OracleGAN

This study explores how to effectively learn time series data using GAN-based neural networks.

The cost function is combination of Pix2Pix's cost function and future predictive loss function using the discount coefficient used in DQN. Also, it is easy to apply future predictive loss function using a one-to-many dataset structure.

Key Featues of OracleGAN

- Time Step Image Dataset
- Cost Function of Generator
- Cost Function of Discriminator

Goal

- Predict future weather images using current weather images.

Install additional libraries

IQA_pytorch is a library which is used to calculate SSIM Score

In [54]:
```
pwd
```

Out[54]:
```
'/data/scratch/pbs.2398.srv-svkmmastermum.x8z'
```

In [55]:
```
import os
os.listdir()
```

Out[55]:
```
['Satellite_imaging_GAN.ipynb',
 'real_series.zip',
 'Satellite_imaging_GAN.o2398',
 'Satellite_imaging_GAN.e2398',
 'ai_series.zip',
 'Discriminator.pth',
 'access.log',
 'input',
 'data',
 '.ipynb_checkpoints',
 'real_series.mp4',
 'dloss.txt',
 'Input_SERIES.png',
 'job.env',
 'oraclegan_series.mp4',
 'runtime',
 'ai_noseries.zip',
 'archive.zip',
 'gloss.txt',
 'real_noseries.zip',
 'Generator.pth']
```

In [56]:
```
import shutil
filename="/data/scratch/pbs.2398.srv-svkmmastermum.x8z/archive.zip"
extract_dir="/data/scratch/pbs.2398.srv-svkmmastermum.x8z/data/"
shutil.unpack_archive(filename, extract_dir)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[56], line 4
      2 filename="/data/scratch/pbs.2398.srv-svkmmastermum.x8z/archive.zip"
      3 extract_dir="/data/scratch/pbs.2398.srv-svkmmastermum.x8z/data/"
----> 4 shutil.unpack_archive(filename, extract_dir)

File /usr/lib/python3.10/shutil.py:1306, in unpack_archive(filename, extract_dir, format)
   1304 func = _UNPACK_FORMATS[format][1]
   1305 kwargs = dict(_UNPACK_FORMATS[format][2])
-> 1306 func(filename, extract_dir, **kwargs)

File /usr/lib/python3.10/shutil.py:1218, in _unpack_zipfile(filename, extract_dir)
   1215         _ensure_directory(targetpath)
   1216         if not name.endswith('/'):
   1217             # file
-> 1218             with zip.open(name, 'r') as source, \
   1219                     open(targetpath, 'wb') as target:
   1220                 copyfileobj(source, target)
   1221 finally:

KeyboardInterrupt:
```

In [ ]:
```
pip install IQA_pytorch #For SSIM Score
```

Import Libraries

In [57]:
```
import torch
import torch.nn as nn
from torch.autograd import Variable
import torchvision
from torch.optim import *
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
from torchvision import transforms
from IQA_pytorch import DISTS, utils

import numpy as np
from PIL import Image
import cv2
import numpy as np
import albumentations
import albumentations.pytorch
from matplotlib import pyplot as plt
import matplotlib.animation as animation
from matplotlib import font_manager, rc
from IPython import display
import random
import glob
import os
from os import listdir
from os.path import isfile, join
import warnings
import sys
from tqdm import tqdm
import pickle
import gc
import random
import urllib.request

warnings.filterwarnings("ignore")
```

In [ ]:
```
pip install tqdm
```

In [ ]:
```
pip install matplotlib
```

In [ ]:
```
pip install albumentations
```

In [ ]:
```
pip install opencv-python-headless
```

In [ ]:
```
pip install torch
```

In [ ]:
```
pip install torchvision
```

In [ ]:
```
gc.collect()
torch.cuda.empty_cache()
```

In [ ]:
```
%matplotlib inline

plt.rcParams['axes.unicode_minus'] = False
fontpath = "../input/koreanfont/NanumBrush.ttf"
fontprop = font_manager.FontProperties(fname=fontpath)

plt.rcParams["animation.html"] = "jshtml"
plt.rcParams['figure.dpi'] = 150
plt.ioff()
```

