Anime Sketch Colorization Pair

Problem statement:

For a given sketch generate a Color Image

Data Overview

Taken data from Danbooru Sketch Pair 128x on kaggle https://www.kaggle.com/wuhecong/danbooru-sketch-pair-128x (https://www.kaggle.com/wuhecong/danbooru-sketch-pair-128x)

We are given two Folders:-

- Skecth - Having sketch images
- Color - Having corresponding Colored images.

We will pick some images from these folders to train the model and then test it on some pairs not seen by the model.

Real-world/Business objectives and constraints.

- No low-latency requirement.
- Images generated should have colors same as original image and shouldn't be blurred.

Performance metric for supervised learning:

- Mean Squared Error.
- L1 Loss (L1 loss per pixel)
- Binary Cross Entropy loss to predict image generated is real or fake(for GAN models)

Solution :-

We will try to generate Color Images using below models -

- Unet model to minimize Mean Squared error at Pixel Level.
- GAN model to minimize binary cross-entropy loss for real vs fake images(model generated image) and L1 loss at pixel level.

1. Download the Images.

In [0]:
```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Referer: https://www.kaggle.com/" "https://storage.googleapis.com/kaggle-data-sets/198718/800362/bundle/archive.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1580041883&Signature=jx2EuyNyQnYGCQzhHmR2tamyAZPhvi9sRESfQS5TQpCF3ncB1TyDnJbBsKLoFRiC3qav0NxgvMBDQmvSd59ONGYt5oMnKPqUrnDdP9lEPPiYeal7POxeazpNggaBDyZGAXVJvVFdkf5WKAOeBC0yx5NZekMJtE%2FlW%2FIQEc%2B%2F3nfTLUhraE3Y1ys%2BT%2F950ZHJyoKFqeFkhq6GorZc19COPbH9QF%2BfMVPoIkT3Cv6mlSWdKSYmrDKARXJKXxq77eVCY%2BPXqmLKi6PUV85jICuGFNUkJd2fbJVYcniPQXGNCq%2B9B9a%2BIak%2FpM2ViEQklfuxk5MD5JfqFADxn0tkXE0q%2FA%3D%3D&response-content-disposition=attachment%3B+filename%3Ddanbooru-sketch-pair-128x.zip" -O "danbooru-sketch-pair-128x.zip" -c
```

```
--2020-01-24 18:32:58--  https://storage.googleapis.com/kaggle-data-sets/198718/800362/bundle/archive.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1580041883&Signature=jx2EuyNyQnYGCQzhHmR2tamyAZPhvi9sRESfQS5TQpCF3ncB1TyDnJbBsKLoFRiC3qav0NxgvMBDQmvSd59ONGYt5oMnKPqUrnDdP9lEPPiYeal7POxeazpNggaBDyZGAXVJvVFdkf5WKAOeBC0yx5NZekMJtE%2FlW%2FIQEc%2B%2F3nfTLUhraE3Y1ys%2BT%2F950ZHJyoKFqeFkhq6GorZc19COPbH9QF%2BfMVPoIkT3Cv6mlSWdKSYmrDKARXJKXxq77eVCY%2BPXqmLKi6PUV85jICuGFNUkJd2fbJVYcniPQXGNCq%2B9B9a%2BIak%2FpM2ViEQklfuxk5MD5JfqFADxn0tkXE0q%2FA%3D%3D&response-content-disposition=attachment%3B+filename%3Ddanbooru-sketch-pair-128x.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 172.217.204.128, 2607:f8b0:400c:c12::80
Connecting to storage.googleapis.com (storage.googleapis.com)|172.217.204.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10424880981 (9.7G) [application/zip]
Saving to: ‘danbooru-sketch-pair-128x.zip’

danbooru-sketch-pai 100%[===================>]   9.71G  47.9MB/s    in 3m 44s

2020-01-24 18:36:43 (44.3 MB/s) - ‘danbooru-sketch-pair-128x.zip’ saved [10424880981/10424880981]
```

In [0]:
```
#Extract Images from zip folder.
#Ref- https://stackoverflow.com/questions/3451111/unzipping-files-in-python
from zipfile import ZipFile
with ZipFile('danbooru-sketch-pair-128x.zip', 'r') as zf:
    zf.extractall('sketch-pair/')
```

2. Import Required Libraries.

In [0]:
```
#Import all the required libraries.
from keras.layers import Activation, Dense, Dropout, Flatten, Input, Add, Conv
2D, Conv2DTranspose, LeakyReLU #Keras Layers.
from keras.layers.normalization import BatchNormalization #Batch Normalization
for keras layer.
from keras.models import Model #Keras Model
from keras.optimizers import Adam #Adam Optimizer.
from keras import losses #Keras Loss function.
import numpy as np #Numpy array handling.
import os #System variables.
import random #Library to generate random numbers
import cv2 #Image processing.
from google.colab.patches import cv2_imshow #library to show images in Colab
import matplotlib.pyplot as pyplot #Plotting graphs
import warnings #Suppress warning.
warnings.filterwarnings("ignore")
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade (https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info (https://colab.research.google.com/notebooks/tensorflow_version.ipynb).

3. Load Data

In [0]:
```
# Load Sketches.
#Ref- https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/ for cv2 f
unctions.

img_arr = []  #Array to store pixels of sketches.
img_name = [] #Array to store name of the image.

for i in  range(25): #Processing first 20 folders.
  folder=i.__str__().zfill(4) #Printing Folder number in 4 byte format.
  loc = 'sketch-pair/danbooru-sketch-pair-128x/color/sketch/' + folder + '/' #
location of sketch images.
  print('processing folder',folder)   #Print the folder name currently getting
processed.
  for filename in os.listdir(loc):    #Process each image from the current fol
der.
    img_arr.append(cv2.imread(loc+filename,1))    #Append pixel value of each
 image.
    img_name.append(folder+'_'+filename)          #Append name of each image.
```

```
processing folder 0000
processing folder 0001
processing folder 0002
processing folder 0003
processing folder 0004
processing folder 0005
processing folder 0006
processing folder 0007
processing folder 0008
processing folder 0009
processing folder 0010
processing folder 0011
processing folder 0012
processing folder 0013
processing folder 0014
processing folder 0015
processing folder 0016
processing folder 0017
processing folder 0018
processing folder 0019
processing folder 0020
processing folder 0021
processing folder 0022
processing folder 0023
processing folder 0024
```

In [0]:
```
#Print total number of image loaded.
len(img_arr)
```

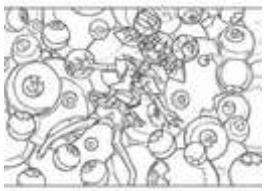
Out[0]: 49489

In [0]:
```
# Load Color Images.
#Ref- https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/ for cv2 f
unctions.

col_arr = []  #Array to store pixels of color images.
col_name = [] #Array to store name of the image.

for i in  range(25): #Processing first 25 folders.
  folder=i.__str__().zfill(4) #Printing Folder number in 4 byte format.
  loc = 'sketch-pair/danbooru-sketch-pair-128x/color/src/' + folder + '/' #loc
ation of color images.
  print('processing folder',folder) #Print the folder name currently getting p
rocessed.
  for filename in os.listdir(loc):  #Process each image from the current folde
r.
    col_arr.append(cv2.imread(loc+filename,1))   #Append pixel value of each i
mage.
    col_name.append(folder+'_'+filename)         #Append name of each image.
```

```
processing folder 0000
processing folder 0001
processing folder 0002
processing folder 0003
processing folder 0004
processing folder 0005
processing folder 0006
processing folder 0007
processing folder 0008
processing folder 0009
processing folder 0010
processing folder 0011
processing folder 0012
processing folder 0013
processing folder 0014
processing folder 0015
processing folder 0016
processing folder 0017
processing folder 0018
processing folder 0019
processing folder 0020
processing folder 0021
processing folder 0022
processing folder 0023
processing folder 0024
```

In [0]:
```
#Convert the pixel variables into numpy array.
col_arr = np.array(col_arr, dtype='uint8')
img_arr = np.array(img_arr, dtype='uint8')
```

In [0]:
```
#Print some sketch and color pair from folders.
for i in range(10):
  cv2_imshow(img_arr[i])
  cv2_imshow(col_arr[i])
```



In [0]:
```
#Check Pixel length
len(col_arr[0][0])
```

Out[0]: 128

4. U-Net Model to minimize Mean Squared Error at Pixel Level

U-Net model

The U-Net model architecture is very similar to encoder decoder model that it involves downsampling to a bottleneck and upsampling again to an output image, but links or skip-connections are made between layers of the same size in the encoder and the decoder, allowing the bottleneck to be circumvented.

Reference - https://machinelearningmastery.com/a-gentle-introduction-to-pix2pix-generative-adversarial-network/ (https://machinelearningmastery.com/a-gentle-introduction-to-pix2pix-generative-adversarial-network/)

In [0]:
```
#Function to generate U-net Model.
#Most of the papers suggested GAN model to generate images. But before jumping directly
#to a complex GAN Model, I tried using simply Generator module to generate images
#improving Mean Squared Error.
#For the Generator Architecture referred - https://github.com/kvfrans/deepcolor/blob/master/main.py.
#The Code above was in tensor-flow. So, changed that to a Keras model.
#Also, most of the papers suggested to normalize pixels value in range of -1 to +1 and use
#tanh as the final layer. But the problem implementating that was, it took lots of memory to
#store float value. So, I tried leaving pixels as uint8 and keeping last layers as Relu,
#So that more images can be trained.
#Also, left the learning paramters as default value, so that model learns faster.

def color_image():

    #Encoder Unit
    #Input Layer (128, 128, 3)
    encoder_input = Input(shape=(128, 128, 3,))
    #Layer-1 Output image of shape - 64 * 64 * 64
    encoder_output_1 = Conv2D(64, (5,5), padding='same',  strides=2)(encoder_input)
    #Leaky ReLUs allow a small, non-zero gradient when the unit is not active.
    encoder_output_2 = LeakyReLU(alpha=0.2)(encoder_output_1)
    #Layer-2 Output image of shape - 32 * 32 * 128
    encoder_output_2 = Conv2D(128, (5,5),  padding='same', strides=2)(encoder_output_2)
    #Batch Normalization.
    encoder_output_2 = BatchNormalization(momentum=0.9)(encoder_output_2)
    #Layer-3 Output image of shape - 16 * 16 * 256
    encoder_output_3 = LeakyReLU(alpha=0.2)(encoder_output_2)
    encoder_output_3 = Conv2D(256, (5,5),  padding='same',  strides=2)(encoder_output_3)
    encoder_output_3 = BatchNormalization(momentum=0.9)(encoder_output_3)
    #Layer-4 Output image of shape - 8 * 8 * 512
    encoder_output_4 = LeakyReLU(alpha=0.2)(encoder_output_3)
    encoder_output_4 = Conv2D(512, (5,5),  padding='same',  strides=2)(encoder_output_4)
    encoder_output_4 = BatchNormalization(momentum=0.9)(encoder_output_4)
    #Layer-5 Output image of shape - 4 * 4 * 512
    encoder_output_5 = LeakyReLU(alpha=0.2)(encoder_output_4)
    encoder_output_5 = Conv2D(512, (5,5),  padding='same',  strides=2)(encoder_output_5)
    encoder_output_5 = BatchNormalization(momentum=0.9)(encoder_output_5)



    #Decoder Unit
    #Conv2DTranspose - The need for transposed convolutions generally arises from the desire to use a transformation going in the opposite
    #direction of a normal convolution, i.e., from something that has the shap
```

```
e of the output of some convolution to
    #something that has the shape of its input while maintaining a connectivit
y pattern that is compatible with said convolution.
    #Ref- https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2DTra
nspose
    #Layer-6 Output image of shape - 8 * 8 * 512
    encoder_output_6 = Conv2DTranspose(512, (5,5), padding='same', activation=
'relu', strides=2)(encoder_output_5)
    encoder_output_6  = BatchNormalization(momentum=0.9)(encoder_output_6)
    #Concatenating output from previous layer with output from layer-4. Both a
re of same dimension.
    encoder_output_6 = Add()([encoder_output_6, encoder_output_4])
    #Layer-7 Output image of shape - 16 * 16 * 256
    encoder_output_7 = Conv2DTranspose(256, (5,5), padding='same', activation=
'relu', strides=2)(encoder_output_6)
    encoder_output_7  = BatchNormalization(momentum=0.9)(encoder_output_7)
    #Concatenating output from previous layer with output from layer-3. Both a
re of same dimension.
    encoder_output_7 = Add()([encoder_output_7, encoder_output_3])
    #Layer-8 Output image of shape - 32 * 32 * 128
    encoder_output_8 = Conv2DTranspose(128, (5,5), padding='same', activation=
'relu', strides=2)(encoder_output_7)
    encoder_output_8  = BatchNormalization(momentum=0.9)(encoder_output_8)
    #Concatenating output from previous layer with output from layer-2. Both a
re of same dimension.
    encoder_output_8 = Add()([encoder_output_8, encoder_output_2])
    #Layer-9 Output image of shape - 64 * 64 * 64
    encoder_output_9 = Conv2DTranspose(64, (5,5), padding='same', activation=
'relu', strides=2)(encoder_output_8)
    encoder_output_9  = BatchNormalization(momentum=0.9)(encoder_output_9)
    #Concatenating output from previous layer with output from layer-1. Both a
re of same dimension
    encoder_output_9 = Add()([encoder_output_9, encoder_output_1])
    #Layer-10 (Output Layer) Output image of shape - 128 * 128 * 3
    encoder_output = Conv2DTranspose(3, (5,5), padding='same', activation='rel
u', strides=2)(encoder_output_9)

    #Defining Model with input and Outputs.
    model = Model(encoder_input, encoder_output)
    #Compile Model with Adam Optimizer and Mean Squared Error loss.
    model.compile(optimizer='adam', loss='mse', )
    #Return Model.
    return model
```

In [0]:
```
#Calling the U-net model eefined above and print its summary.
model = color_image()
model.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:66: The name tf.get_default_graph is deprecated. Please
use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please use
tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:4432: The name tf.random_uniform is deprecated. Please
use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:190: The name tf.get_default_session is deprecated. Ple
ase use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:197: The name tf.ConfigProto is deprecated. Please use
tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:203: The name tf.Session is deprecated. Please use tf.c
ompat.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:207: The name tf.global_variables is deprecated. Please
use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:216: The name tf.is_variable_initialized is deprecated.
Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:223: The name tf.variables_initializer is deprecated. P
lease use tf.compat.v1.variables_initializer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:2041: The name tf.nn.fused_batch_norm is deprecated. Pl
ease use tf.compat.v1.nn.fused_batch_norm instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:148: The name tf.placeholder_with_default is deprecate
d. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimize
rs.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compat.v
1.train.Optimizer instead.

Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 128, 128, 3)       0
_________________________________________________________________
```

```
conv2d_1 (Conv2D)               (None, 64, 64, 64)   4864        input_1[0]
[0]
__________________________________________________________________________
leaky_re_lu_1 (LeakyReLU)       (None, 64, 64, 64)   0           conv2d_1[0]
[0]
__________________________________________________________________________
conv2d_2 (Conv2D)               (None, 32, 32, 128)  204928      leaky_re_lu_
1[0][0]
__________________________________________________________________________
batch_normalization_1 (BatchNor (None, 32, 32, 128)  512         conv2d_2[0]
[0]
__________________________________________________________________________
leaky_re_lu_2 (LeakyReLU)       (None, 32, 32, 128)  0           batch_normal
ization_1[0][0]
__________________________________________________________________________
conv2d_3 (Conv2D)               (None, 16, 16, 256)  819456      leaky_re_lu_
2[0][0]
__________________________________________________________________________
batch_normalization_2 (BatchNor (None, 16, 16, 256)  1024        conv2d_3[0]
[0]
__________________________________________________________________________
leaky_re_lu_3 (LeakyReLU)       (None, 16, 16, 256)  0           batch_normal
ization_2[0][0]
__________________________________________________________________________
conv2d_4 (Conv2D)               (None, 8, 8, 512)    3277312     leaky_re_lu_
3[0][0]
__________________________________________________________________________
batch_normalization_3 (BatchNor (None, 8, 8, 512)    2048        conv2d_4[0]
[0]
__________________________________________________________________________
leaky_re_lu_4 (LeakyReLU)       (None, 8, 8, 512)    0           batch_normal
ization_3[0][0]
__________________________________________________________________________
conv2d_5 (Conv2D)               (None, 4, 4, 512)    6554112     leaky_re_lu_
4[0][0]
__________________________________________________________________________
batch_normalization_4 (BatchNor (None, 4, 4, 512)    2048        conv2d_5[0]
[0]
__________________________________________________________________________
conv2d_transpose_1 (Conv2DTrans (None, 8, 8, 512)    6554112     batch_normal
ization_4[0][0]
__________________________________________________________________________
batch_normalization_5 (BatchNor (None, 8, 8, 512)    2048        conv2d_trans
```

```
pose_1[0][0]
__________________________________________________________________________
__________________________
add_1 (Add)                     (None, 8, 8, 512)    0           batch_normal
ization_5[0][0]
                                                                batch_normal
ization_3[0][0]
__________________________________________________________________________
__________________________
conv2d_transpose_2 (Conv2DTrans (None, 16, 16, 256)  3277056     add_1[0][0]
__________________________________________________________________________
__________________________
batch_normalization_6 (BatchNor (None, 16, 16, 256)  1024        conv2d_trans
pose_2[0][0]
__________________________________________________________________________
__________________________
add_2 (Add)                     (None, 16, 16, 256)  0           batch_normal
ization_6[0][0]
                                                                batch_normal
ization_2[0][0]
__________________________________________________________________________
__________________________
conv2d_transpose_3 (Conv2DTrans (None, 32, 32, 128)  819328      add_2[0][0]
__________________________________________________________________________
__________________________
batch_normalization_7 (BatchNor (None, 32, 32, 128)  512         conv2d_trans
pose_3[0][0]
__________________________________________________________________________
__________________________
add_3 (Add)                     (None, 32, 32, 128)  0           batch_normal
ization_7[0][0]
                                                                batch_normal
ization_1[0][0]
__________________________________________________________________________
__________________________
conv2d_transpose_4 (Conv2DTrans (None, 64, 64, 64)   204864      add_3[0][0]
__________________________________________________________________________
__________________________
batch_normalization_8 (BatchNor (None, 64, 64, 64)   256         conv2d_trans
pose_4[0][0]
__________________________________________________________________________
__________________________
add_4 (Add)                     (None, 64, 64, 64)   0           batch_normal
ization_8[0][0]
                                                                conv2d_1[0]
[0]
__________________________________________________________________________
__________________________
conv2d_transpose_5 (Conv2DTrans (None, 128, 128, 3)  4803        add_4[0][0]
==========================================================================
====================
Total params: 21,730,307
Trainable params: 21,725,571
Non-trainable params: 4,736
__________________________________________________________________________
__________________________
```

In [0]:
```
#Train model for 300 epochs, with batch size of 128 and validation on 0.1.
model.fit(img_arr,col_arr, epochs=300, batch_size=128, validation_split=0.1)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Please use
tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:1020: The name tf.assign is deprecated. Please use tf.c
ompat.v1.assign instead.

