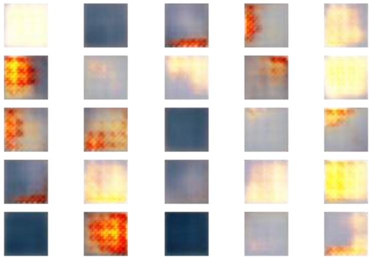


Epoch 150

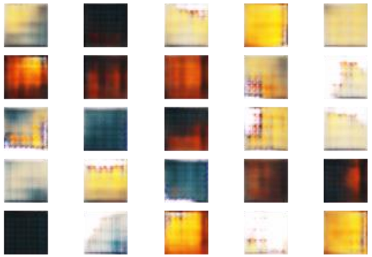


Model 2 (Model with Batch Normalisation in Generator)

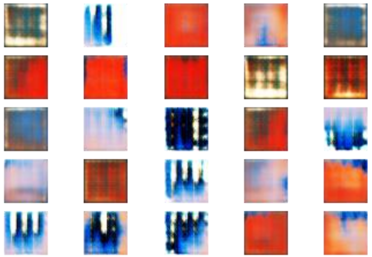
Epoch 10



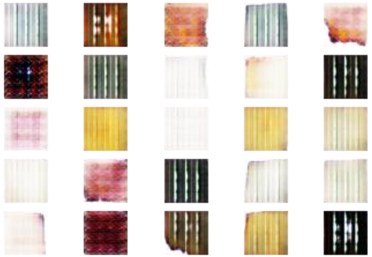
Epoch 50



Epoch 100

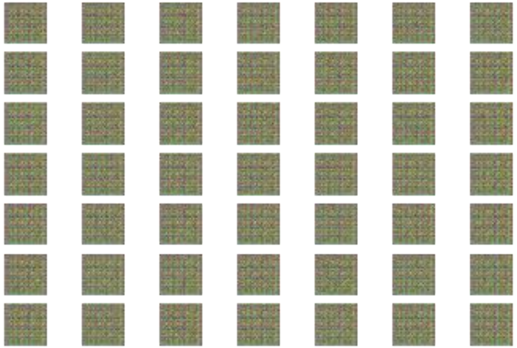


Epoch 150



Model 3 (Model with Batch Normalisation in Discriminator & Generator)

Epoch 10

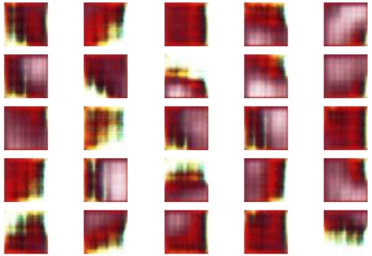


Model Losses hit 0

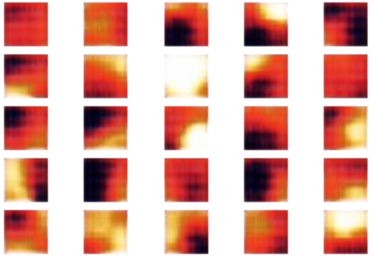
```
>3, 21/43, d1=0.000, d2=0.000 g=0.000
>3, 22/43, d1=0.000, d2=0.000 g=0.000
>3, 23/43, d1=0.000, d2=0.000 g=0.001
>3, 24/43, d1=0.001, d2=0.000 g=0.000
>3, 25/43, d1=0.000, d2=0.000 g=0.000
>3, 26/43, d1=0.000, d2=0.000 g=0.000
>3, 27/43, d1=0.000, d2=0.000 g=0.000
>3, 28/43, d1=0.000, d2=0.000 g=0.001
>3, 29/43, d1=0.000, d2=0.000 g=0.001
>3, 30/43, d1=0.000, d2=0.000 g=0.000
>3, 31/43, d1=0.000, d2=0.000 g=0.000
>3, 32/43, d1=0.000, d2=0.000 g=0.000
>3, 33/43, d1=0.001, d2=0.000 g=0.001
>3, 34/43, d1=0.000, d2=0.000 g=0.000
>3, 35/43, d1=0.000, d2=0.000 g=0.000
>3, 36/43, d1=0.000, d2=0.000 g=0.001
>3, 37/43, d1=0.000, d2=0.000 g=0.001
>3, 38/43, d1=0.000, d2=0.000 g=0.000
>3, 39/43, d1=0.000, d2=0.000 g=0.000
>3, 40/43, d1=0.000, d2=0.000 g=0.000
```

Model 4 (Model with Curated Dataset)