Define Hyperparameters

Name of Hyperparameter	Explanation
USE_CUDA	whether to use GPU
DEBUG	whether to print specific logs
RANDOM_SEED	random seed of pytorch, random, numpy
start_epoch	this is used to continuing train from checkpoint
all_epochs	Epochs
batch_size	Batch Size
lrG	the learning rate of Generator
lrD	the learning rate of Discriminator
beta1, beta2	the beta1 and beta2 of Generator and Discriminator
L1Lambda	lambda of pix2pix objective function
GAMMA	factor similar to discount factor of DQN. (0<γ<1) (check cost function of OracleGAN Generator)
TIME_STEP	the number of future images which is used to calculate loss (check cost function of OracleGAN Generator)

In [87]:
```
# Device
USE_CUDA = torch.cuda.is_available()

print("Device : {0}".format("GPU" if USE_CUDA else "CPU"))
device = torch.device("cuda" if USE_CUDA else "cpu")
cpu_device = torch.device("cpu")

DEBUG = False

RANDOM_SEED = 2004

# Train
start_epoch = 0
all_epochs = 10
batch_size = 14

lrG = 0.0002
lrD = 0.0002
beta1 = 0.5
beta2 = 0.999

L1lambda = 100
GAMMA = 0.59

TIME_STEP = 4
TEST_TIME_STEP = 6

patch = (1,256//2**4,256//2**4)

# Path
DATASET1_PATH = '/data/scratch/pbs.2398.srv-svkmmastermum.x8z/data/'

# Checkpoint
USE_CHECKPOINT = True

OLD_PATH = '/data/scratch/pbs.2398.srv-svkmmastermum.x8z/'
OLD_GENERATOR_MODEL = os.path.join(OLD_PATH, 'Generator.pth')
OLD_DISCRIMINATOR_MODEL = os.path.join(OLD_PATH, 'Discriminator.pth')
OLD_G_LOSS = os.path.join(OLD_PATH, 'gloss.txt')
OLD_D_LOSS = os.path.join(OLD_PATH, 'dloss.txt')
```

```
Device : GPU
```

In [59]:
```
torch.manual_seed(RANDOM_SEED)
torch.cuda.manual_seed(RANDOM_SEED)
torch.cuda.manual_seed_all(RANDOM_SEED)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
np.random.seed(RANDOM_SEED)
random.seed(RANDOM_SEED)

print('Random Seed : {0}'.format(RANDOM_SEED))
```

```
Random Seed : 2004
```

In [60]:
```
def log(text):
    global DEBUG
    if DEBUG:
        print(text)
```

Visualize Data

Name of Function	Explanation
torch_tensor_to_plt	Convert torch image to matplotlib image
plt_image_animation	show a video by update_function

In [61]:
```
def torch_tensor_to_plt(img):
    img = img.detach().numpy()[0]
    img = np.transpose(img, (1, 2, 0))
    return img
```

In [62]:
```
def show_video_in_jupyter_nb(width, height, video_url):
    from IPython.display import HTML
    return HTML("""<video width="{}" height="{}" controls>
    <source src={} type="video/mp4">
    </video>""".format(width, height, video_url))
```

In [63]:
```
def plt_image_animation(frames, update_func):
    fig, ax = plt.subplots(figsize=(4,4))
    plt.axis('off')
    anim = animation.FuncAnimation(fig, update_func, frames=frames)
    video = anim.to_html5_video()
    html = display.HTML(video)
    display.display(html)
    plt.close()
```