Train on 44540 samples, validate on 4949 samples
Epoch 1/300
44540/44540 [==============================] - 109s 2ms/step - loss: 10147.79
32 - val_loss: 8622.6459
Epoch 2/300
44540/44540 [==============================] - 101s 2ms/step - loss: 6285.758
7 - val_loss: 8856.1424
Epoch 3/300
44540/44540 [==============================] - 101s 2ms/step - loss: 4457.394
6 - val_loss: 3942.9479
Epoch 4/300
44540/44540 [==============================] - 101s 2ms/step - loss: 3730.764
8 - val_loss: 4562.5072
Epoch 5/300
44540/44540 [==============================] - 101s 2ms/step - loss: 3472.852
3 - val_loss: 3539.7759
Epoch 6/300
44540/44540 [==============================] - 101s 2ms/step - loss: 3143.207
7 - val_loss: 5094.0595
Epoch 7/300
44540/44540 [==============================] - 101s 2ms/step - loss: 3059.088
0 - val_loss: 18398.1825
Epoch 8/300
44540/44540 [==============================] - 101s 2ms/step - loss: 3228.697
7 - val_loss: 4938.7114
Epoch 9/300
44540/44540 [==============================] - 101s 2ms/step - loss: 3082.871
7 - val_loss: 3196.5249
Epoch 10/300
44540/44540 [==============================] - 101s 2ms/step - loss: 3000.504
8 - val_loss: 3199.7661
Epoch 11/300
44540/44540 [==============================] - 101s 2ms/step - loss: 2460.684
0 - val_loss: 4768.8950
Epoch 12/300
44540/44540 [==============================] - 101s 2ms/step - loss: 2871.804
9 - val_loss: 5418.5672
Epoch 13/300
44540/44540 [==============================] - 101s 2ms/step - loss: 2525.152
3 - val_loss: 3152.3699
Epoch 14/300
44540/44540 [==============================] - 101s 2ms/step - loss: 2394.120
9 - val_loss: 3266.1306
Epoch 15/300
44540/44540 [==============================] - 101s 2ms/step - loss: 2339.551
0 - val_loss: 3416.9577
Epoch 16/300
44540/44540 [==============================] - 101s 2ms/step - loss: 2335.885
6 - val_loss: 5676.9489
```

```
Epoch 17/300
44540/44540 [==============================] - 101s 2ms/step - loss: 1938.431
4 - val_loss: 3172.6518
Epoch 18/300
44540/44540 [==============================] - 101s 2ms/step - loss: 2443.156
4 - val_loss: 3189.3645
Epoch 19/300
44540/44540 [==============================] - 101s 2ms/step - loss: 1986.989
4 - val_loss: 3108.8972
Epoch 20/300
44540/44540 [==============================] - 101s 2ms/step - loss: 1815.843
1 - val_loss: 3304.6499
Epoch 21/300
44540/44540 [==============================] - 101s 2ms/step - loss: 1864.410
8 - val_loss: 7594.2358
Epoch 22/300
44540/44540 [==============================] - 101s 2ms/step - loss: 1850.454
4 - val_loss: 3128.3837
Epoch 23/300
44540/44540 [==============================] - 101s 2ms/step - loss: 1829.524
4 - val_loss: 3076.3513
Epoch 24/300
44540/44540 [==============================] - 101s 2ms/step - loss: 1903.769
0 - val_loss: 3091.3565
Epoch 25/300
44540/44540 [==============================] - 101s 2ms/step - loss: 1672.211
9 - val_loss: 3726.7027
Epoch 26/300
44540/44540 [==============================] - 101s 2ms/step - loss: 1510.937
2 - val_loss: 3217.2559
Epoch 27/300
44540/44540 [==============================] - 101s 2ms/step - loss: 1616.012
9 - val_loss: 8316.3488
Epoch 28/300
44540/44540 [==============================] - 100s 2ms/step - loss: 1630.283
9 - val_loss: 3109.1927
Epoch 29/300
44540/44540 [==============================] - 101s 2ms/step - loss: 1469.013
3 - val_loss: 3327.7821
Epoch 30/300
44540/44540 [==============================] - 100s 2ms/step - loss: 1604.598
9 - val_loss: 3062.5974
Epoch 31/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1291.6535
- val_loss: 10304.9349
Epoch 32/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1771.4424
- val_loss: 3381.3976
Epoch 33/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1251.7417
- val_loss: 4318.6998
Epoch 34/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1476.6061
- val_loss: 3964.9206
Epoch 35/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1208.9087
- val_loss: 3186.2230
```

```
Epoch 36/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1179.6499
- val_loss: 3332.1924
Epoch 37/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1369.5076
- val_loss: 3878.1674
Epoch 38/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1146.3211
- val_loss: 3196.7084
Epoch 39/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1300.1962
- val_loss: 3689.8448
Epoch 40/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1272.6842
- val_loss: 4972.8345
Epoch 41/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1065.7962
- val_loss: 5251.8672
Epoch 42/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1193.0000
- val_loss: 8090.5890
Epoch 43/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1086.4616
- val_loss: 3729.0525
Epoch 44/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1217.3568
- val_loss: 4324.7476
Epoch 45/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1005.3166
- val_loss: 4783.9769
Epoch 46/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1141.1438
- val_loss: 3537.4626
Epoch 47/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1043.6742
- val_loss: 13677.8770
Epoch 48/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1025.6131
- val_loss: 3838.2682
Epoch 49/300
44540/44540 [==============================] - 99s 2ms/step - loss: 958.1216
- val_loss: 3199.4474
Epoch 50/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1213.2406
- val_loss: 19178.6361
Epoch 51/300
44540/44540 [==============================] - 99s 2ms/step - loss: 931.3558
- val_loss: 4567.2789
Epoch 52/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1138.8599
- val_loss: 25071.6734
Epoch 53/300
44540/44540 [==============================] - 99s 2ms/step - loss: 932.7925
- val_loss: 4030.4510
Epoch 54/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1254.6714
- val_loss: 16588.8802
```

```
Epoch 55/300
44540/44540 [==============================] - 99s 2ms/step - loss: 897.5269
- val_loss: 6845.2908
Epoch 56/300
44540/44540 [==============================] - 99s 2ms/step - loss: 888.8844
- val_loss: 3531.9504
Epoch 57/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1205.4298
- val_loss: 3326.3108
Epoch 58/300
44540/44540 [==============================] - 99s 2ms/step - loss: 862.8416
- val_loss: 3282.1340
Epoch 59/300
44540/44540 [==============================] - 99s 2ms/step - loss: 879.8193
- val_loss: 8274.6903
Epoch 60/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1150.1111
- val_loss: 3364.5571
Epoch 61/300
44540/44540 [==============================] - 99s 2ms/step - loss: 857.2830
- val_loss: 3184.1366
Epoch 62/300
44540/44540 [==============================] - 99s 2ms/step - loss: 968.7311
- val_loss: 5165.4302
Epoch 63/300
44540/44540 [==============================] - 99s 2ms/step - loss: 839.2956
- val_loss: 3254.1007
Epoch 64/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1068.1185
- val_loss: 6773.0327
Epoch 65/300
44540/44540 [==============================] - 99s 2ms/step - loss: 813.5213
- val_loss: 3415.1423
Epoch 66/300
44540/44540 [==============================] - 99s 2ms/step - loss: 807.6076
- val_loss: 7878.1316
Epoch 67/300
44540/44540 [==============================] - 99s 2ms/step - loss: 963.6589
- val_loss: 4082.6831
Epoch 68/300
44540/44540 [==============================] - 99s 2ms/step - loss: 861.3737
- val_loss: 13406.5104
Epoch 69/300
44540/44540 [==============================] - 99s 2ms/step - loss: 791.9756
- val_loss: 3218.5775
Epoch 70/300
44540/44540 [==============================] - 99s 2ms/step - loss: 869.0449
- val_loss: 3479.8734
Epoch 71/300
44540/44540 [==============================] - 99s 2ms/step - loss: 942.2498
- val_loss: 3676.5152
Epoch 72/300
44540/44540 [==============================] - 99s 2ms/step - loss: 787.7076
- val_loss: 3469.7838
Epoch 73/300
44540/44540 [==============================] - 99s 2ms/step - loss: 879.9107
- val_loss: 3679.9958
```

```
Epoch 74/300
44540/44540 [==============================] - 99s 2ms/step - loss: 775.2992
- val_loss: 3336.4801
Epoch 75/300
44540/44540 [==============================] - 99s 2ms/step - loss: 931.1185
- val_loss: 13351.6240
Epoch 76/300
44540/44540 [==============================] - 99s 2ms/step - loss: 826.5608
- val_loss: 20463.0421
Epoch 77/300
44540/44540 [==============================] - 99s 2ms/step - loss: 756.6988
- val_loss: 14352.3008
Epoch 78/300
44540/44540 [==============================] - 99s 2ms/step - loss: 916.8612
- val_loss: 3647.2666
Epoch 79/300
44540/44540 [==============================] - 99s 2ms/step - loss: 741.1870
- val_loss: 4829.5876
Epoch 80/300
44540/44540 [==============================] - 99s 2ms/step - loss: 954.0451
- val_loss: 3362.9079
Epoch 81/300
44540/44540 [==============================] - 99s 2ms/step - loss: 742.9709
- val_loss: 3296.0723
Epoch 82/300
44540/44540 [==============================] - 99s 2ms/step - loss: 746.4209
- val_loss: 3878.8733
Epoch 83/300
44540/44540 [==============================] - 99s 2ms/step - loss: 934.1943
- val_loss: 12032.4850
Epoch 84/300
44540/44540 [==============================] - 99s 2ms/step - loss: 755.0007
- val_loss: 7814.6373
Epoch 85/300
44540/44540 [==============================] - 99s 2ms/step - loss: 723.5583
- val_loss: 3714.8453
Epoch 86/300
44540/44540 [==============================] - 99s 2ms/step - loss: 747.0082
- val_loss: 5196.1285
Epoch 87/300
44540/44540 [==============================] - 99s 2ms/step - loss: 702.4617
- val_loss: 3234.8301
Epoch 88/300
44540/44540 [==============================] - 99s 2ms/step - loss: 881.2055
- val_loss: 88502.0721
Epoch 89/300
44540/44540 [==============================] - 99s 2ms/step - loss: 868.4468
- val_loss: 3944.0091
Epoch 90/300
44540/44540 [==============================] - 99s 2ms/step - loss: 749.1512
- val_loss: 23278.4883
Epoch 91/300
44540/44540 [==============================] - 99s 2ms/step - loss: 748.6395
- val_loss: 4024.2623
Epoch 92/300
44540/44540 [==============================] - 99s 2ms/step - loss: 685.9147
- val_loss: 5158.0577
```

```
Epoch 93/300
44540/44540 [==============================] - 99s 2ms/step - loss: 914.7542
- val_loss: 3540.7234
Epoch 94/300
44540/44540 [==============================] - 99s 2ms/step - loss: 689.6870
- val_loss: 3283.0620
Epoch 95/300
44540/44540 [==============================] - 99s 2ms/step - loss: 688.2244
- val_loss: 3266.8349
Epoch 96/300
44540/44540 [==============================] - 99s 2ms/step - loss: 926.2386
- val_loss: 47225.7790
Epoch 97/300
44540/44540 [==============================] - 99s 2ms/step - loss: 726.8214
- val_loss: 4013.7862
Epoch 98/300
44540/44540 [==============================] - 99s 2ms/step - loss: 666.4832
- val_loss: 3648.0925
Epoch 99/300
44540/44540 [==============================] - 99s 2ms/step - loss: 688.5449
- val_loss: 3577.3743
Epoch 100/300
44540/44540 [==============================] - 99s 2ms/step - loss: 668.8009
- val_loss: 3418.1302
Epoch 101/300
44540/44540 [==============================] - 99s 2ms/step - loss: 782.8869
- val_loss: 3546.7114
Epoch 102/300
44540/44540 [==============================] - 99s 2ms/step - loss: 660.4990
- val_loss: 3852.1786
Epoch 103/300
44540/44540 [==============================] - 99s 2ms/step - loss: 672.8443
- val_loss: 3409.5224
Epoch 104/300
44540/44540 [==============================] - 99s 2ms/step - loss: 649.5613
- val_loss: 3755.5426
Epoch 105/300
44540/44540 [==============================] - 99s 2ms/step - loss: 1006.7770
- val_loss: 3907.6600
Epoch 106/300
44540/44540 [==============================] - 99s 2ms/step - loss: 660.9132
- val_loss: 4447.5466
Epoch 107/300
44540/44540 [==============================] - 99s 2ms/step - loss: 657.3107
- val_loss: 3385.1275
Epoch 108/300
44540/44540 [==============================] - 99s 2ms/step - loss: 633.8118
- val_loss: 3456.3745
Epoch 109/300
44540/44540 [==============================] - 99s 2ms/step - loss: 633.3686
- val_loss: 3294.3023
Epoch 110/300
44540/44540 [==============================] - 99s 2ms/step - loss: 720.7908
- val_loss: 3632.2213
Epoch 111/300
44540/44540 [==============================] - 99s 2ms/step - loss: 638.2835
- val_loss: 3419.1826
```

```
Epoch 112/300
44540/44540 [==============================] - 99s 2ms/step - loss: 866.3460
- val_loss: 3789.2664
Epoch 113/300
44540/44540 [==============================] - 99s 2ms/step - loss: 634.0802
- val_loss: 5899.0364
Epoch 114/300
44540/44540 [==============================] - 99s 2ms/step - loss: 619.0574
- val_loss: 3633.0149
Epoch 115/300
44540/44540 [==============================] - 99s 2ms/step - loss: 776.5456
- val_loss: 3255.2862
Epoch 116/300
44540/44540 [==============================] - 99s 2ms/step - loss: 628.6247
- val_loss: 3584.2760
Epoch 117/300
44540/44540 [==============================] - 99s 2ms/step - loss: 617.7092
- val_loss: 3460.6137
Epoch 118/300
44540/44540 [==============================] - 99s 2ms/step - loss: 623.1687
- val_loss: 3505.8810
Epoch 119/300
44540/44540 [==============================] - 99s 2ms/step - loss: 617.4608
- val_loss: 3414.0112
Epoch 120/300
44540/44540 [==============================] - 99s 2ms/step - loss: 679.7887
- val_loss: 3662.6918
Epoch 121/300
44540/44540 [==============================] - 99s 2ms/step - loss: 613.4460
- val_loss: 3616.3852
Epoch 122/300
44540/44540 [==============================] - 99s 2ms/step - loss: 611.4945
- val_loss: 3340.8216
Epoch 123/300
44540/44540 [==============================] - 99s 2ms/step - loss: 921.0241
- val_loss: 85732.7850
Epoch 124/300
44540/44540 [==============================] - 99s 2ms/step - loss: 673.8156
- val_loss: 3251.3410
Epoch 125/300
44540/44540 [==============================] - 99s 2ms/step - loss: 605.6076
- val_loss: 5143.1874
Epoch 126/300
44540/44540 [==============================] - 99s 2ms/step - loss: 592.0532
- val_loss: 3656.5443
Epoch 127/300
44540/44540 [==============================] - 99s 2ms/step - loss: 594.4266
- val_loss: 3977.0397
Epoch 128/300
44540/44540 [==============================] - 99s 2ms/step - loss: 593.6745
- val_loss: 3270.0814
Epoch 129/300
44540/44540 [==============================] - 99s 2ms/step - loss: 599.0227
- val_loss: 64035.0853
Epoch 130/300
44540/44540 [==============================] - 99s 2ms/step - loss: 818.8697
- val_loss: 5742.0268
```

```
Epoch 131/300
44540/44540 [==============================] - 99s 2ms/step - loss: 597.8970
- val_loss: 3352.3532
Epoch 132/300
44540/44540 [==============================] - 99s 2ms/step - loss: 588.4173
- val_loss: 3394.1594
Epoch 133/300
44540/44540 [==============================] - 99s 2ms/step - loss: 588.7898
- val_loss: 3410.7135
Epoch 134/300
44540/44540 [==============================] - 99s 2ms/step - loss: 611.2096
- val_loss: 3483.6194
Epoch 135/300
44540/44540 [==============================] - 99s 2ms/step - loss: 616.6699
- val_loss: 4297.0731
Epoch 136/300
44540/44540 [==============================] - 99s 2ms/step - loss: 639.4018
- val_loss: 3812.0431
Epoch 137/300
44540/44540 [==============================] - 99s 2ms/step - loss: 623.1203
- val_loss: 3383.7154
Epoch 138/300
44540/44540 [==============================] - 99s 2ms/step - loss: 578.6315
- val_loss: 3458.3192
Epoch 139/300
44540/44540 [==============================] - 99s 2ms/step - loss: 570.7181
- val_loss: 4033.2953
Epoch 140/300
44540/44540 [==============================] - 99s 2ms/step - loss: 572.6745
- val_loss: 3385.2572
Epoch 141/300
44540/44540 [==============================] - 99s 2ms/step - loss: 798.6104
- val_loss: 7232.0779
Epoch 142/300
44540/44540 [==============================] - 99s 2ms/step - loss: 650.5584
- val_loss: 4314.0709
Epoch 143/300
44540/44540 [==============================] - 99s 2ms/step - loss: 574.2995
- val_loss: 3386.9662
Epoch 144/300
44540/44540 [==============================] - 99s 2ms/step - loss: 562.2024
- val_loss: 3357.7010
Epoch 145/300
44540/44540 [==============================] - 99s 2ms/step - loss: 559.9426
- val_loss: 3659.9490
Epoch 146/300
44540/44540 [==============================] - 99s 2ms/step - loss: 680.9051
- val_loss: 3403.2587
Epoch 147/300
44540/44540 [==============================] - 99s 2ms/step - loss: 561.0261
- val_loss: 3294.2413
Epoch 148/300
44540/44540 [==============================] - 99s 2ms/step - loss: 558.0885
- val_loss: 23617.2706
Epoch 149/300
44540/44540 [==============================] - 99s 2ms/step - loss: 811.1606
- val_loss: 3324.8316
```

```
Epoch 150/300
44540/44540 [==============================] - 99s 2ms/step - loss: 570.0879
- val_loss: 3441.6419
Epoch 151/300
44540/44540 [==============================] - 99s 2ms/step - loss: 547.8157
- val_loss: 3441.8568
Epoch 152/300
44540/44540 [==============================] - 99s 2ms/step - loss: 541.2799
- val_loss: 3436.8887
Epoch 153/300
44540/44540 [==============================] - 99s 2ms/step - loss: 564.2860
- val_loss: 4079.4667
Epoch 154/300
44540/44540 [==============================] - 99s 2ms/step - loss: 549.7886
- val_loss: 3501.7512
Epoch 155/300
44540/44540 [==============================] - 99s 2ms/step - loss: 679.0076
- val_loss: 8210.5894
Epoch 156/300
44540/44540 [==============================] - 99s 2ms/step - loss: 555.7458
- val_loss: 3387.4464
Epoch 157/300
44540/44540 [==============================] - 99s 2ms/step - loss: 539.7338
- val_loss: 6548.0582
Epoch 158/300
44540/44540 [==============================] - 99s 2ms/step - loss: 539.3498
- val_loss: 4615.5681
Epoch 159/300
44540/44540 [==============================] - 99s 2ms/step - loss: 540.1125
- val_loss: 3583.9630
Epoch 160/300
44540/44540 [==============================] - 99s 2ms/step - loss: 662.7258
- val_loss: 3514.5577
Epoch 161/300
44540/44540 [==============================] - 99s 2ms/step - loss: 550.4945
- val_loss: 6472.1557
Epoch 162/300
44540/44540 [==============================] - 99s 2ms/step - loss: 535.8266
- val_loss: 3308.4374
Epoch 163/300
44540/44540 [==============================] - 99s 2ms/step - loss: 531.7263
- val_loss: 3759.9465
Epoch 164/300
44540/44540 [==============================] - 99s 2ms/step - loss: 696.8360
- val_loss: 4360.8521
Epoch 165/300
44540/44540 [==============================] - 99s 2ms/step - loss: 553.6650
- val_loss: 3658.8087
Epoch 166/300
44540/44540 [==============================] - 99s 2ms/step - loss: 533.7614
- val_loss: 7169.1063
Epoch 167/300
44540/44540 [==============================] - 99s 2ms/step - loss: 520.2717
- val_loss: 3777.8591
Epoch 168/300
44540/44540 [==============================] - 99s 2ms/step - loss: 582.6400
- val_loss: 3354.6656
```