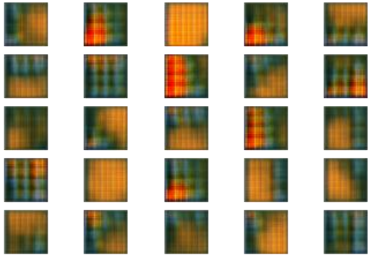
Epoch 10



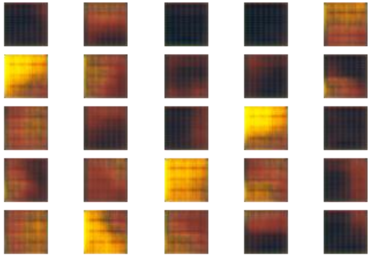
Epoch 50



Epoch 100

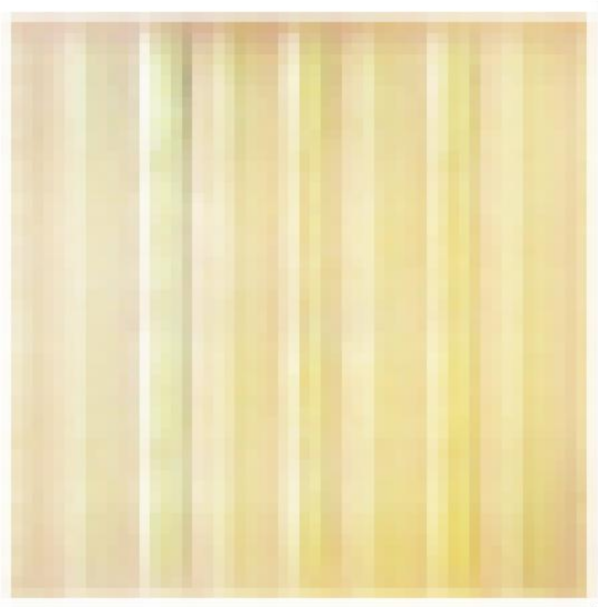


Epoch 150

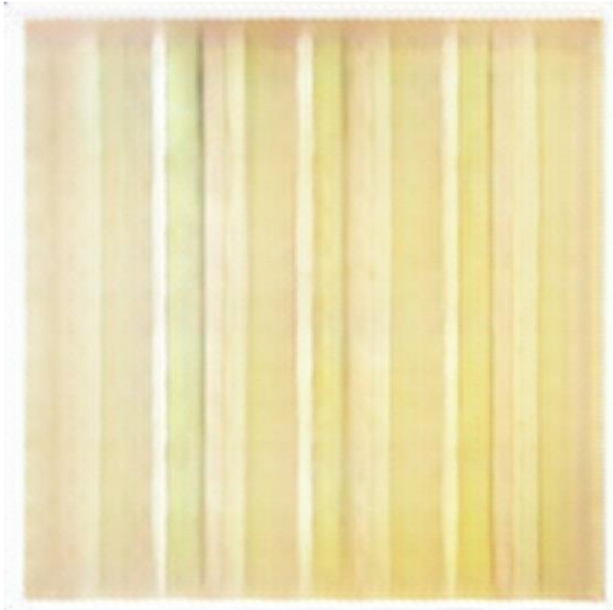


Enhancing results with ESRGAN

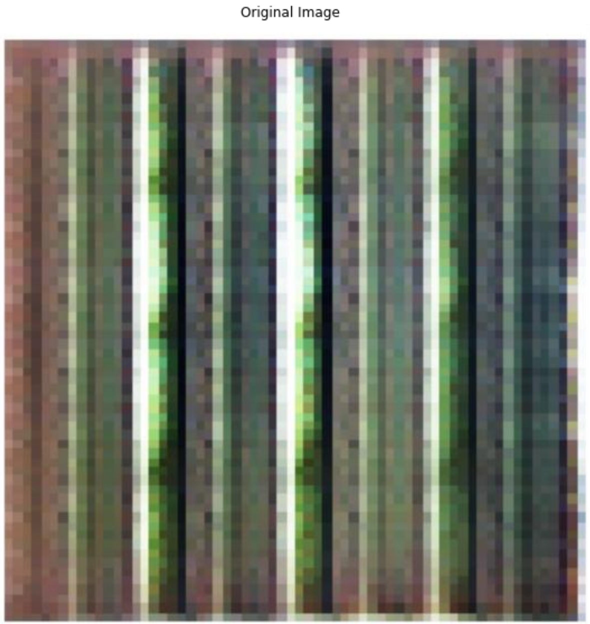
Low Resolution

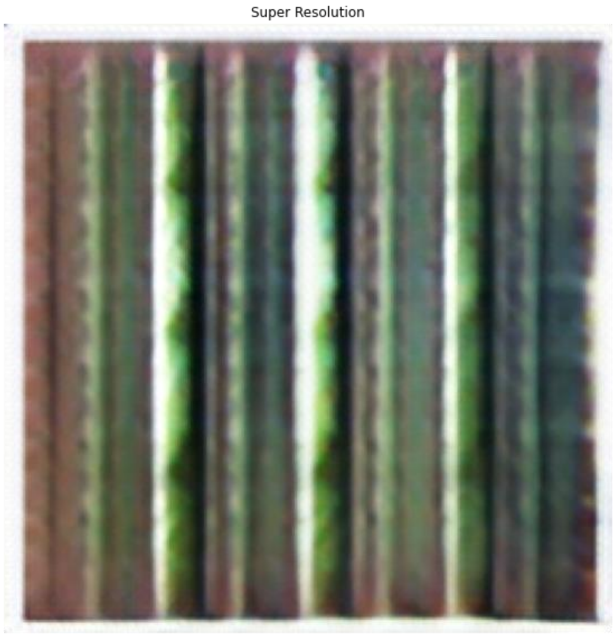


Super Resolution



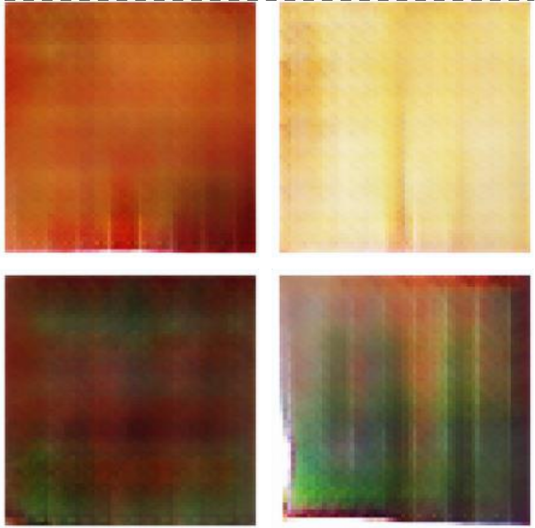
Enhancing results with ESRGAN



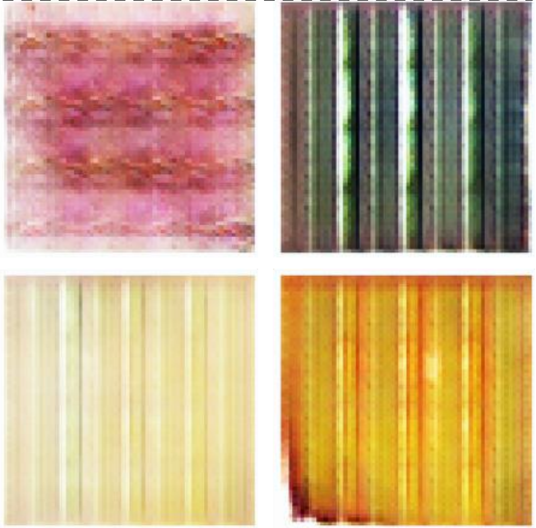


Results

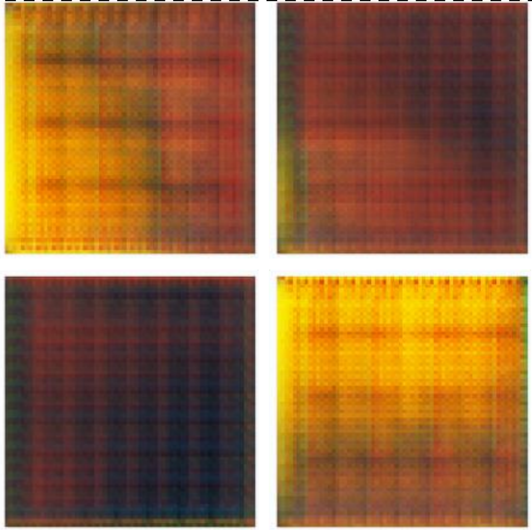
Model 1



Model 2



Model 4



Results

Real Paintings:

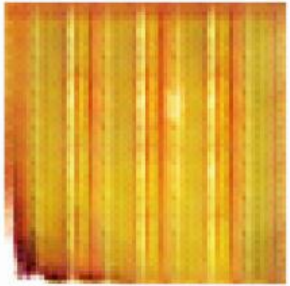
Frank Stella

Gene Davis

Gene Davis

Sam Gilliam

‘Fake’ Paintings:



Takeaways

Best Practices

- Quality of dataset is most important
- Gaussian-distributed values as inputs for generator
- LeakyRELU activation in layers
- Use strided convolutions instead of pooling
- Adam optimiser
- Separate real and fake training for discriminator
- Use batch normalization in generator
- Monitor training progress

THANKS!

github.com/gregwcy