In [64]:
```
plt_image_animation(15, lambda t : plt.imshow(np.load(join(DATASET1_PATH, '2
```

```
---------------------------------------------------------------------------
-
RuntimeError                              Traceback (most recent call las
t)
Cell In[64], line 1
----> 1 plt_image_animation(15, lambda t : plt.imshow(np.load(join(DATASET
1_PATH, '2017M01', '{0}.npy'.format(t))), cmap='gray'))

Cell In[63], line 5, in plt_image_animation(frames, update_func)
      3 plt.axis('off')
      4 anim = animation.FuncAnimation(fig, update_func, frames=frames)
----> 5 video = anim.to_html5_video()
      6 html = display.HTML(video)
      7 display.display(html)

File ~/.local/lib/python3.10/site-packages/matplotlib/animation.py:1285, i
n Animation.to_html5_video(self, embed_limit)
   1282 path = Path(tmpdir, "temp.m4v")
   1283 # We create a writer manually so that we can get the
   1284 # appropriate size for the tag
-> 1285 Writer = writers[mpl.rcParams['animation.writer']]
   1286 writer = Writer(codec='h264',
   1287                 bitrate=mpl.rcParams['animation.bitrate'],
   1288                 fps=1000. / self._interval)
   1289 self.save(str(path), writer=writer)

File ~/.local/lib/python3.10/site-packages/matplotlib/animation.py:148, in
MovieWriterRegistry.__getitem__(self, name)
    146 if self.is_available(name):
    147     return self._registered[name]
--> 148 raise RuntimeError(f"Requested MovieWriter ({name}) not availabl
e")

RuntimeError: Requested MovieWriter (ffmpeg) not available
```

Preprocess Dataset

In [65]:
```
transformer = transforms.Compose([transforms.ToTensor(),
                                  torchvision.transforms.Resize(128),
                                  transforms.Normalize((0.5), (0.5)), #Gray
                                 ])
```

Time Step Image Dataset

OracleGAN calculates loss between predicted image and real image not only after 15 minutes but also **after 15×TimeStep minutes**.

So, dataset need to have **multiple output** images per **one input** image.

In [66]:
```
nowpath = ""

class TimeStepImageDataset(Dataset):
    def __init__(self, date, time_step, transform=None):
        self.date = date
        self.time_step = time_step
        self.transformer = transform
        self.file = []

        file_list = glob.glob(join(self.date, '*'))
        self.file = [file for file in file_list if (file.endswith(".npy") ar

    def __len__(self):
        return len(self.file)-self.time_step

    def transform(self, image):
        if self.transformer:
            return self.transformer(image)
        else :
            return image

    def __getitem__(self, idx):
        global nowpath
        log(join(self.date, str(idx)+'.npy'))
        X = self.transform(np.load(join(self.date, str(idx)+'.npy')))
        nowpath = join(self.date, str(idx)+'.npy')
        Y_list = []
        for i in range(1, self.time_step+1):
            Y_list.append(self.transform(np.load(join(self.date, str(idx+i)+
        Y = torch.cat(Y_list)
        return X, Y
```

In [67]:
```
DATASET1_DIRS = glob.glob(join(DATASET1_PATH, '*'))

random.shuffle(DATASET1_DIRS)

traindatasetlist = []
for ind, name in enumerate(DATASET1_DIRS[:20]):
    traindatasetlist.append(TimeStepImageDataset(name, TIME_STEP, transform=
train_dataset = torch.utils.data.ConcatDataset(traindatasetlist)

testdatasetlist = []
for ind, name in enumerate(DATASET1_DIRS[20:]):
    testdatasetlist.append(TimeStepImageDataset(name, TEST_TIME_STEP, transf
test_dataset = torch.utils.data.ConcatDataset(testdatasetlist)
```

In [68]:
```
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=
test_dataloader = DataLoader(test_dataset, batch_size=batch_size, shuffle=Tr

test_dataloader_bs1_shuffle = DataLoader(test_dataset, batch_size=1, shuffle
test_dataloader_bs1_noshuffle = DataLoader(test_dataset, batch_size=1, shuff
```