```
Epoch 169/300
44540/44540 [==============================] - 99s 2ms/step - loss: 528.9124
- val_loss: 3365.0296
Epoch 170/300
44540/44540 [==============================] - 99s 2ms/step - loss: 526.6344
- val_loss: 3407.7367
Epoch 171/300
44540/44540 [==============================] - 99s 2ms/step - loss: 644.2972
- val_loss: 3441.6810
Epoch 172/300
44540/44540 [==============================] - 99s 2ms/step - loss: 533.7297
- val_loss: 3345.7505
Epoch 173/300
44540/44540 [==============================] - 99s 2ms/step - loss: 521.0771
- val_loss: 3634.1302
Epoch 174/300
44540/44540 [==============================] - 99s 2ms/step - loss: 658.5817
- val_loss: 3349.1643
Epoch 175/300
44540/44540 [==============================] - 99s 2ms/step - loss: 529.4065
- val_loss: 3408.3859
Epoch 176/300
44540/44540 [==============================] - 99s 2ms/step - loss: 515.6957
- val_loss: 3397.2574
Epoch 177/300
44540/44540 [==============================] - 99s 2ms/step - loss: 516.4890
- val_loss: 3773.3002
Epoch 178/300
44540/44540 [==============================] - 99s 2ms/step - loss: 515.1107
- val_loss: 3423.1509
Epoch 179/300
44540/44540 [==============================] - 99s 2ms/step - loss: 678.9862
- val_loss: 3440.5758
Epoch 180/300
44540/44540 [==============================] - 99s 2ms/step - loss: 515.5163
- val_loss: 7288.6948
Epoch 181/300
44540/44540 [==============================] - 99s 2ms/step - loss: 507.4762
- val_loss: 3431.2892
Epoch 182/300
44540/44540 [==============================] - 99s 2ms/step - loss: 503.8039
- val_loss: 3356.7455
Epoch 183/300
44540/44540 [==============================] - 99s 2ms/step - loss: 508.1932
- val_loss: 3383.4273
Epoch 184/300
44540/44540 [==============================] - 99s 2ms/step - loss: 694.8108
- val_loss: 3451.3038
Epoch 185/300
44540/44540 [==============================] - 99s 2ms/step - loss: 518.1596
- val_loss: 3606.6494
Epoch 186/300
44540/44540 [==============================] - 99s 2ms/step - loss: 503.2321
- val_loss: 5023.6911
Epoch 187/300
44540/44540 [==============================] - 99s 2ms/step - loss: 501.8434
- val_loss: 3753.2401
```

```
Epoch 188/300
44540/44540 [==============================] - 99s 2ms/step - loss: 501.9605
- val_loss: 3326.5279
Epoch 189/300
44540/44540 [==============================] - 99s 2ms/step - loss: 656.5778
- val_loss: 3483.8319
Epoch 190/300
44540/44540 [==============================] - 99s 2ms/step - loss: 510.6595
- val_loss: 3535.3605
Epoch 191/300
44540/44540 [==============================] - 99s 2ms/step - loss: 500.4282
- val_loss: 3350.9350
Epoch 192/300
44540/44540 [==============================] - 99s 2ms/step - loss: 495.7163
- val_loss: 3649.3753
Epoch 193/300
44540/44540 [==============================] - 99s 2ms/step - loss: 651.0419
- val_loss: 3423.8880
Epoch 194/300
44540/44540 [==============================] - 99s 2ms/step - loss: 500.6897
- val_loss: 3390.7248
Epoch 195/300
44540/44540 [==============================] - 99s 2ms/step - loss: 494.4202
- val_loss: 4781.9861
Epoch 196/300
44540/44540 [==============================] - 99s 2ms/step - loss: 489.6494
- val_loss: 3783.7701
Epoch 197/300
44540/44540 [==============================] - 99s 2ms/step - loss: 493.3844
- val_loss: 3484.6685
Epoch 198/300
44540/44540 [==============================] - 99s 2ms/step - loss: 499.2212
- val_loss: 3381.5402
Epoch 199/300
44540/44540 [==============================] - 99s 2ms/step - loss: 522.9042
- val_loss: 3411.3141
Epoch 200/300
44540/44540 [==============================] - 99s 2ms/step - loss: 493.3865
- val_loss: 3681.9921
Epoch 201/300
44540/44540 [==============================] - 99s 2ms/step - loss: 710.7519
- val_loss: 3770.5662
Epoch 202/300
44540/44540 [==============================] - 99s 2ms/step - loss: 497.3451
- val_loss: 3573.1808
Epoch 203/300
44540/44540 [==============================] - 99s 2ms/step - loss: 483.9733
- val_loss: 3414.1073
Epoch 204/300
44540/44540 [==============================] - 99s 2ms/step - loss: 485.7774
- val_loss: 3379.7939
Epoch 205/300
44540/44540 [==============================] - 99s 2ms/step - loss: 498.2721
- val_loss: 3449.5841
Epoch 206/300
44540/44540 [==============================] - 99s 2ms/step - loss: 490.6076
- val_loss: 3519.3333
```

```
Epoch 207/300
44540/44540 [==============================] - 99s 2ms/step - loss: 490.8627
- val_loss: 3429.6025
Epoch 208/300
44540/44540 [==============================] - 99s 2ms/step - loss: 630.0151
- val_loss: 3994.0503
Epoch 209/300
44540/44540 [==============================] - 99s 2ms/step - loss: 491.0394
- val_loss: 3493.8567
Epoch 210/300
44540/44540 [==============================] - 99s 2ms/step - loss: 596.3132
- val_loss: 3524.0283
Epoch 211/300
44540/44540 [==============================] - 99s 2ms/step - loss: 482.4974
- val_loss: 3425.0075
Epoch 212/300
44540/44540 [==============================] - 99s 2ms/step - loss: 475.9432
- val_loss: 4093.3309
Epoch 213/300
44540/44540 [==============================] - 99s 2ms/step - loss: 485.4934
- val_loss: 3453.4674
Epoch 214/300
44540/44540 [==============================] - 99s 2ms/step - loss: 593.3600
- val_loss: 3936.8192
Epoch 215/300
44540/44540 [==============================] - 99s 2ms/step - loss: 493.9781
- val_loss: 3662.7189
Epoch 216/300
44540/44540 [==============================] - 99s 2ms/step - loss: 478.1682
- val_loss: 3628.1291
Epoch 217/300
44540/44540 [==============================] - 99s 2ms/step - loss: 552.4044
- val_loss: 3395.0851
Epoch 218/300
44540/44540 [==============================] - 99s 2ms/step - loss: 476.4116
- val_loss: 3385.4928
Epoch 219/300
44540/44540 [==============================] - 99s 2ms/step - loss: 474.6181
- val_loss: 3503.6648
Epoch 220/300
44540/44540 [==============================] - 99s 2ms/step - loss: 472.5034
- val_loss: 3384.8848
Epoch 221/300
44540/44540 [==============================] - 99s 2ms/step - loss: 487.3919
- val_loss: 3491.3174
Epoch 222/300
44540/44540 [==============================] - 99s 2ms/step - loss: 485.4643
- val_loss: 3758.8306
Epoch 223/300
44540/44540 [==============================] - 99s 2ms/step - loss: 480.3855
- val_loss: 4316.2257
Epoch 224/300
44540/44540 [==============================] - 99s 2ms/step - loss: 619.7343
- val_loss: 3408.7459
Epoch 225/300
44540/44540 [==============================] - 99s 2ms/step - loss: 486.1134
- val_loss: 3479.7777
```

```
Epoch 226/300
44540/44540 [==============================] - 99s 2ms/step - loss: 466.0596
- val_loss: 3371.2889
Epoch 227/300
44540/44540 [==============================] - 99s 2ms/step - loss: 466.0188
- val_loss: 3872.1136
Epoch 228/300
44540/44540 [==============================] - 99s 2ms/step - loss: 468.8297
- val_loss: 3734.1070
Epoch 229/300
44540/44540 [==============================] - 99s 2ms/step - loss: 468.9234
- val_loss: 3378.2597
Epoch 230/300
44540/44540 [==============================] - 99s 2ms/step - loss: 471.0995
- val_loss: 3994.2141
Epoch 231/300
44540/44540 [==============================] - 99s 2ms/step - loss: 789.3470
- val_loss: 3770.5880
Epoch 232/300
44540/44540 [==============================] - 99s 2ms/step - loss: 498.8064
- val_loss: 3661.6610
Epoch 233/300
44540/44540 [==============================] - 99s 2ms/step - loss: 460.1960
- val_loss: 3442.3065
Epoch 234/300
44540/44540 [==============================] - 99s 2ms/step - loss: 454.3326
- val_loss: 3528.4925
Epoch 235/300
44540/44540 [==============================] - 99s 2ms/step - loss: 456.8901
- val_loss: 3420.9108
Epoch 236/300
44540/44540 [==============================] - 99s 2ms/step - loss: 464.0041
- val_loss: 5501.9796
Epoch 237/300
44540/44540 [==============================] - 99s 2ms/step - loss: 462.2733
- val_loss: 3357.0464
Epoch 238/300
44540/44540 [==============================] - 99s 2ms/step - loss: 469.0180
- val_loss: 3437.4857
Epoch 239/300
44540/44540 [==============================] - 99s 2ms/step - loss: 627.3381
- val_loss: 3451.3029
Epoch 240/300
44540/44540 [==============================] - 99s 2ms/step - loss: 464.0778
- val_loss: 4296.5557
Epoch 241/300
44540/44540 [==============================] - 99s 2ms/step - loss: 453.8298
- val_loss: 3731.3057
Epoch 242/300
44540/44540 [==============================] - 99s 2ms/step - loss: 455.0099
- val_loss: 3466.6967
Epoch 243/300
44540/44540 [==============================] - 99s 2ms/step - loss: 469.9331
- val_loss: 3591.0916
Epoch 244/300
44540/44540 [==============================] - 99s 2ms/step - loss: 560.0060
- val_loss: 3423.6649
```

```
Epoch 245/300
44540/44540 [==============================] - 99s 2ms/step - loss: 469.4184
- val_loss: 3382.9011
Epoch 246/300
44540/44540 [==============================] - 99s 2ms/step - loss: 453.5691
- val_loss: 3403.8533
Epoch 247/300
44540/44540 [==============================] - 99s 2ms/step - loss: 460.3530
- val_loss: 3470.1992
Epoch 248/300
44540/44540 [==============================] - 99s 2ms/step - loss: 455.4547
- val_loss: 3424.4306
Epoch 249/300
44540/44540 [==============================] - 99s 2ms/step - loss: 455.3654
- val_loss: 4472.6976
Epoch 250/300
44540/44540 [==============================] - 99s 2ms/step - loss: 464.9190
- val_loss: 3573.4426
Epoch 251/300
44540/44540 [==============================] - 99s 2ms/step - loss: 470.0503
- val_loss: 3480.0822
Epoch 252/300
44540/44540 [==============================] - 99s 2ms/step - loss: 595.2204
- val_loss: 3535.2283
Epoch 253/300
44540/44540 [==============================] - 99s 2ms/step - loss: 465.3265
- val_loss: 3394.5742
Epoch 254/300
44540/44540 [==============================] - 99s 2ms/step - loss: 448.5791
- val_loss: 3454.8058
Epoch 255/300
44540/44540 [==============================] - 99s 2ms/step - loss: 447.9859
- val_loss: 3439.0621
Epoch 256/300
44540/44540 [==============================] - 99s 2ms/step - loss: 450.0087
- val_loss: 4655.2939
Epoch 257/300
44540/44540 [==============================] - 99s 2ms/step - loss: 467.1792
- val_loss: 3850.8592
Epoch 258/300
44540/44540 [==============================] - 99s 2ms/step - loss: 453.9010
- val_loss: 4098.0379
Epoch 259/300
44540/44540 [==============================] - 99s 2ms/step - loss: 458.8888
- val_loss: 25039.8829
Epoch 260/300
44540/44540 [==============================] - 99s 2ms/step - loss: 453.8286
- val_loss: 3506.8682
Epoch 261/300
44540/44540 [==============================] - 99s 2ms/step - loss: 703.1432
- val_loss: 3433.9379
Epoch 262/300
44540/44540 [==============================] - 99s 2ms/step - loss: 455.0352
- val_loss: 3423.4998
Epoch 263/300
44540/44540 [==============================] - 99s 2ms/step - loss: 437.9571
- val_loss: 3632.7751
```

```
Epoch 264/300
44540/44540 [==============================] - 99s 2ms/step - loss: 440.0401
- val_loss: 4059.5181
Epoch 265/300
44540/44540 [==============================] - 99s 2ms/step - loss: 439.2545
- val_loss: 3608.2373
Epoch 266/300
44540/44540 [==============================] - 99s 2ms/step - loss: 446.5662
- val_loss: 3591.1824
Epoch 267/300
44540/44540 [==============================] - 99s 2ms/step - loss: 469.7728
- val_loss: 4221.9735
Epoch 268/300
44540/44540 [==============================] - 99s 2ms/step - loss: 449.3295
- val_loss: 3774.7973
Epoch 269/300
44540/44540 [==============================] - 99s 2ms/step - loss: 442.5411
- val_loss: 3604.5955
Epoch 270/300
44540/44540 [==============================] - 99s 2ms/step - loss: 446.9354
- val_loss: 3620.6169
Epoch 271/300
44540/44540 [==============================] - 99s 2ms/step - loss: 482.3575
- val_loss: 3542.2826
Epoch 272/300
44540/44540 [==============================] - 99s 2ms/step - loss: 440.6953
- val_loss: 3460.9664
Epoch 273/300
44540/44540 [==============================] - 99s 2ms/step - loss: 440.3153
- val_loss: 13980.5697
Epoch 274/300
44540/44540 [==============================] - 99s 2ms/step - loss: 531.2421
- val_loss: 3554.9030
Epoch 275/300
44540/44540 [==============================] - 99s 2ms/step - loss: 441.6330
- val_loss: 3660.7516
Epoch 276/300
44540/44540 [==============================] - 99s 2ms/step - loss: 432.1249
- val_loss: 3370.5203
Epoch 277/300
44540/44540 [==============================] - 99s 2ms/step - loss: 436.1243
- val_loss: 4123.8330
Epoch 278/300
44540/44540 [==============================] - 99s 2ms/step - loss: 444.8287
- val_loss: 6744.7296
Epoch 279/300
44540/44540 [==============================] - 99s 2ms/step - loss: 444.3331
- val_loss: 3398.5714
Epoch 280/300
44540/44540 [==============================] - 99s 2ms/step - loss: 579.4839
- val_loss: 3544.8031
Epoch 281/300
44540/44540 [==============================] - 99s 2ms/step - loss: 442.4762
- val_loss: 3424.8174
Epoch 282/300
44540/44540 [==============================] - 99s 2ms/step - loss: 428.6541
- val_loss: 3782.0551
```

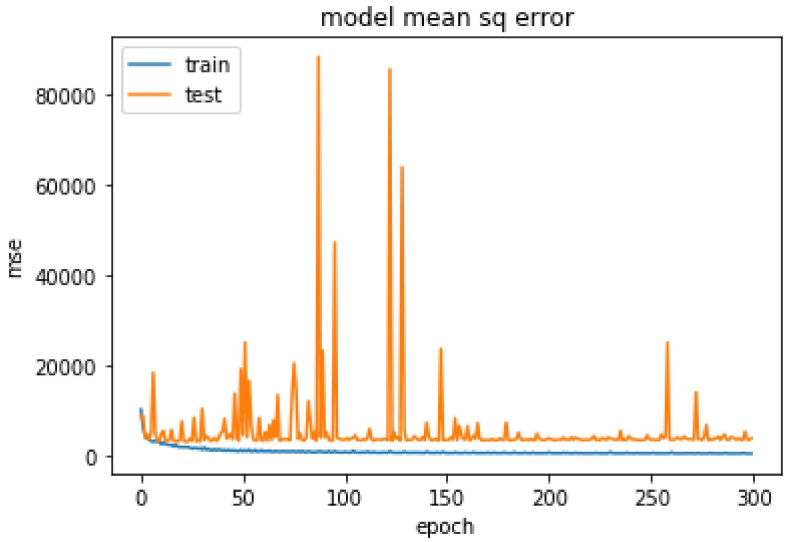
```
Epoch 283/300
44540/44540 [==============================] - 99s 2ms/step - loss: 431.6176
- val_loss: 3747.7261
Epoch 284/300
44540/44540 [==============================] - 99s 2ms/step - loss: 435.4263
- val_loss: 4201.7012
Epoch 285/300
44540/44540 [==============================] - 99s 2ms/step - loss: 433.5957
- val_loss: 3419.4100
Epoch 286/300
44540/44540 [==============================] - 99s 2ms/step - loss: 434.9704
- val_loss: 4118.0370
Epoch 287/300
44540/44540 [==============================] - 99s 2ms/step - loss: 588.8200
- val_loss: 4631.0388
Epoch 288/300
44540/44540 [==============================] - 99s 2ms/step - loss: 459.8041
- val_loss: 3471.6201
Epoch 289/300
44540/44540 [==============================] - 99s 2ms/step - loss: 430.5604
- val_loss: 3415.4289
Epoch 290/300
44540/44540 [==============================] - 99s 2ms/step - loss: 424.5452
- val_loss: 4056.5421
Epoch 291/300
44540/44540 [==============================] - 99s 2ms/step - loss: 425.9744
- val_loss: 3944.7303
Epoch 292/300
44540/44540 [==============================] - 99s 2ms/step - loss: 432.1702
- val_loss: 3698.1023
Epoch 293/300
44540/44540 [==============================] - 99s 2ms/step - loss: 442.5922
- val_loss: 3466.7356
Epoch 294/300
44540/44540 [==============================] - 99s 2ms/step - loss: 439.1447
- val_loss: 3829.7121
Epoch 295/300
44540/44540 [==============================] - 99s 2ms/step - loss: 434.6115
- val_loss: 3434.1967
Epoch 296/300
44540/44540 [==============================] - 99s 2ms/step - loss: 570.6531
- val_loss: 3439.3651
Epoch 297/300
44540/44540 [==============================] - 99s 2ms/step - loss: 430.9317
- val_loss: 5265.2566
Epoch 298/300
44540/44540 [==============================] - 99s 2ms/step - loss: 426.3315
- val_loss: 3649.2890
Epoch 299/300
44540/44540 [==============================] - 99s 2ms/step - loss: 426.2318
- val_loss: 3366.3263
Epoch 300/300
44540/44540 [==============================] - 99s 2ms/step - loss: 423.8286
- val_loss: 3799.5276
```

Out[0]: <keras.callbacks.History at 0x7f95da5ffa20>

In [0]:
```
#Saving the model.
model.save_weights("model_300.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

In [0]:
```
# Plotting the Mean Squared Error vs Epoch Graph
import matplotlib.pyplot as plt
plt.plot(model.history.history['loss'])
plt.plot(model.history.history['val_loss'])
plt.title('model mean sq error')
plt.ylabel('mse')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



Observation -

Model couldn't improve much on test data.

In [0]:
```
#Generating Color Image on some train data.
tst = img_arr[0:10]
res = model.predict(tst)
```

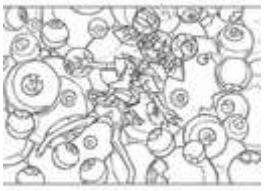
In [0]:
```
#Print color image to compare results.
for i in range(10):
  print("Sketch Image:")
  cv2_imshow(img_arr[i])
  print("Original Color Image:")
  cv2_imshow(col_arr[i])
  print("Model Generated Color Image:")
  cv2_imshow(res[i])
```

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:



Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

In [0]:
```
#Load test data - Sketch.
img_tst = []

for j in  range(1):
  i = j + 25
  folder=i.__str__().zfill(4)
  loc = 'sketch-pair/danbooru-sketch-pair-128x/color/sketch/' + folder + '/'
  print('processing folder',folder)
  for filename in os.listdir(loc):
    img_tst.append(cv2.imread(loc+filename,1))
```

```
processing folder 0025
```

In [0]:
```
#Load test data - Color.
col_tst = []

for j in  range(1):
  i = j + 25
  folder=i.__str__().zfill(4)
  loc = 'sketch-pair/danbooru-sketch-pair-128x/color/src/' + folder + '/'
  print('processing folder',folder)
  for filename in os.listdir(loc):
    col_tst.append(cv2.imread(loc+filename,1))
```

```
processing folder 0025
```

In [0]:
```
#Converting test data as numpy array.
col_tst = np.array(col_tst, dtype='uint8')
img_tst = np.array(img_tst, dtype='uint8')
```

In [0]:
```
#Predict on test data.
tst = img_tst[0:10]
res = model.predict(tst)
```

In [0]:
```
#Print result on test data.
for i in range(10):
  print("Sketch Image:")
  cv2_imshow(img_tst[i])
  print("Original Color Image:")
  cv2_imshow(col_tst[i])
  print("Model Generated Color Image:")
  cv2_imshow(res[i])
```

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

In [0]:
```
#Ran model on couple of more epochs to check if model is improving.
model.fit(img_arr,col_arr, epochs=50, batch_size=128, validation_split=0.1)
```

```
Train on 44540 samples, validate on 4949 samples
Epoch 1/50
44540/44540 [==============================] - 99s 2ms/step - loss: 429.7041
- val_loss: 4535.2155
Epoch 2/50
44540/44540 [==============================] - 99s 2ms/step - loss: 527.8133
- val_loss: 3822.2476
Epoch 3/50
44540/44540 [==============================] - 99s 2ms/step - loss: 435.8667
- val_loss: 3451.6853
Epoch 4/50
44540/44540 [==============================] - 99s 2ms/step - loss: 423.1363
- val_loss: 3491.5013
Epoch 5/50
44540/44540 [==============================] - 99s 2ms/step - loss: 435.4601
- val_loss: 4023.7424
Epoch 6/50
44540/44540 [==============================] - 99s 2ms/step - loss: 422.7055
- val_loss: 3578.7343
Epoch 7/50
44540/44540 [==============================] - 99s 2ms/step - loss: 425.9501
- val_loss: 4868.5509
Epoch 8/50
44540/44540 [==============================] - 99s 2ms/step - loss: 612.7867
- val_loss: 3455.5539
Epoch 9/50
44540/44540 [==============================] - 99s 2ms/step - loss: 433.8966
- val_loss: 3647.2949
Epoch 10/50
44540/44540 [==============================] - 99s 2ms/step - loss: 420.3842
- val_loss: 3412.2513
Epoch 11/50
44540/44540 [==============================] - 99s 2ms/step - loss: 417.8572
- val_loss: 3505.3448
Epoch 12/50
44540/44540 [==============================] - 99s 2ms/step - loss: 421.0102
- val_loss: 3691.3650
Epoch 13/50
44540/44540 [==============================] - 99s 2ms/step - loss: 422.3709
- val_loss: 3474.3437
Epoch 14/50
44540/44540 [==============================] - 99s 2ms/step - loss: 419.5429
- val_loss: 3425.0236
Epoch 15/50
44540/44540 [==============================] - 99s 2ms/step - loss: 512.7872
- val_loss: 5662.4831
Epoch 16/50
44540/44540 [==============================] - 99s 2ms/step - loss: 446.2543
- val_loss: 3588.2777
Epoch 17/50
44540/44540 [==============================] - 99s 2ms/step - loss: 415.7253
- val_loss: 3470.4172
Epoch 18/50
44540/44540 [==============================] - 99s 2ms/step - loss: 414.8269
- val_loss: 3446.5136
Epoch 19/50
44540/44540 [==============================] - 99s 2ms/step - loss: 414.4282
```

```
- val_loss: 3458.6294
Epoch 20/50
44540/44540 [==============================] - 99s 2ms/step - loss: 419.8763
- val_loss: 3549.1867
Epoch 21/50
44540/44540 [==============================] - 99s 2ms/step - loss: 426.4634
- val_loss: 3579.9213
Epoch 22/50
39680/44540 [=========================>....] - ETA: 10s - loss: 425.4926
```

5. GAN Model to generate Color Images

In [0]:
```
# Load Sketches.
img_arr = []  #Array to store pixels of sketches.
img_name = [] #Array to store name of the image.

for i in  range(6): #Processing first 15 folders.
  folder=i.__str__().zfill(4) #Printing Folder number in 4 byte format.
  loc = 'sketch-pair/danbooru-sketch-pair-128x/color/sketch/' + folder + '/' #
location of sketch images.
  print('processing folder',folder)   #Print the folder name currently getting
processed.
  for filename in os.listdir(loc):    #Process each image from the current fol
der.
    img_arr.append(cv2.imread(loc+filename,1))    #Append pixel value of each
 image.
    img_name.append(folder+'_'+filename)          #Append name of each image.
```

```
processing folder 0000
processing folder 0001
processing folder 0002
processing folder 0003
processing folder 0004
processing folder 0005
```

In [0]:
```
#Print total number of image loaded.
len(img_arr)
```

Out[0]:
```
11875
```

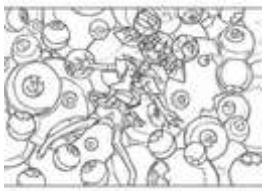
In [0]:
```
# Load Color Images.
col_arr = []  #Array to store pixels of color images.
col_name = [] #Array to store name of the image.

for i in  range(6): #Processing first 15 folders.
  folder=i.__str__().zfill(4) #Printing Folder number in 4 byte format.
  loc = 'sketch-pair/danbooru-sketch-pair-128x/color/src/' + folder + '/' #loc
ation of color images.
  print('processing folder',folder) #Print the folder name currently getting p
rocessed.
  for filename in os.listdir(loc):  #Process each image from the current folde
r.
    col_arr.append(cv2.imread(loc+filename,1))   #Append pixel value of each i
mage.
    col_name.append(folder+'_'+filename)         #Append name of each image.
```

```
processing folder 0000
processing folder 0001
processing folder 0002
processing folder 0003
processing folder 0004
processing folder 0005
```

In [0]:
```
#Convert the pixel variables into numpy array.
col_arr = np.array(col_arr, dtype='uint8')
img_arr = np.array(img_arr, dtype='uint8')
```

In [0]:
```
#Normalize the pixel value to be between -1 & +1.
col_arr = (col_arr - 127.5) / 127.5
img_arr = (img_arr - 127.5) / 127.5
```

In [0]:
```
#Restoring the pixel values and generating images to ensure we haven't lost an
y info.
for i in range(10):
  cv2_imshow(((img_arr[i] + 1)/2.0)*255.0)
  cv2_imshow(((col_arr[i] + 1)/2.0)*255.0)
```



In [0]:
```
#Defining a Random Gaussian variable to initialize weight for Conv layers.
from keras.initializers import RandomNormal
init = RandomNormal(stddev=0.02)
```

In [0]:
```
#Generator Model with U-net architecture.
#For the Generator Architecture referred - https://github.com/kvfrans/deepcolo
r/blob/master/main.py.
#The Code above was in tensor-flow. So, changed that to a Keras model.

def build_generator():

    #Encoder Unit
    #Input Layer (128, 128, 3)
    encoder_input = Input(shape=(128, 128, 3,))
    #Layer-1 Output image of shape - 64 * 64 * 64
    encoder_output_1 = Conv2D(64, (5,5), padding='same',  strides=2, kernel_in
itializer=init)(encoder_input)
    #Leaky ReLUs allow a small, non-zero gradient when the unit is not active.
    encoder_output_2 = LeakyReLU(alpha=0.2)(encoder_output_1)
    #Layer-2 Output image of shape - 32 * 32 * 128
    encoder_output_2 = Conv2D(128, (5,5),  padding='same', strides=2, kernel_i
nitializer=init)(encoder_output_2)
    #Batch Normalization.
    encoder_output_2 = BatchNormalization(momentum=0.9)(encoder_output_2)
    #Layer-3 Output image of shape - 16 * 16 * 256
    encoder_output_3 = LeakyReLU(alpha=0.2)(encoder_output_2)
    encoder_output_3 = Conv2D(256, (5,5),  padding='same',  strides=2, kernel_
initializer=init)(encoder_output_3)
    encoder_output_3 = BatchNormalization(momentum=0.9)(encoder_output_3)
    #Layer-4 Output image of shape - 8 * 8 * 512
    encoder_output_4 = LeakyReLU(alpha=0.2)(encoder_output_3)
    encoder_output_4 = Conv2D(512, (5,5),  padding='same',  strides=2, kernel_
initializer=init)(encoder_output_4)
    encoder_output_4 = BatchNormalization(momentum=0.9)(encoder_output_4)
    #Layer-5 Output image of shape - 4 * 4 * 512
    encoder_output_5 = LeakyReLU(alpha=0.2)(encoder_output_4)
    encoder_output_5 = Conv2D(512, (5,5),  padding='same',  strides=2, kernel_
initializer=init)(encoder_output_5)
    encoder_output_5 = BatchNormalization(momentum=0.9)(encoder_output_5)



    #Decoder Unit
    #Conv2DTranspose - The need for transposed convolutions generally arises f
rom the desire to use a transformation going in the opposite
    #direction of a normal convolution, i.e., from something that has the shap
e of the output of some convolution to
    #something that has the shape of its input while maintaining a connectivit
y pattern that is compatible with said convolution.
    #Ref- https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2DTra
nspose
    #Layer-6 Output image of shape - 8 * 8 * 512
    encoder_output_6 = Conv2DTranspose(512, (5,5), padding='same', activation=
'relu', strides=2, kernel_initializer=init )(encoder_output_5)
    encoder_output_6  = BatchNormalization(momentum=0.9)(encoder_output_6)
    #Concatenating output from previous layer with output from layer-4. Both a
re of same dimension.
    encoder_output_6 = Add()([encoder_output_6, encoder_output_4])
    #Layer-7 Output image of shape - 16 * 16 * 256
    encoder_output_7 = Conv2DTranspose(256, (5,5), padding='same', activation=
```

```
'relu', strides=2, kernel_initializer=init)(encoder_output_6)
    encoder_output_7  = BatchNormalization(momentum=0.9)(encoder_output_7)
    #Concatenating output from previous layer with output from layer-3. Both a
re of same dimension.
    encoder_output_7 = Add()([encoder_output_7, encoder_output_3])
    #Layer-8 Output image of shape - 32 * 32 * 128
    encoder_output_8 = Conv2DTranspose(128, (5,5), padding='same', activation=
'relu', strides=2, kernel_initializer=init)(encoder_output_7)
    encoder_output_8  = BatchNormalization(momentum=0.9)(encoder_output_8)
    #Concatenating output from previous layer with output from layer-2. Both a
re of same dimension.
    encoder_output_8 = Add()([encoder_output_8, encoder_output_2])
    #Layer-9 Output image of shape - 64 * 64 * 64
    encoder_output_9 = Conv2DTranspose(64, (5,5), padding='same', activation=
'relu', strides=2, kernel_initializer=init)(encoder_output_8)
    encoder_output_9  = BatchNormalization(momentum=0.9)(encoder_output_9)
    #Concatenating output from previous layer with output from layer-1. Both a
re of same dimension
    encoder_output_9 = Add()([encoder_output_9, encoder_output_1])
    #Layer-10 (Output Layer) Output image of shape - 128 * 128 * 3
    encoder_output = Conv2DTranspose(3, (5,5), padding='same', activation='tan
h', strides=2, kernel_initializer=init, name="gen_output")(encoder_output_9)

    #Defining Model with input and Outputs.
    model = Model(encoder_input, encoder_output)
    #Return Model.
    return model
```

In [0]:
```
#Discriminator Model.
#For the Discriminator Architecture referred - https://github.com/kvfrans/deep
color/blob/master/main.py.
#The Code above was in tensor-flow. So, changed that to a Keras model.

def build_discriminator():

    #Input Layer (128, 128, 3)
    encoder_input = Input(shape=(128, 128, 3,))
    #Layer-1 Output image of shape - 64 * 64 * 64
    encoder_output_1 = Conv2D(64, (5,5), padding='same',  strides=2, kernel_in
itializer=init)(encoder_input)
    #Leaky ReLUs allow a small, non-zero gradient when the unit is not active.
    encoder_output_2 = LeakyReLU(alpha=0.2)(encoder_output_1)
    #Layer-2 Output image of shape - 32 * 32 * 128
    encoder_output_2 = Conv2D(128, (5,5),  padding='same', strides=2, kernel_i
nitializer=init)(encoder_output_2)
    encoder_output_2 = BatchNormalization(momentum=0.9)(encoder_output_2)
    #Layer-3 Output image of shape - 16 * 16 * 256
    encoder_output_3 = LeakyReLU(alpha=0.2)(encoder_output_2)
    encoder_output_3 = Conv2D(256, (5,5),  padding='same',  strides=2, kernel_
initializer=init)(encoder_output_3)
    encoder_output_3 = BatchNormalization(momentum=0.9)(encoder_output_3)
    #Layer-4 Output image of shape - 8 * 8 * 512
    encoder_output_4 = LeakyReLU(alpha=0.2)(encoder_output_3)
    encoder_output_4 = Conv2D(512, (5,5),  padding='same',  strides=2, kernel_
initializer=init)(encoder_output_4)
    encoder_output_4 = BatchNormalization(momentum=0.9)(encoder_output_4)
    encoder_output_5 = LeakyReLU(alpha=0.2)(encoder_output_4)
    #Flatten Output to apply Dense layer.
    encoder_output_5 = Flatten()(encoder_output_5)
    #Activation unit with Sigmoid layer.
    encoder_output = Dense(units=1, activation='sigmoid', name="disc_output")(
encoder_output_5)
    #Defining Model with input and Outputs.
    model = Model(encoder_input, encoder_output)
    #Adam Optimizer.
    opt = Adam(lr=0.0002, beta_1=0.5)
    #Compile the model.
    model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accurac
y'])
    #Return Model.
    return model


#Call the Discriminator model.
d =build_discriminator()


#Print the Summary.
d.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:66: The name tf.get_default_graph is deprecated. Please
use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please use
tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:4409: The name tf.random_normal is deprecated. Please u
se tf.random.normal instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:190: The name tf.get_default_session is deprecated. Ple
ase use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:197: The name tf.ConfigProto is deprecated. Please use
tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:203: The name tf.Session is deprecated. Please use tf.c
ompat.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:207: The name tf.global_variables is deprecated. Please
use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:216: The name tf.is_variable_initialized is deprecated.
Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:223: The name tf.variables_initializer is deprecated. P
lease use tf.compat.v1.variables_initializer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:2041: The name tf.nn.fused_batch_norm is deprecated. Pl
ease use tf.compat.v1.nn.fused_batch_norm instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:148: The name tf.placeholder_with_default is deprecate
d. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:4432: The name tf.random_uniform is deprecated. Please
use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimize
rs.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compat.v
1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:3657: The name tf.log is deprecated. Please use tf.mat
h.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_cor
```

```
e/python/ops/nn_impl.py:183: where (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "model_1"
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 128, 128, 3)	0
conv2d_1 (Conv2D)	(None, 64, 64, 64)	4864
leaky_re_lu_1 (LeakyReLU)	(None, 64, 64, 64)	0
conv2d_2 (Conv2D)	(None, 32, 32, 128)	204928
batch_normalization_1 (Batch	(None, 32, 32, 128)	512
leaky_re_lu_2 (LeakyReLU)	(None, 32, 32, 128)	0
conv2d_3 (Conv2D)	(None, 16, 16, 256)	819456
batch_normalization_2 (Batch	(None, 16, 16, 256)	1024
leaky_re_lu_3 (LeakyReLU)	(None, 16, 16, 256)	0
conv2d_4 (Conv2D)	(None, 8, 8, 512)	3277312
batch_normalization_3 (Batch	(None, 8, 8, 512)	2048
leaky_re_lu_4 (LeakyReLU)	(None, 8, 8, 512)	0
flatten_1 (Flatten)	(None, 32768)	0
disc_output (Dense)	(None, 1)	32769

```
Total params: 4,342,913
Trainable params: 4,341,121
Non-trainable params: 1,792
```

In [0]:
```
#Defining Loss function to calculate L1 loss at pixel level.
#Ref - https://towardsdatascience.com/sketch-to-color-anime-translation-using-generative-adversarial-networks-gans-8f4f69594aeb

import tensorflow.keras.backend as K

def pixLevel_loss(y, g): #Arguments  - y (Real target images), g (Output Images from generator model)

  #Defining a Pixel level function that would be returned
  def finalPLLoss(y_true, y_pred):
    return K.mean( K.abs( y - g ) )

  #Return loss function of the prototype keras requires.
  return finalPLLoss
```

In [0]:
```
#Define the GAN function.
#Ref - https://medium.com/datadriveninvestor/generative-adversarial-network-gan-using-keras-ce1c05cfdfd3 to create a sample GAN module.
#Ref- https://towardsdatascience.com/sketch-to-color-anime-translation-using-generative-adversarial-networks-gans-8f4f69594aeb to add L1 pixel level loss.

def create_gan(discriminator, generator): #Arguments to be passed  - Discriminator & Generator models.
    #Setting trainable to False, so that discriminator weights doesn't get updated while training GANs.
    discriminator.trainable=False

    #Input - Sketch Image.
    gan_input = Input(shape=(128,128,3))

    #Execute generator model to generate predicted image.
    x = generator(gan_input)

    #Input - Color Image.
    col_input = Input(shape=(128,128,3))

    #Call function to calculate pixel level loss.
    pixLevelLoss = pixLevel_loss(col_input, x)

    #Predict image generated thru Generator whether its real or fake.
    gan_output= discriminator(x)

    #Define model with inputs and outputs.
    gan= Model(inputs=[gan_input, col_input], outputs=gan_output)

    #Adam Optimizer.
    opt = Adam(lr=0.0002, beta_1=0.5)

    #Compile model, with weighted loss as 1 * binary_crossentropy +  100  *  pixel level loss.
    gan.compile(optimizer=opt, loss=lambda y_true, y_pred : losses.binary_crossentropy(y_true, y_pred) + \
                100 *  pixLevelLoss(y_true, y_pred), metrics=['accuracy'])

    #Return GAN model.
    return gan

#Execute GAN model.
gan = create_gan(build_discriminator(),build_generator())

#Print Summary of the model.
gan.summary()
```

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         (None, 128, 128, 3)       0
_________________________________________________________________
model_3 (Model)              (None, 128, 128, 3)       21730307
_________________________________________________________________
model_2 (Model)              (None, 1)                 4342913
=================================================================
Total params: 26,073,220
Trainable params: 21,725,571
Non-trainable params: 4,347,649
_________________________________________________________________
```

In [0]:

```
#Function to generate batch of real and generated images for model training.

TOTAL_IMAGES = len(img_arr) #Get total number of images.

def generate_samples(skc, col, n_samples, generator):
#Arguments.
#skc       - Array of sketches as input.
#col       - Array of Original Color Outputs.
#n_samples - Number of samples to be generated.
#generator - Generator model to generate image.