In [69]:
```
def ShowDatasetImage(x, y):
    grid = torchvision.utils.make_grid(y)

    fig = plt.figure(figsize=(2, 2))
    plt.imshow(torch_tensor_to_plt(x.unsqueeze(0)), cmap='gray')
    plt.axis('off')
    plt.title('Input (Now)', fontproperties=fontprop)
    plt.show()

    fig = plt.figure(figsize=(8, 2.5))
    plt.title('Real Weather Image', fontproperties=fontprop)
    plt.axis('off')
    for i in range(1, TIME_STEP+1):
        ax = fig.add_subplot(1, TIME_STEP, i)
        ax.axis('off')
        ax.imshow(torch_tensor_to_plt(y[i-1].unsqueeze(0)), cmap='gray')
        ax.set_title('after {0} minutes'.format(15*i), fontproperties=fontpr
    plt.show()

    del x, y
```

In [70]:
```
for ind, (x, y) in enumerate(train_dataset):
    if ind != 0:
        continue
    ShowDatasetImage(x, y)
    break
```

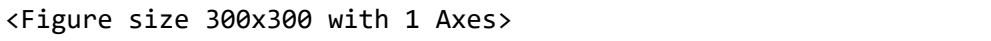
```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
File /usr/local/lib/python3.10/dist-packages/IPython/core/formatters.py:340, in BaseFormatter.__call__(self, obj)
    338     pass
    339 else:
--> 340     return printer(obj)
    341 # Finally look for special method names
    342 method = get_real_method(obj, self.print_method)

File /usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:152, in print_figure(fig, fmt, bbox_inches, base64, **kwargs)
    149     from matplotlib.backend_bases import FigureCanvasBase
    150     FigureCanvasBase(fig)
--> 152 fig.canvas.print_figure(bytes_io, **kw)
    153 data = bytes_io.getvalue()
    154 if fmt == 'svg':

File ~/.local/lib/python3.10/site-packages/matplotlib/backend_bases.py:2164, in FigureCanvasBase.print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format, bbox_inches, pad_inches, bbox_extra_artists, backend, **kwargs)
   2161     # we do this instead of `self.figure.draw_without_rendering`
   2162     # so that we can inject the orientation
   2163     with getattr(renderer, "_draw_disabled", nullcontext)():
-> 2164         self.figure.draw(renderer)
   2165 if bbox_inches:
   2166     if bbox_inches == "tight":

File ~/.local/lib/python3.10/site-packages/matplotlib/artist.py:95, in _finalize_rasterization.<locals>.draw_wrapper(artist, renderer, *args, **kwargs)
     93 @wraps(draw)
     94 def draw_wrapper(artist, renderer, *args, **kwargs):
---> 95     result = draw(artist, renderer, *args, **kwargs)
     96     if renderer._rasterizing:
     97         renderer.stop_rasterizing()

File ~/.local/lib/python3.10/site-packages/matplotlib/artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist, renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~/.local/lib/python3.10/site-packages/matplotlib/figure.py:3154, in Figure.draw(self, renderer)
   3151         # ValueError can occur when resizing a window.
   3153 self.patch.draw(renderer)
-> 3154 mimage._draw_list_compositing_images(
   3155     renderer, self, artists, self.suppressComposite)
   3157 for sfig in self.subfigs:
   3158     sfig.draw(renderer)

File ~/.local/lib/python3.10/site-packages/matplotlib/image.py:132, in _draw_list_compositing_images(renderer, parent, artists, suppress_composite)
    130 if not_composite or not has_images:
    131     for a in artists:
```

```
--> 132         a.draw(renderer)
    133 else:
    134     # Composite any adjacent images together
    135     image_group = []

File ~/.local/lib/python3.10/site-packages/matplotlib/artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist, renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~/.local/lib/python3.10/site-packages/matplotlib/axes/_base.py:3034, in _AxesBase.draw(self, renderer)
   3031     for spine in self.spines.values():
   3032         artists.remove(spine)
-> 3034 self._update_title_position(renderer)