  #Generate n random numbers between the range of total images.
  ix = np.random.randint(0, TOTAL_IMAGES, n_samples)

  #Arrays to store batch sample of sketches, color images and generated image
s.
  X_sketches = []
  X_color = []
  X_res = []

  #Loop to generate sketeches and color image arrays.
  for i in ix:
    X_sketches.append(skc[i])
    X_color.append(col[i])

  #Converting list to arrays.
  X_sketches = np.array(X_sketches)
  X_color = np.array(X_color)

  #Generate color images.
  X_res = generator.predict(X_sketches)

  #return batch of sketches, color images, and generated images.
  return X_sketches, X_color, X_res
```

In [0]:
```
#Plot model performance.
def plot_history(d_loss_hist, g_loss_hist, d_acc_hist, g_acc_hist):
# plot discriminator loss
 pyplot.subplot(1, 1, 1)
 pyplot.plot(d_loss_hist, label='Loss-dis')
 pyplot.gca().set_title("Discriminator Loss")
 pyplot.legend()
 pyplot.show()
 # plot Generator loss
 pyplot.subplot(1, 1, 1)
 pyplot.plot(g_loss_hist, label='Loss-gen')
 pyplot.gca().set_title("Generator Loss")
 pyplot.legend()
 pyplot.show()
 # plot discriminator accuracy
 pyplot.subplot(1, 1, 1)
 pyplot.plot(d_acc_hist, label='acc-dis')
 pyplot.gca().set_title("Discriminator Accuracy")
 pyplot.legend()
 pyplot.show()
 # plot discriminator accuracy
 pyplot.subplot(1, 1, 1)
 pyplot.plot(g_acc_hist, label='acc-gen')
 pyplot.gca().set_title("Generator Accuracy")
 pyplot.legend()
 pyplot.show()
```

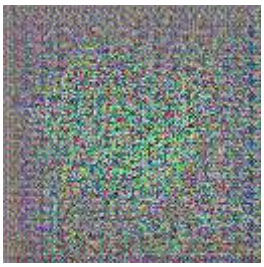
In [0]:
```
#Show generated images.
def summarize_performance(g_model):
  tst = img_arr[0:3]
  res = g_model.predict(tst)
  res = (res + 1) / 2.0
  res = res * 255.0
  for i in range(3):
    print("Model Generated Color Image:")
    cv2_imshow(res[i])
```

In [0]:
```
#Function to train the GAN model.
#Referred - https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/

from datetime import datetime



def training(epochs=1, batch_size=128):

    start_time = datetime.now()

    # Build Generator Model.
    generator= build_generator()

    # Build Discriminator Model.
    discriminator= build_discriminator()

    #Build GAN model.
    gan = create_gan(discriminator, generator)

    #Defining lists to store model performances.
    d_loss_hist, g_loss_hist, d_acc_hist, g_acc_hist = list(), list(), list(), list()

    #Train the model for given number of iterations.
    for e in range(epochs):

        #Half of batch size as discriminator model will be trained half on real images and half on generated images.
        half_batch = int(batch_size / 2)

        #Train discriminator only on alternate iterations.
        if not e%2:
           #Generate samples.
           skc, col, res = generate_samples(img_arr, col_arr, half_batch, generator)

           #Target variable for real images.
           y_real = np.ones((half_batch, 1))

           #Target variable for fake images.
           y_fake = np.zeros((half_batch, 1))

           #Make the discrimator model trainable.
           discriminator.trainable=True

           #Train discriminator on real images.
           d_loss_real, d_acc_real = discriminator.train_on_batch(col, y_real * .9)

           #Train discriminator on fake images.
           d_loss_fake, d_acc_fake = discriminator.train_on_batch(res, y_fake)

           #Calculate total loss
           d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
```

```
            #Calculate total accuracy.
            d_acc = 0.5 * np.add(d_acc_real, d_acc_fake)

            #Appending the values generated above.
            d_loss_hist.append(d_loss)
            d_acc_hist.append(d_acc)

        #Generate sample to train GAN model.
        skc, col, res = generate_samples(img_arr, col_arr, batch_size, generat
or)

        #Generate target variable considering all the generated images as rea
l.
        y_real = np.ones((batch_size, 1))

        #Train the model.
        g_loss, g_acc = gan.train_on_batch([skc,col], y_real)

        #Append performances.
        g_loss_hist.append(g_loss)
        g_acc_hist.append(g_acc)

        #Plot performance after each 20 epochs.
        #After every 15k iterations (20 epochs) displaying results and saving
 model.
        if e == 1 or e % 15000 == 0 or e% (epochs-1) == 0:
            print('Iteration number:',e)
            elapsed_time = datetime.now() - start_time
            print('Elapsed Time:', elapsed_time)
            print('Generator Loss:',g_loss)
            print('Discriminator Loss:',d_loss)
            print('Generator Accuracy:',g_acc)
            print('Discriminator Accuracy:',d_acc)
            if  (e % 15000 == 0 or e% (epochs-1) == 0) and (e != 0):
                plot_history(d_loss_hist, g_loss_hist, d_acc_hist, g_acc_hist)
            summarize_performance(generator)
            #Saving the model.
            generator.save_weights('model_gen_' + str(e) + '.h5')
            discriminator.save_weights('model_dis_' + str(e) + '.h5')
            print("Saved model to disk")
```

In [0]:
```
#Trained model for 195000 iterations (approx 260 epochs) with batch size of 1
6.
training(195000,16)
```

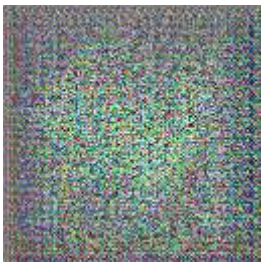
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Please use
tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:1020: The name tf.assign is deprecated. Please use tf.c
ompat.v1.assign instead.

Iteration number: 0
Elapsed Time: 0:00:17.976923
Generator Loss: 74.60054
Discriminator Loss: 2.1713900566101074
Generator Accuracy: 0.375
Discriminator Accuracy: 0.0
Model Generated Color Image:
```



```
Model Generated Color Image:
```

```
Model Generated Color Image:
```

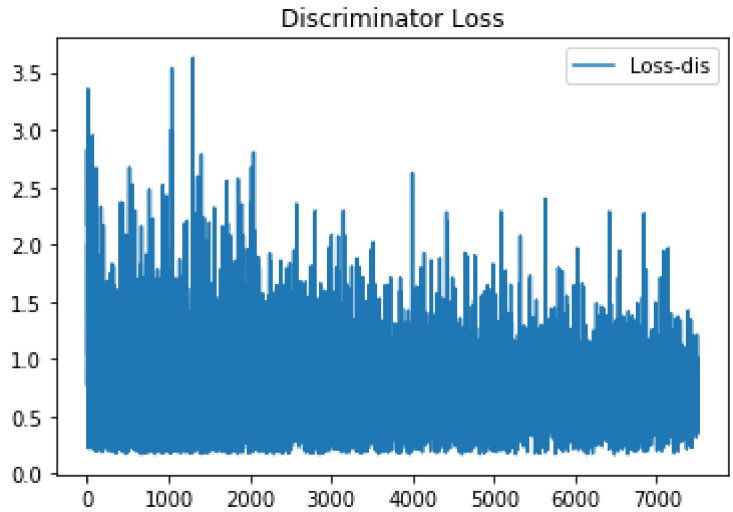
```
Saved model to disk
Iteration number: 1
Elapsed Time: 0:00:22.901588
Generator Loss: 76.787964
Discriminator Loss: 2.1713900566101074
Generator Accuracy: 1.0
Discriminator Accuracy: 0.0
Model Generated Color Image:
```

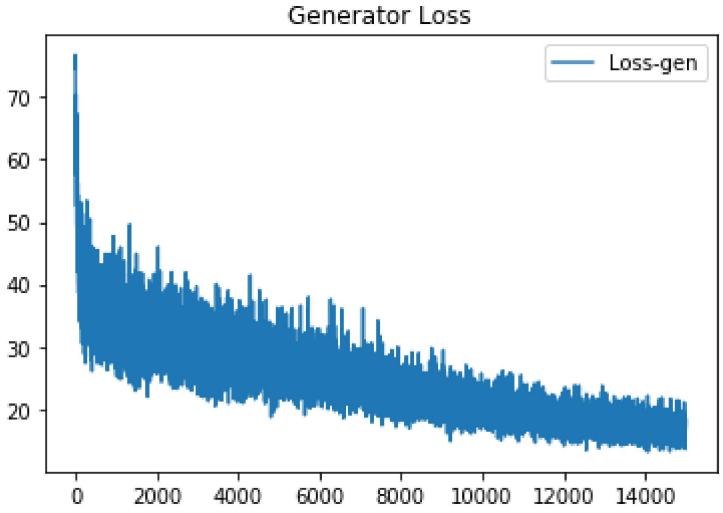


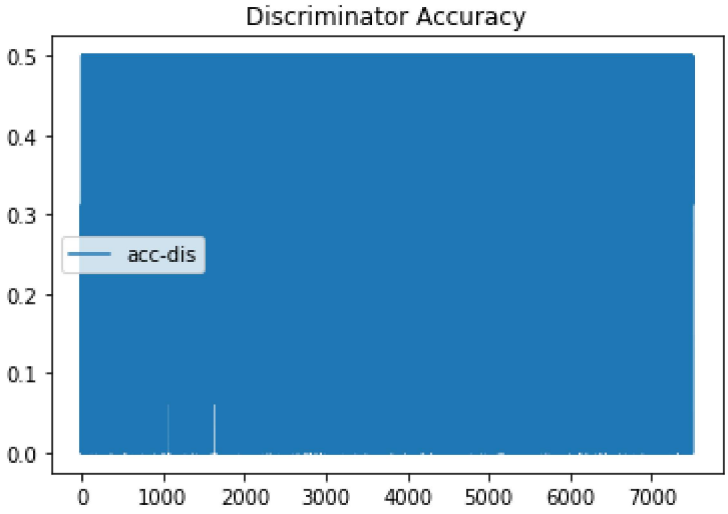
Model Generated Color Image:

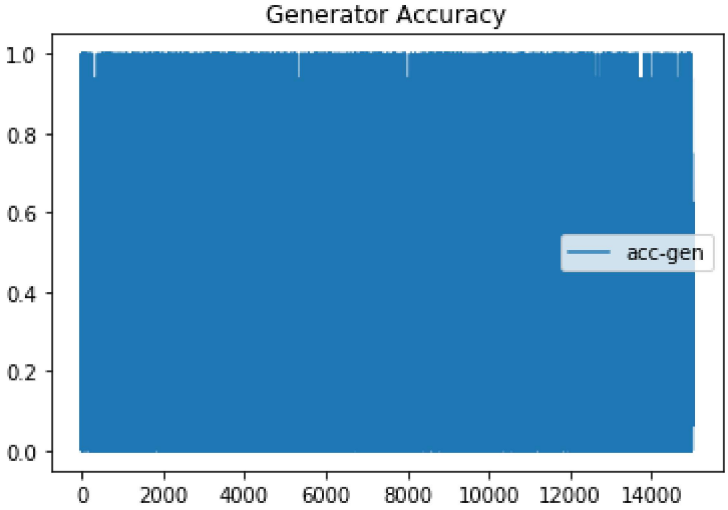
Model Generated Color Image:

```
Saved model to disk
Iteration number: 15000
Elapsed Time: 0:31:34.408158
Generator Loss: 17.31664
Discriminator Loss: 0.8395586609840393
Generator Accuracy: 0.25
Discriminator Accuracy: 0.0
```







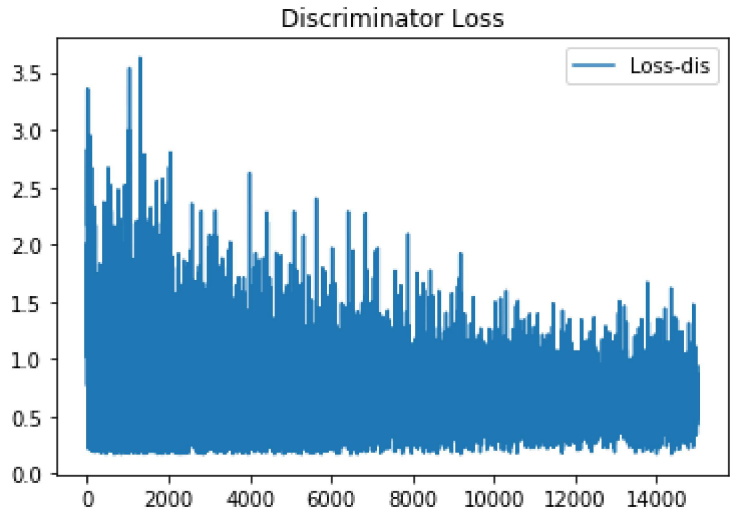


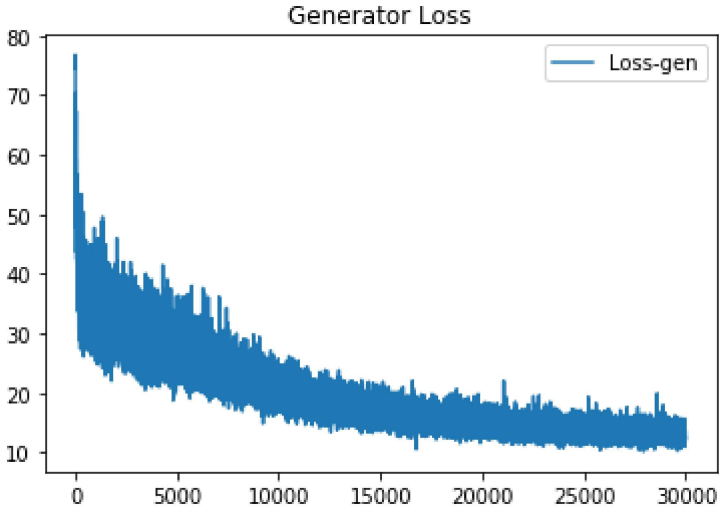
Model Generated Color Image:

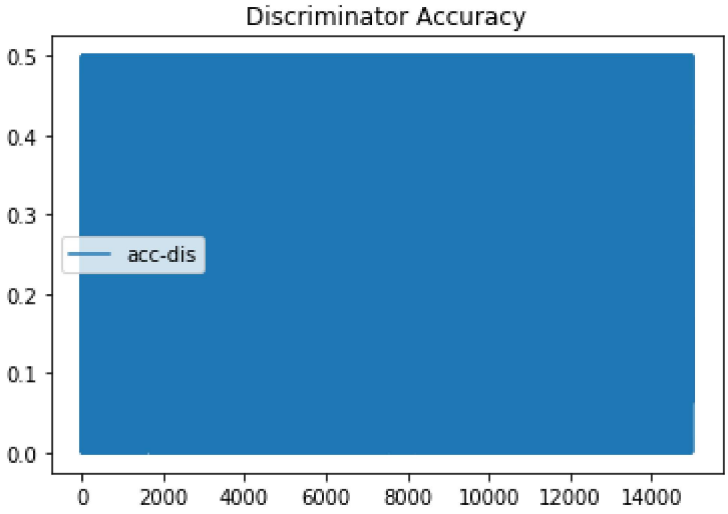
Model Generated Color Image:

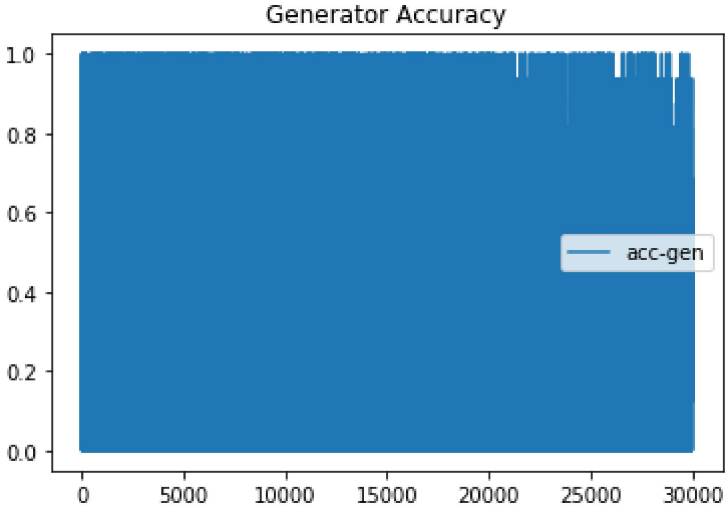
Model Generated Color Image:

Saved model to disk
Iteration number: 30000
Elapsed Time: 1:02:36.951986
Generator Loss: 12.50378
Discriminator Loss: 0.4920552968978882
Generator Accuracy: 0.3125
Discriminator Accuracy: 0.5







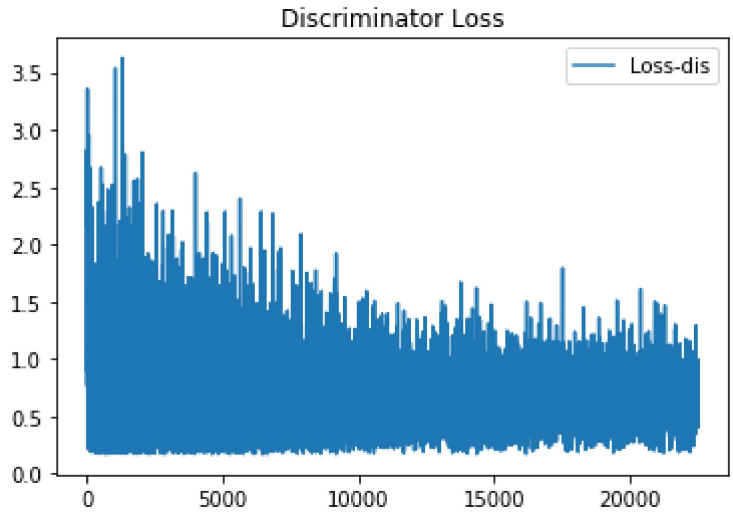


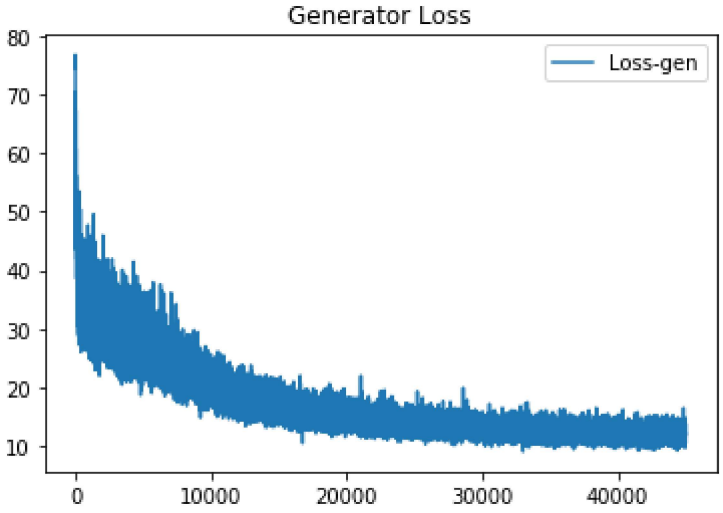
Model Generated Color Image:

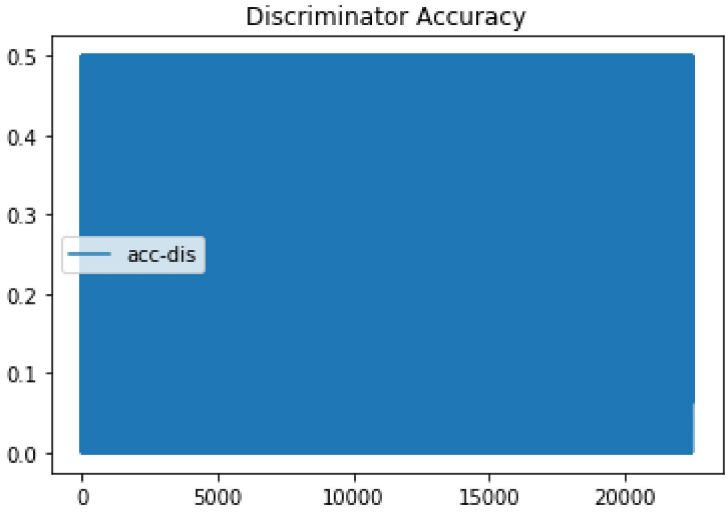
Model Generated Color Image:

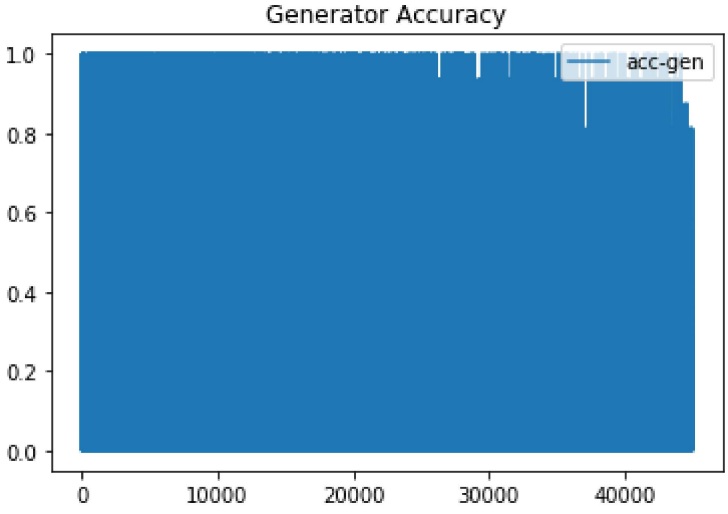
Model Generated Color Image:

```
Saved model to disk
Iteration number: 45000
Elapsed Time: 1:33:04.562023
Generator Loss: 12.114935
Discriminator Loss: 0.4042017459869385
Generator Accuracy: 0.0
Discriminator Accuracy: 0.5
```







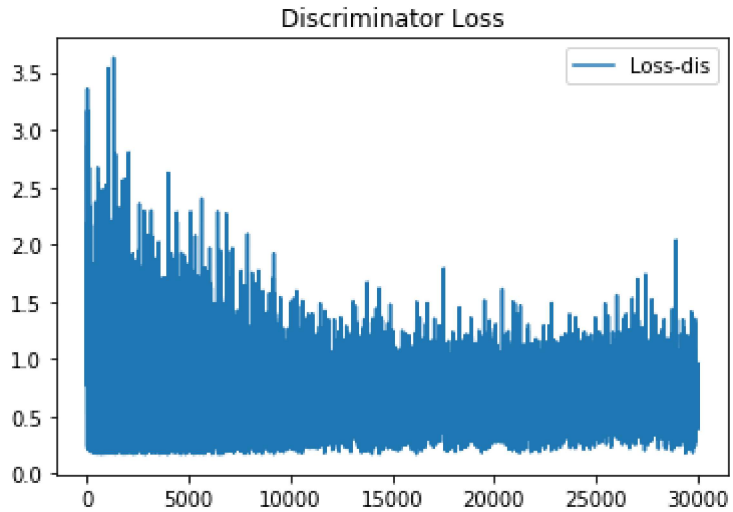


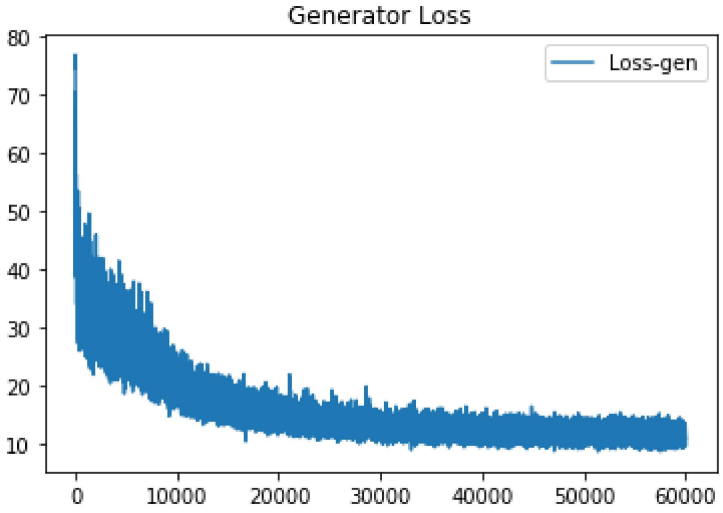
Model Generated Color Image:

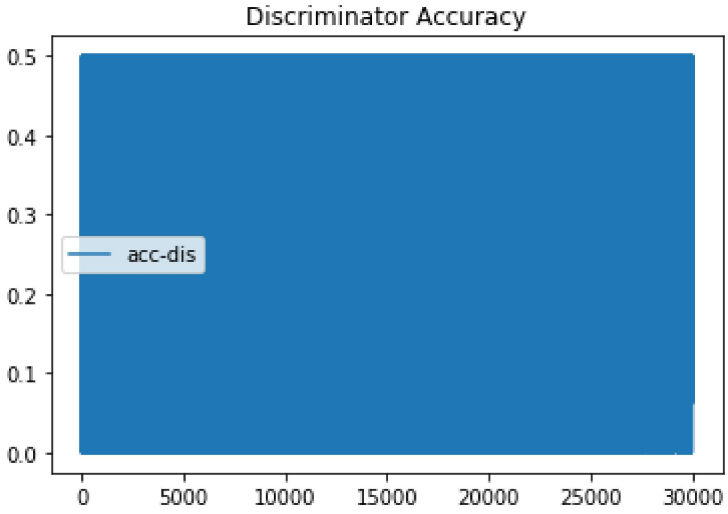
Model Generated Color Image:

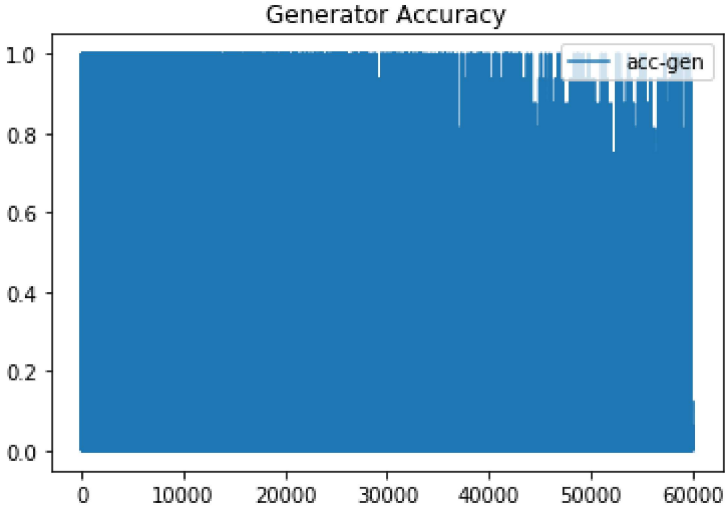
Model Generated Color Image:

Saved model to disk
Iteration number: 60000
Elapsed Time: 2:03:19.764126
Generator Loss: 11.056055
Discriminator Loss: 0.5151284337043762
Generator Accuracy: 0.0
Discriminator Accuracy: 0.5







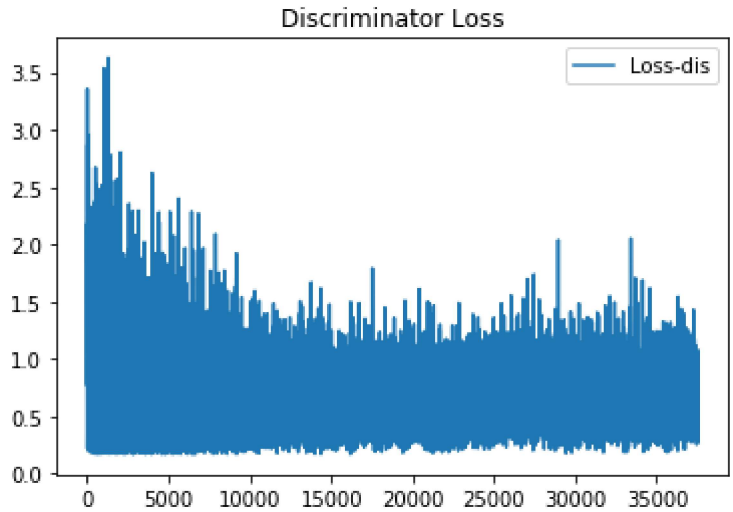


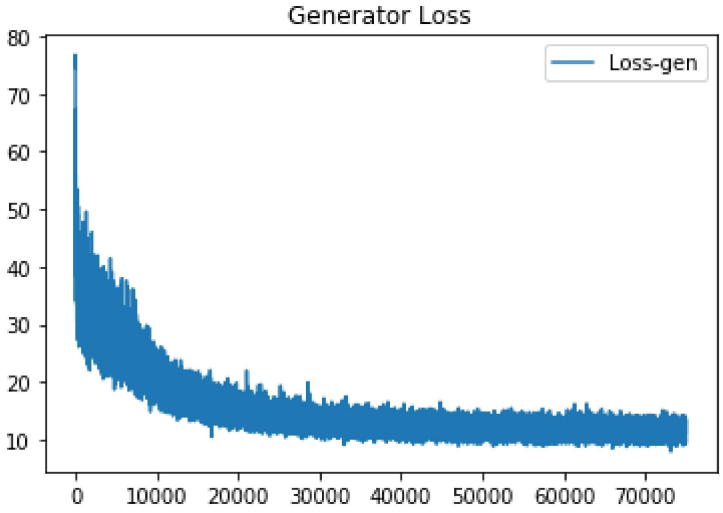
Model Generated Color Image:

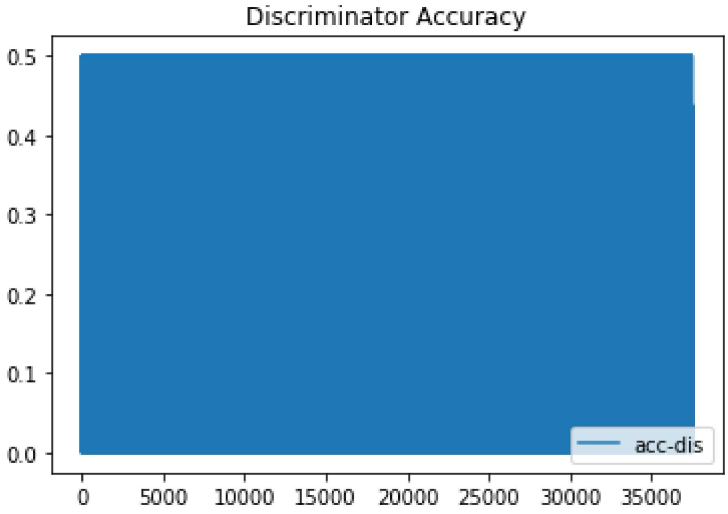
Model Generated Color Image:

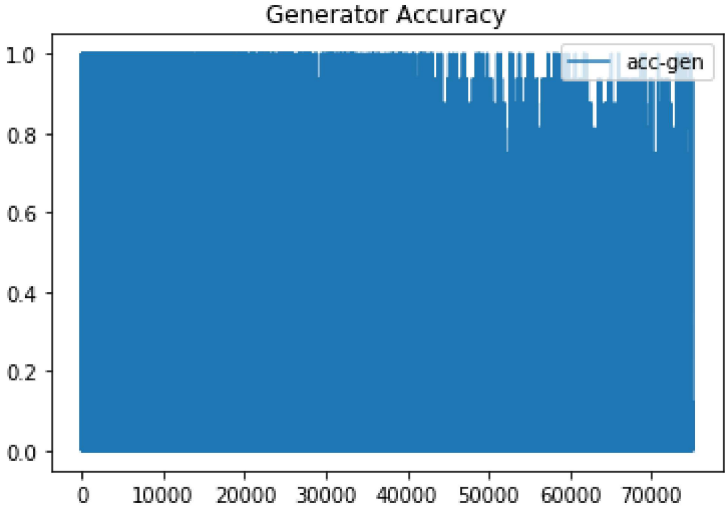
Model Generated Color Image:

Saved model to disk
Iteration number: 75000
Elapsed Time: 2:33:18.616149
Generator Loss: 13.515541
Discriminator Loss: 0.7256525754928589
Generator Accuracy: 0.0
Discriminator Accuracy: 0.1875







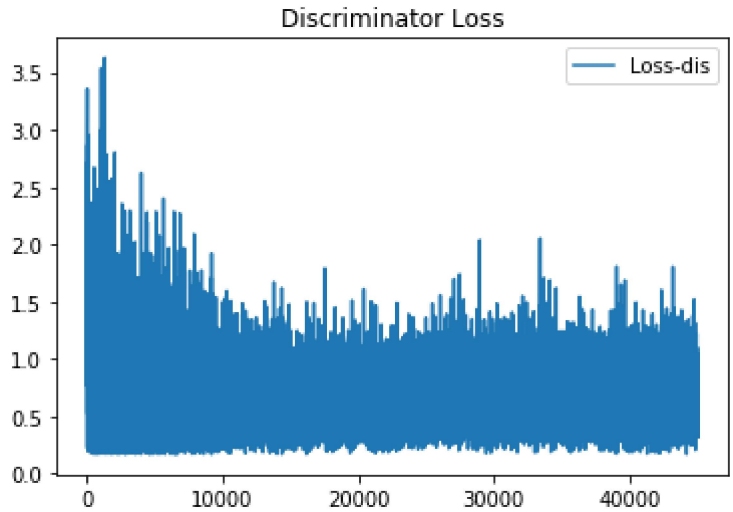


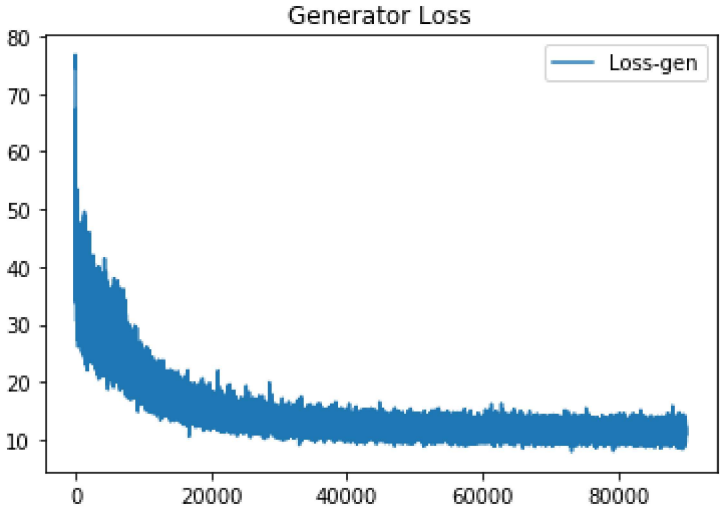
Model Generated Color Image:

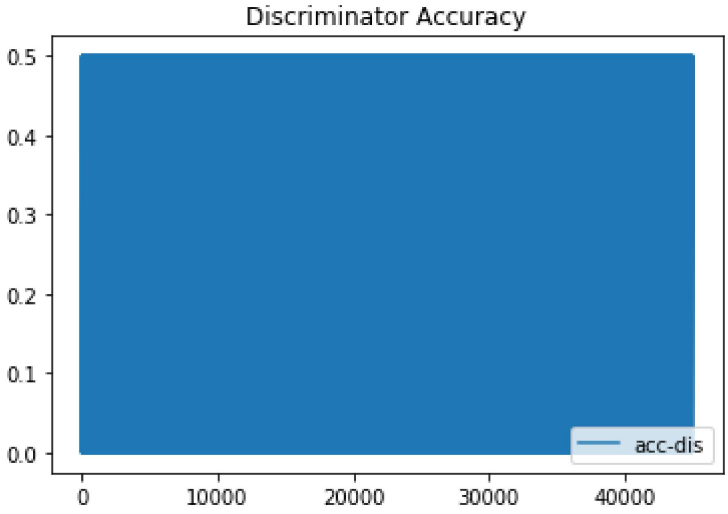
Model Generated Color Image:

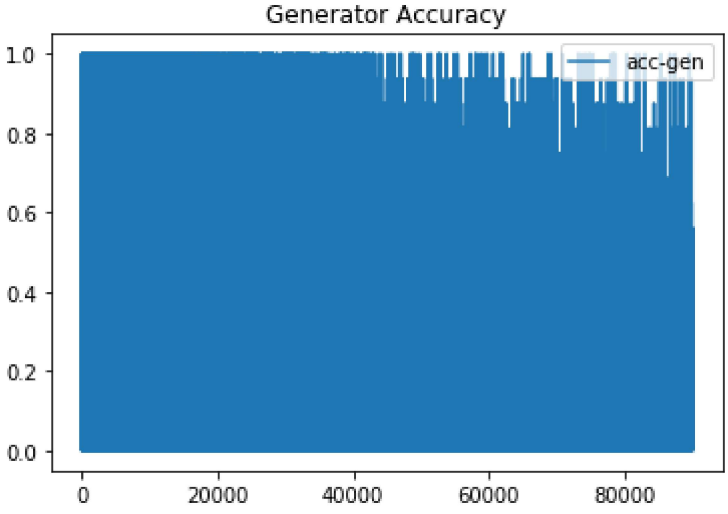
Model Generated Color Image:

Saved model to disk
Iteration number: 90000
Elapsed Time: 3:03:05.703308
Generator Loss: 12.266983
Discriminator Loss: 0.3944017291069031
Generator Accuracy: 0.0
Discriminator Accuracy: 0.5







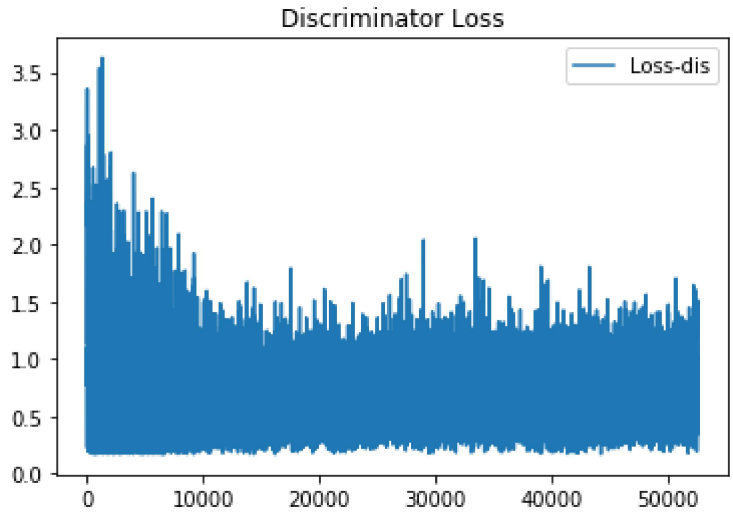


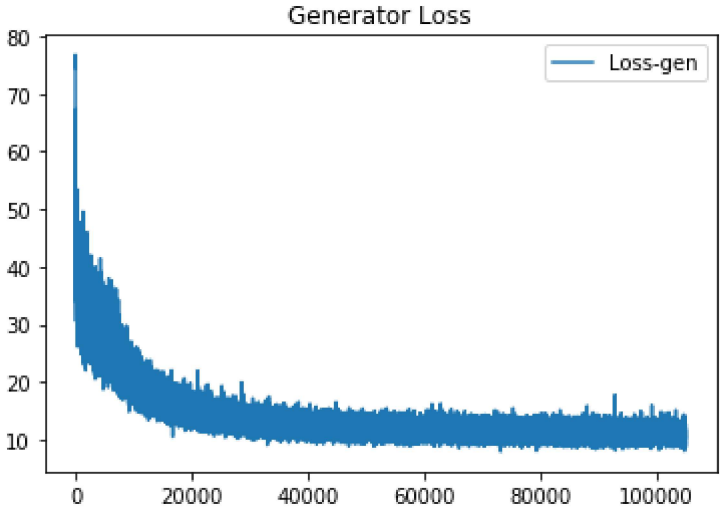
Model Generated Color Image:

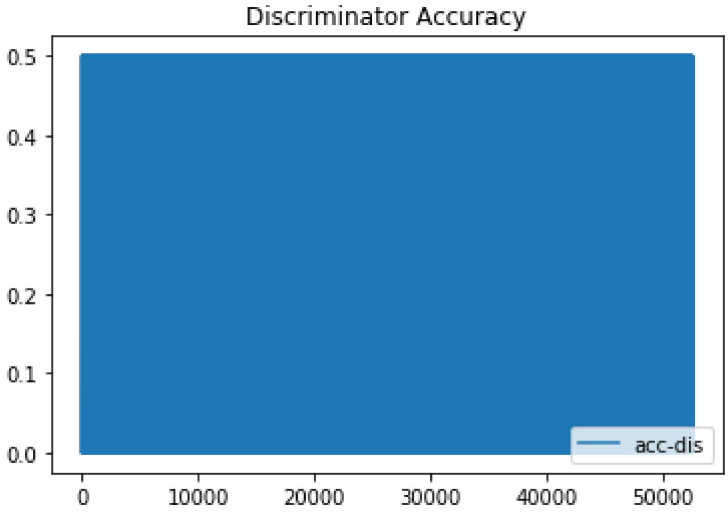
Model Generated Color Image:

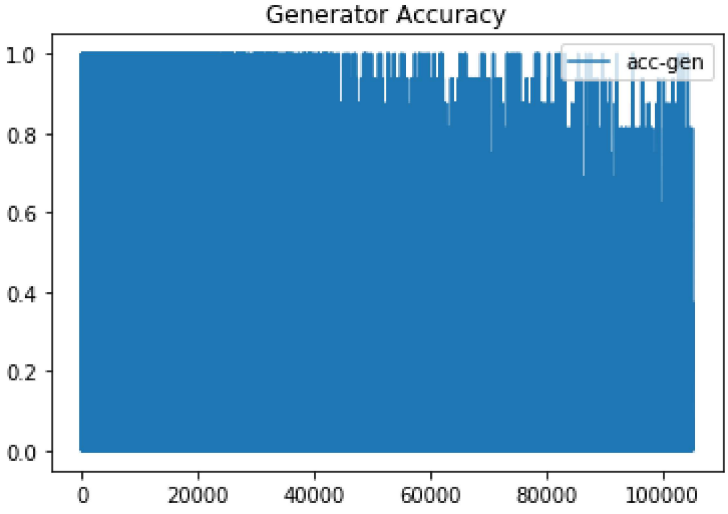
Model Generated Color Image:

```
Saved model to disk
Iteration number: 105000
Elapsed Time: 3:32:48.967303
Generator Loss: 11.524936
Discriminator Loss: 0.4991360902786255
Generator Accuracy: 0.0
Discriminator Accuracy: 0.375
```







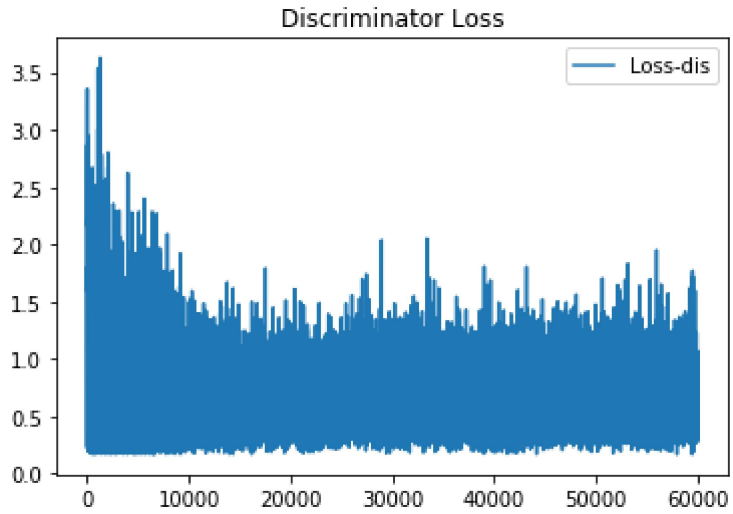


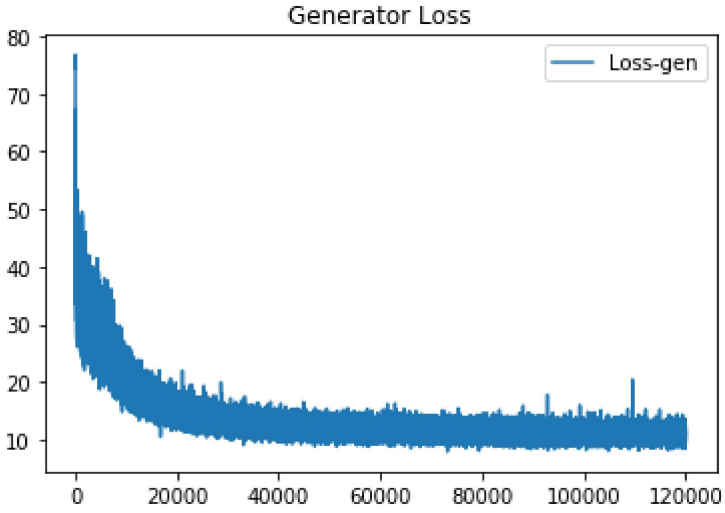
Model Generated Color Image:

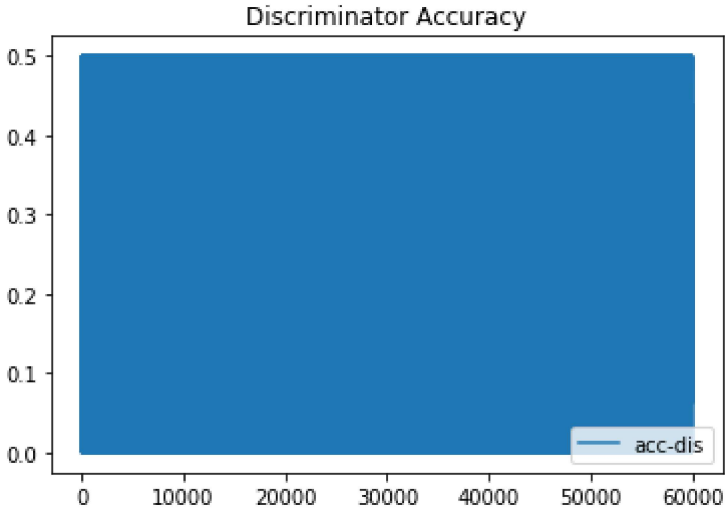
Model Generated Color Image:

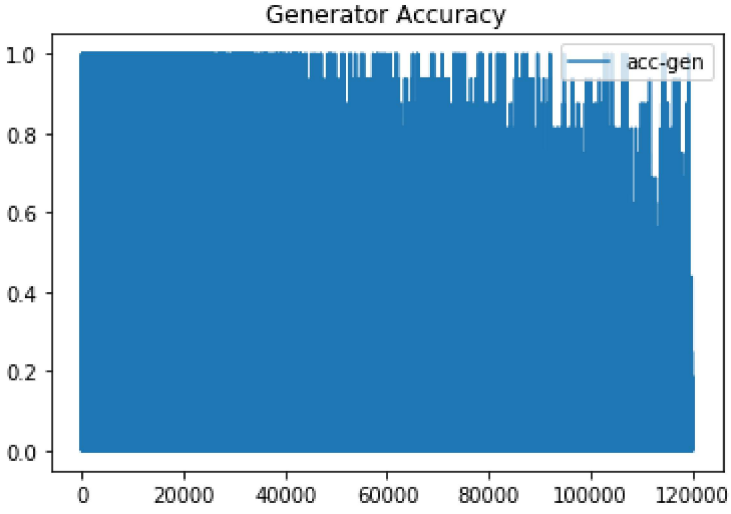
Model Generated Color Image:

Saved model to disk
Iteration number: 120000
Elapsed Time: 4:02:03.512759
Generator Loss: 10.842516
Discriminator Loss: 0.4437292516231537
Generator Accuracy: 0.0
Discriminator Accuracy: 0.4375







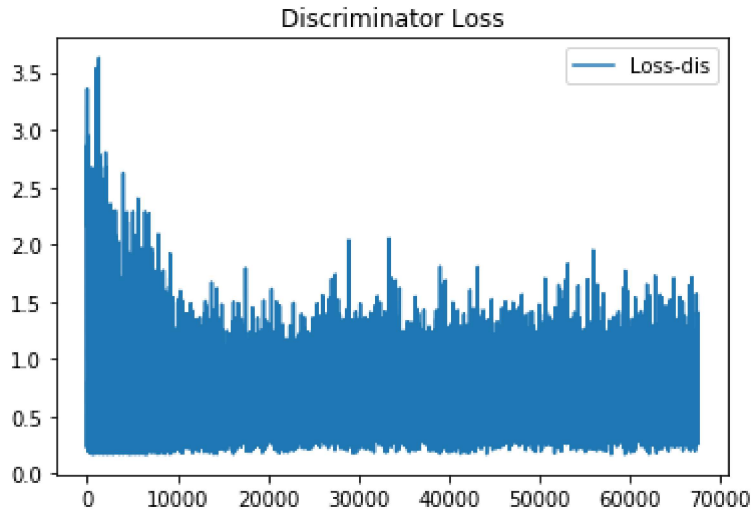


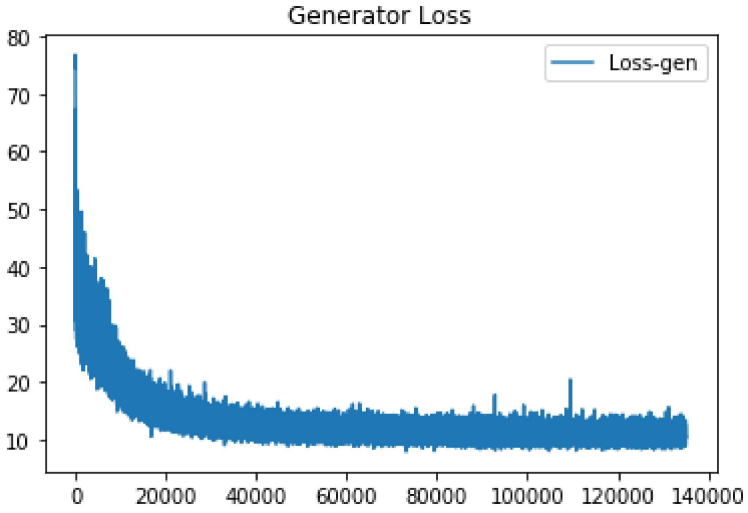
Model Generated Color Image:

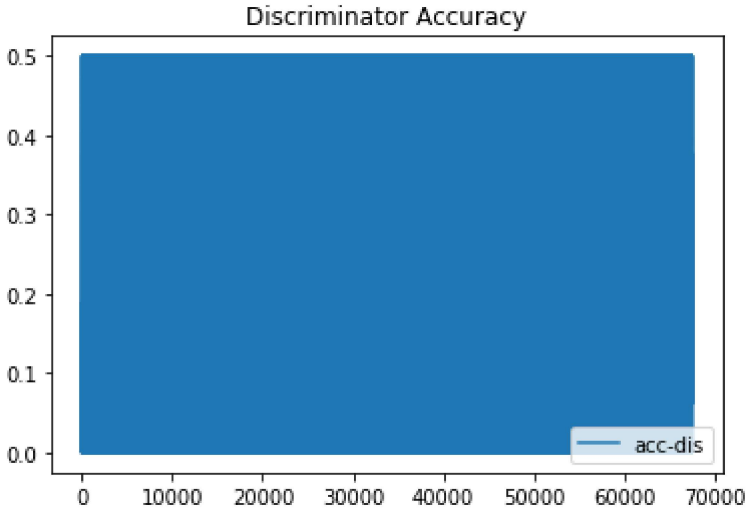
Model Generated Color Image:

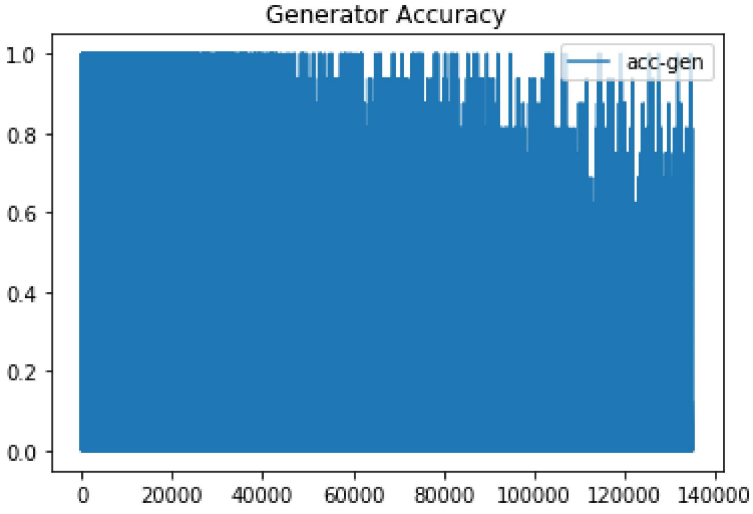
Model Generated Color Image:

Saved model to disk
Iteration number: 135000
Elapsed Time: 4:30:55.984668
Generator Loss: 10.358095
Discriminator Loss: 0.735002875328064
Generator Accuracy: 0.0
Discriminator Accuracy: 0.375







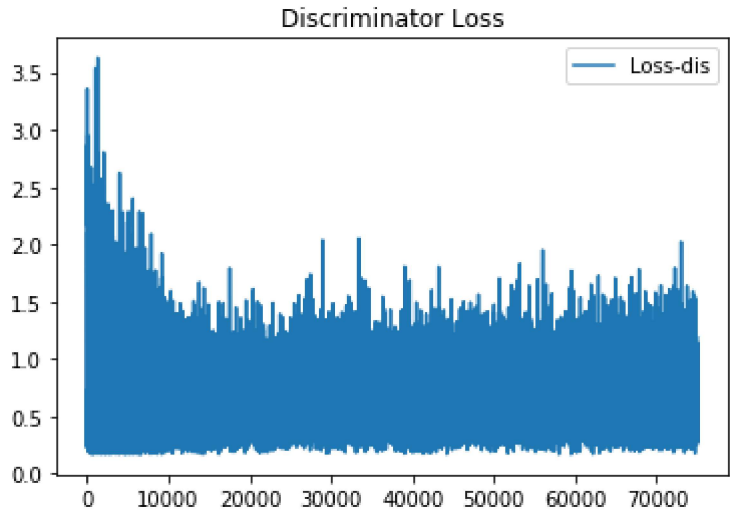


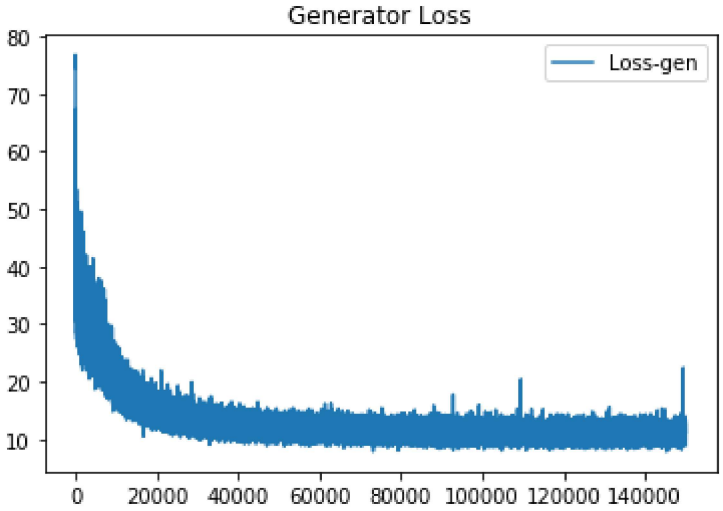
Model Generated Color Image:

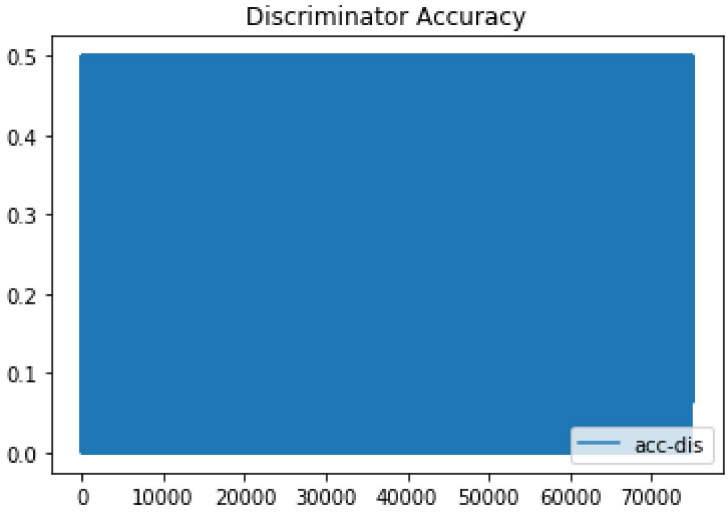
Model Generated Color Image:

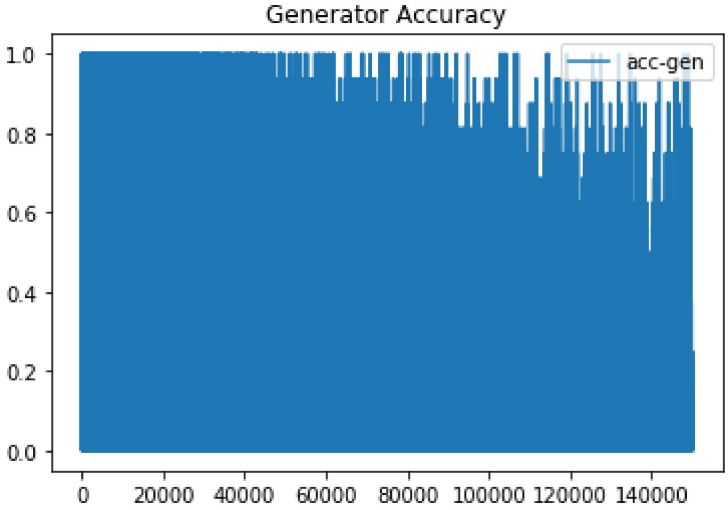
Model Generated Color Image:

Saved model to disk
Iteration number: 150000
Elapsed Time: 4:59:46.723885
Generator Loss: 8.985292
Discriminator Loss: 1.0369932651519775
Generator Accuracy: 0.0
Discriminator Accuracy: 0.5