   3036 if not self.axison:
   3037     for _axis in self._axis_map.values():

File ~/.local/lib/python3.10/site-packages/matplotlib/axes/_base.py:2988, in _AxesBase._update_title_position(self, renderer)
   2986     _log.debug('top of Axes not in the figure, so title not moved')
   2987     return
-> 2988 if title.get_window_extent(renderer).ymin < top:
   2989     _, y = self.transAxes.inverted().transform((0, top))
   2990     title.set_position((x, y))

File ~/.local/lib/python3.10/site-packages/matplotlib/text.py:956, in Text.get_window_extent(self, renderer, dpi)
    951     raise RuntimeError(
    952         "Cannot get window extent of text w/o renderer. You likely "
    953         "want to call 'figure.draw_without_rendering()' first.")
    955 with cbook._setattr_cm(self.figure, dpi=dpi):
--> 956     bbox, info, descent = self._get_layout(self._renderer)
    957     x, y = self.get_unitless_position()
    958     x, y = self.get_transform().transform((x, y))

File ~/.local/lib/python3.10/site-packages/matplotlib/text.py:373, in Text._get_layout(self, renderer)
    370 ys = []
    372 # Full vertical extent of font, including ascenders and descenders:
--> 373 _, lp_h, lp_d = _get_text_metrics_with_cache(
    374     renderer, "lp", self._fontproperties,
    375     ismath="TeX" if self.get_usetex() else False, dpi=self.figure.dpi)
    376 min_dy = (lp_h - lp_d) * self._linespacing
    378 for i, line in enumerate(lines):

File ~/.local/lib/python3.10/site-packages/matplotlib/text.py:69, in _get_text_metrics_with_cache(renderer, text, fontprop, ismath, dpi)
     66 """Call ``renderer.get_text_width_height_descent``, caching the results."""
     67 # Cached based on a copy of fontprop so that later in-place mutations of
     68 # the passed-in argument do not mess up the cache.
---> 69 return _get_text_metrics_with_cache_impl(
```

```
     70     weakref.ref(renderer), text, fontprop.copy(), ismath, dpi)

File ~/.local/lib/python3.10/site-packages/matplotlib/text.py:77, in _get_
text_metrics_with_cache_impl(renderer_ref, text, fontprop, ismath, dpi)
     73 @functools.lru_cache(4096)
     74 def _get_text_metrics_with_cache_impl(
     75         renderer_ref, text, fontprop, ismath, dpi):
     76     # dpi is unused, but participates in cache invalidation (via t
he renderer).
---> 77     return renderer_ref().get_text_width_height_descent(text, font
prop, ismath)

File ~/.local/lib/python3.10/site-packages/matplotlib/backends/backend_ag
g.py:220, in RendererAgg.get_text_width_height_descent(self, s, prop, isma
th)
    216     ox, oy, width, height, descent, font_image = \
    217         self.mathtext_parser.parse(s, self.dpi, prop)
    218     return width, height, descent
--> 220 font = self._prepare_font(prop)
    221 font.set_text(s, 0.0, flags=get_hinting_flag())
    222 w, h = font.get_width_height()  # width and height of unrotated st
ring

File ~/.local/lib/python3.10/site-packages/matplotlib/backends/backend_ag
g.py:254, in RendererAgg._prepare_font(self, font_prop)
    250 def _prepare_font(self, font_prop):
    251     """
    252     Get the `.FT2Font` for *font_prop*, clear its buffer, and set
its size.
    253     """
--> 254     font = get_font(_fontManager._find_fonts_by_props(font_prop))
    255     font.clear()
    256     size = font_prop.get_size_in_points()

File ~/.local/lib/python3.10/site-packages/matplotlib/font_manager.py:155
4, in get_font(font_filepaths, hinting_factor)
   1551 if hinting_factor is None:
   1552     hinting_factor = mpl.rcParams['text.hinting_factor']
-> 1554 return _get_font(
   1555     # must be a tuple to be cached
   1556     paths,
   1557     hinting_factor,