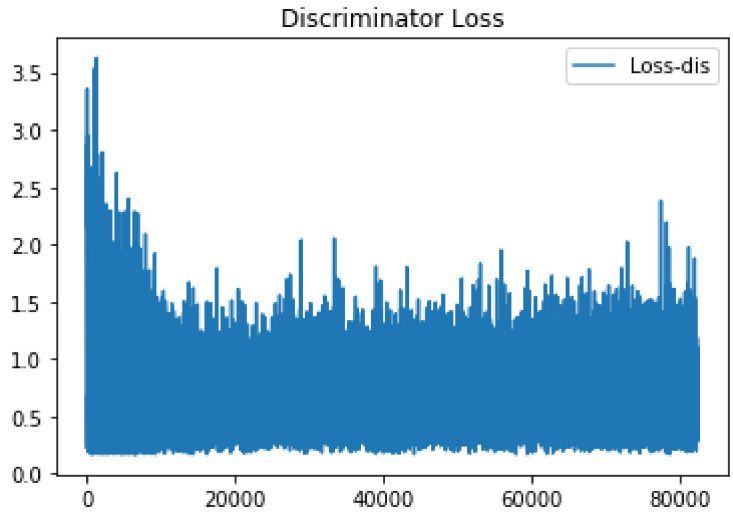


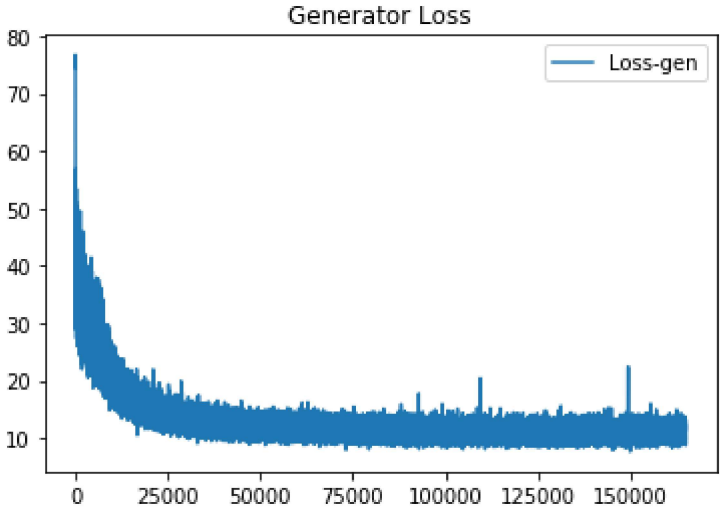
Model Generated Color Image:

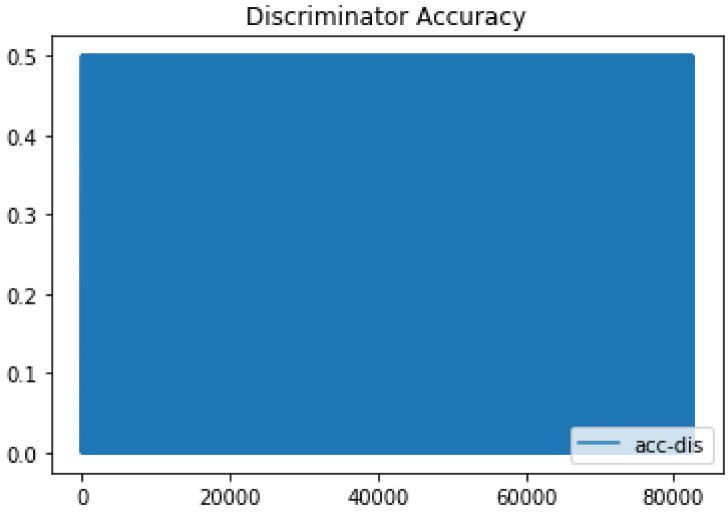
Model Generated Color Image:

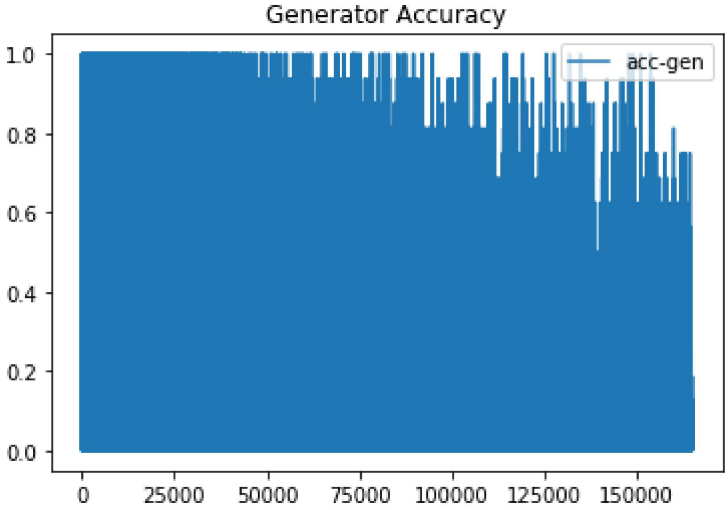
Model Generated Color Image:

Saved model to disk
Iteration number: 165000
Elapsed Time: 5:28:33.040858
Generator Loss: 9.171709
Discriminator Loss: 0.6093946695327759
Generator Accuracy: 0.125
Discriminator Accuracy: 0.5







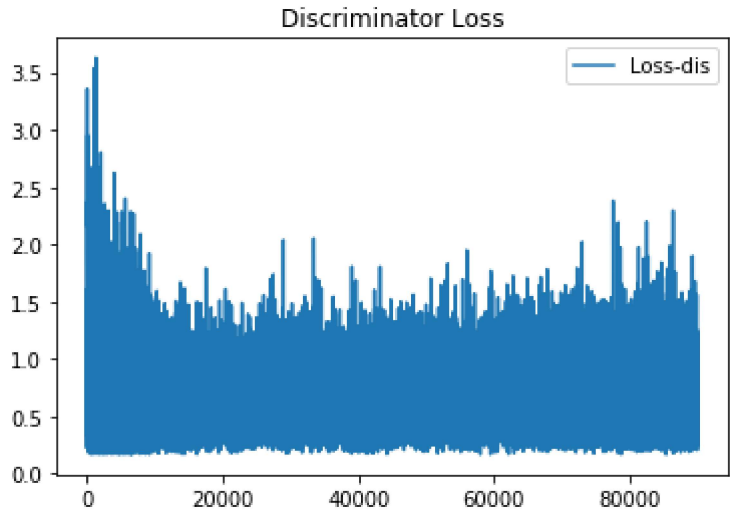


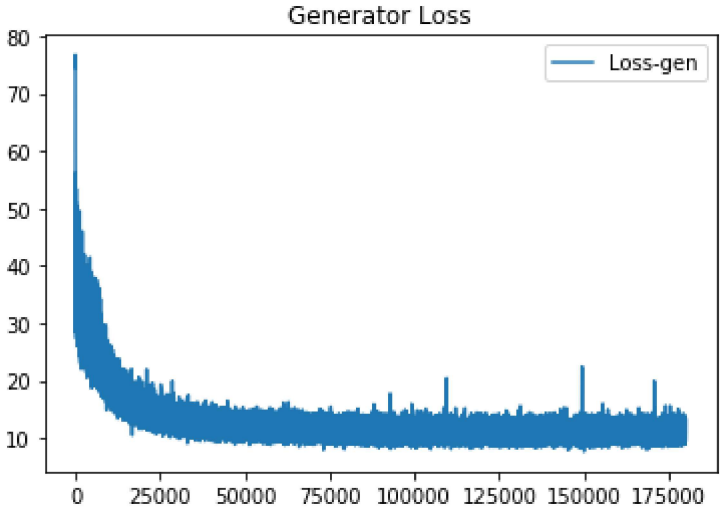
Model Generated Color Image:

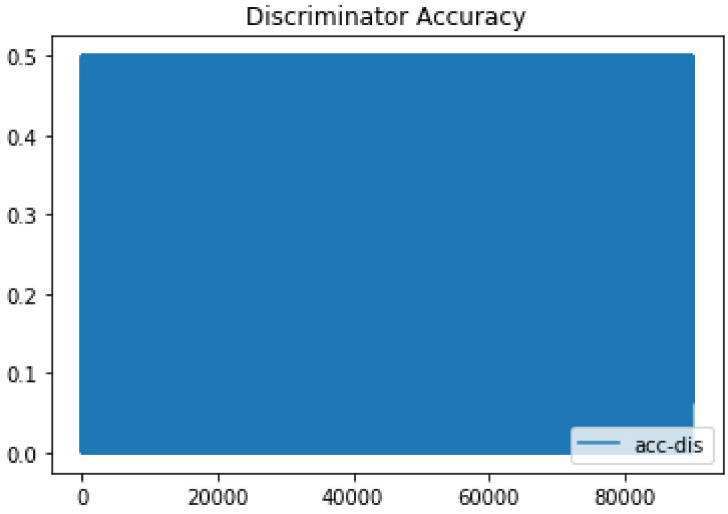
Model Generated Color Image:

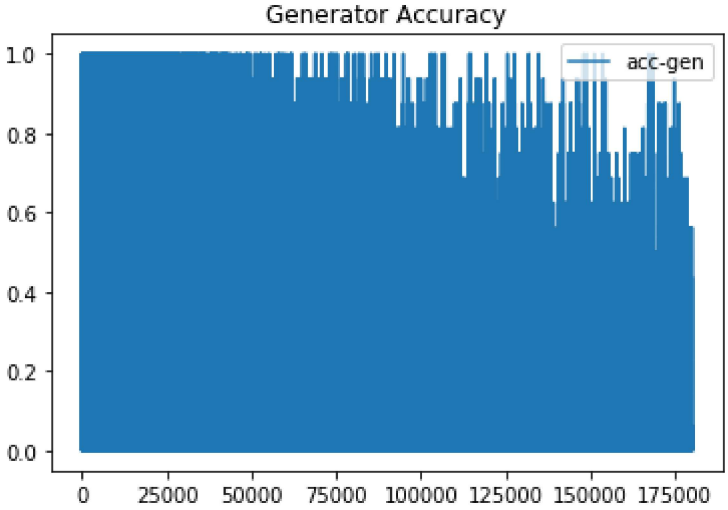
Model Generated Color Image:

```
Saved model to disk
Iteration number: 180000
Elapsed Time: 5:57:15.119194
Generator Loss: 11.167053
Discriminator Loss: 0.2558596134185791
Generator Accuracy: 0.0
Discriminator Accuracy: 0.5
```







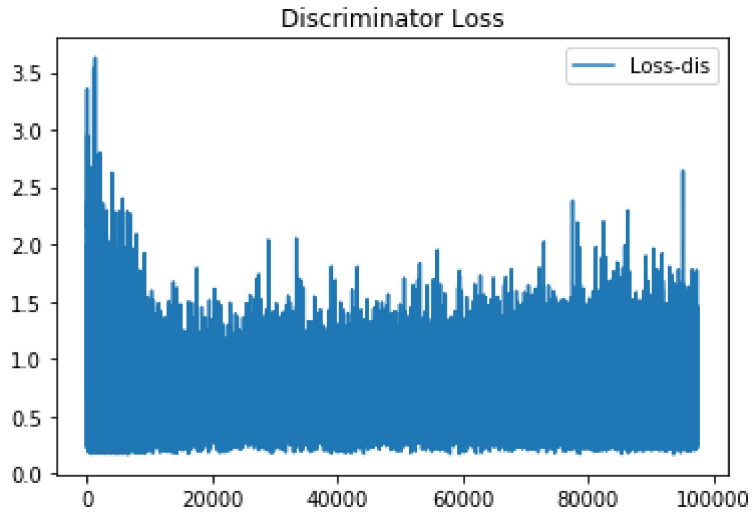


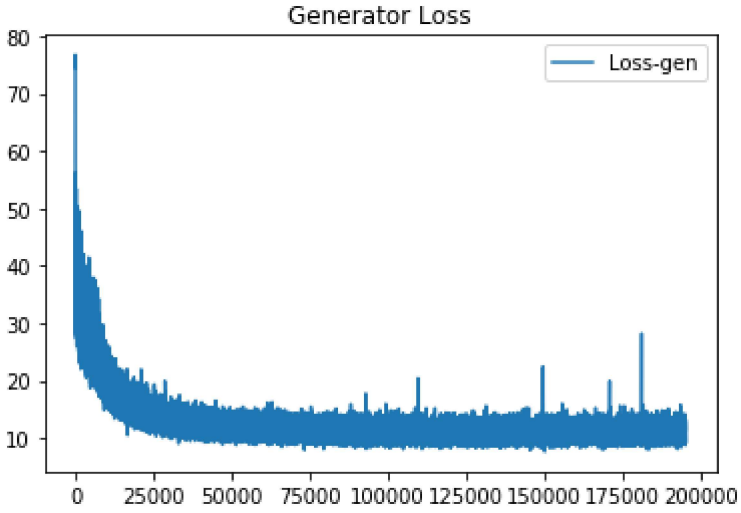
Model Generated Color Image:

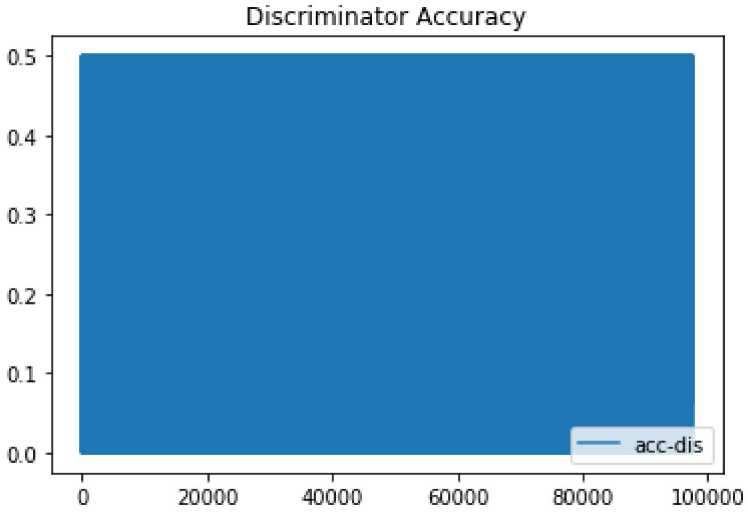
Model Generated Color Image:

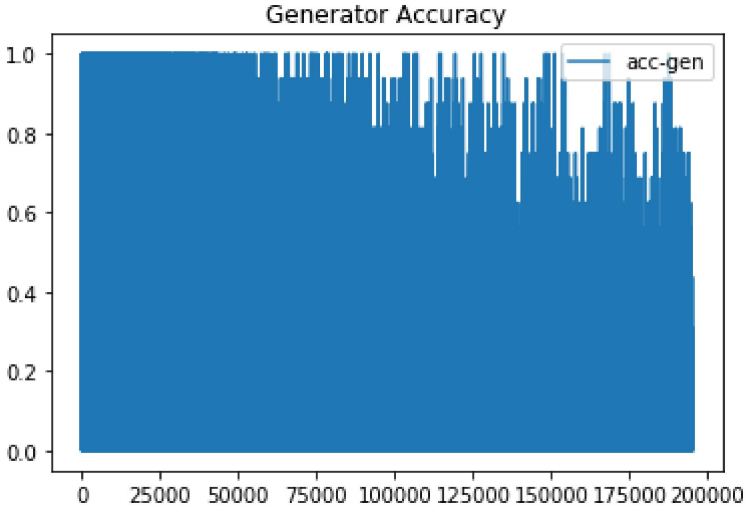
Model Generated Color Image:

```
Saved model to disk
Iteration number: 194999
Elapsed Time: 6:26:05.549282
Generator Loss: 11.529121
Discriminator Loss: 0.4311524033546448
Generator Accuracy: 0.0
Discriminator Accuracy: 0.5
```









Model Generated Color Image:

Model Generated Color Image:

Model Generated Color Image:

Saved model to disk

In [0]:
```
generator= build_generator()
```

In [0]:
```
generator.load_weights('model_gen_150000.h5')
```

In [0]:
```
#Generating Color Image on some train data.
tst = img_arr[0:10]
res = generator.predict(tst)
```

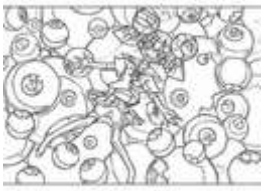
In [0]:
```
#Print color image to compare results.
for i in range(10):
  print("Sketch Image:")
  cv2_imshow(((img_arr[i] + 1)/2.0)*255.0)
  print("Original Color Image:")
  cv2_imshow(((col_arr[i] + 1)/2.0)*255.0)
```

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:



Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

In [0]:
```
#Load test data - Sketch.
img_tst = []

for j in  range(1):
  i = j + 6
  folder=i.__str__().zfill(4)
  loc = 'sketch-pair/danbooru-sketch-pair-128x/color/sketch/' + folder + '/'
  print('processing folder',folder)
  for filename in os.listdir(loc):
    img_tst.append(cv2.imread(loc+filename,1))
```

```
processing folder 0006
```

In [0]:
```
#Load test data - Color.
col_tst = []

for j in  range(1):
  i = j + 6
  folder=i.__str__().zfill(4)
  loc = 'sketch-pair/danbooru-sketch-pair-128x/color/src/' + folder + '/'
  print('processing folder',folder)
  for filename in os.listdir(loc):
    col_tst.append(cv2.imread(loc+filename,1))
```

```
processing folder 0006
```

In [0]:
```
#Converting test data as numpy array.
col_tst = np.array(col_tst, dtype='uint8')
img_tst = np.array(img_tst, dtype='uint8')
```

In [0]:
```
#Normalize the pixel value to be between -1 & +1.
col_tst = (col_tst - 127.5) / 127.5
img_tst = (img_tst - 127.5) / 127.5
```

In [0]:
```
#Generating Color Image on some test data.
tst = img_tst[0:10]
res = generator.predict(tst)
```

In [0]:
```
#Print color image on test data to compare results.
for i in range(10):
  print("Sketch Image:")
  cv2_imshow(((img_tst[i] + 1)/2.0)*255.0)
  print("Original Color Image:")
  cv2_imshow(((col_tst[i] + 1)/2.0)*255.0)
  print("Model Generated Color Image:")
  cv2_imshow(((res[i] + 1)/2.0)*255.0)
```

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

Sketch Image:

Original Color Image:

Model Generated Color Image:

6. Steps Followed

- Downloaded data from Danbooru Sketch Pair 128x on kaggle https://www.kaggle.com/wuhecong/danbooru-sketch-pair-128xsv (https://www.kaggle.com/wuhecong/danbooru-sketch-pair-128xsv)"
- Unzipped the folder.
- Imported all the required libraries. We have uses - Keras libraries, numpy, os, random, cv2, matplotlib.pyplot
- Loaded Train data - Sketches and Corresponding color images.
- Defined an U-net model.
- Trained model on train dataset and have saved weights.
- Plotted train and test results.
- Loaded test data and generated image using model on train and test data.
- Then defined a GAN model. Defined below functions -
  - Generator
  - Discriminator
  - Pixel Level loss.
  - GAN
  - Generate Sample for batch processing
  - Plot history
  - Summarize performance.
  - Training
- Trained the GAN model and plotted results.
- Saved model weights.
- Generated images on train and test data using model generated above.

7. Results

- GAN Model generated better images than a simple U-net model.
- Model couldn't perform well on unseen images. Possible reason could be, we used less train data because of system constraints.

8. References

- https://machinelearningmastery.com/a-gentle-introduction-to-pix2pix-generative-adversarial-network/ (https://machinelearningmastery.com/a-gentle-introduction-to-pix2pix-generative-adversarial-network/)
- https://machinelearningmastery.com/upsampling-and-transpose-convolution-layers-for-generative-adversarial-networks/ (https://machinelearningmastery.com/upsampling-and-transpose-convolution-layers-for-generative-adversarial-networks/)
- https://arxiv.org/pdf/1611.07004v1.pdf (https://arxiv.org/pdf/1611.07004v1.pdf)
- https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/ (https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/)

- https://www.kaggle.com/wuhecong/danbooru-sketch-pair-128x (https://www.kaggle.com/wuhecong/danbooru-sketch-pair-128x)
- https://github.com/kvfrans/deepcolor/blob/master/main.py (https://github.com/kvfrans/deepcolor/blob/master/main.py)
- https://towardsdatascience.com/sketch-to-color-anime-translation-using-generative-adversarial-networks-gans-8f4f69594aeb (https://towardsdatascience.com/sketch-to-color-anime-translation-using-generative-adversarial-networks-gans-8f4f69594aeb)
- https://arxiv.org/pdf/1808.03240.pdf (https://arxiv.org/pdf/1808.03240.pdf)
- http://kvfrans.com/coloring-and-shading-line-art-automatically-through-conditional-gans/ (http://kvfrans.com/coloring-and-shading-line-art-automatically-through-conditional-gans/)
- https://arxiv.org/pdf/1705.01908v2.pdf (https://arxiv.org/pdf/1705.01908v2.pdf)