   1558     _kerning_factor=mpl.rcParams['text.kerning_factor'],
   1559     # also key on the thread ID to prevent segfaults with multi-th
reading
   1560     thread_id=threading.get_ident()
   1561 )

File ~/.local/lib/python3.10/site-packages/matplotlib/font_manager.py:149
6, in _get_font(font_filepaths, hinting_factor, _kerning_factor, thread_i
d)
   1493 @lru_cache(64)
   1494 def _get_font(font_filepaths, hinting_factor, *, _kerning_factor,
thread_id):
   1495     first_fontpath, *rest = font_filepaths
-> 1496     return ft2font.FT2Font(
   1497         first_fontpath, hinting_factor,
   1498         _fallback_list=[
   1499             ft2font.FT2Font(
   1500                 fpath, hinting_factor,
   1501                 _kerning_factor=_kerning_factor
```

```
            )
            for fpath in rest
        ],
        _kerning_factor=_kerning_factor
    )
```

```
FileNotFoundError: [Errno 2] No such file or directory: '/data/scratch/inp
ut/koreanfont/NanumBrush.ttf'
```



```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
File /usr/local/lib/python3.10/dist-packages/IPython/core/formatters.py:340, in BaseFormatter.__call__(self, obj)
    338     pass
    339 else:
--> 340     return printer(obj)
    341 # Finally look for special method names
    342 method = get_real_method(obj, self.print_method)

File /usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:152, in print_figure(fig, fmt, bbox_inches, base64, **kwargs)
    149     from matplotlib.backend_bases import FigureCanvasBase
    150     FigureCanvasBase(fig)
--> 152 fig.canvas.print_figure(bytes_io, **kw)
    153 data = bytes_io.getvalue()
    154 if fmt == 'svg':

File ~/.local/lib/python3.10/site-packages/matplotlib/backend_bases.py:2164, in FigureCanvasBase.print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format, bbox_inches, pad_inches, bbox_extra_artists, backend, **kwargs)
   2161     # we do this instead of `self.figure.draw_without_rendering`
   2162     # so that we can inject the orientation
   2163     with getattr(renderer, "_draw_disabled", nullcontext)():
-> 2164         self.figure.draw(renderer)
   2165 if bbox_inches:
   2166     if bbox_inches == "tight":

File ~/.local/lib/python3.10/site-packages/matplotlib/artist.py:95, in _finalize_rasterization.<locals>.draw_wrapper(artist, renderer, *args, **kwargs)
     93 @wraps(draw)
     94 def draw_wrapper(artist, renderer, *args, **kwargs):
---> 95     result = draw(artist, renderer, *args, **kwargs)
     96     if renderer._rasterizing:
     97         renderer.stop_rasterizing()

File ~/.local/lib/python3.10/site-packages/matplotlib/artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist, renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~/.local/lib/python3.10/site-packages/matplotlib/figure.py:3154, in Figure.draw(self, renderer)
   3151         # ValueError can occur when resizing a window.
   3153 self.patch.draw(renderer)
-> 3154 mimage._draw_list_compositing_images(
   3155     renderer, self, artists, self.suppressComposite)
   3157 for sfig in self.subfigs:
   3158     sfig.draw(renderer)

File ~/.local/lib/python3.10/site-packages/matplotlib/image.py:132, in _draw_list_compositing_images(renderer, parent, artists, suppress_composite)
    130 if not_composite or not has_images:
    131     for a in artists:
```

```
--> 132         a.draw(renderer)
    133 else:
    134     # Composite any adjacent images together
    135     image_group = []

File ~/.local/lib/python3.10/site-packages/matplotlib/artist.py:72, in allow_rasterization.<locals>.draw_wrapper(artist, renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File ~/.local/lib/python3.10/site-packages/matplotlib/axes/_base.py:3034, in _AxesBase.draw(self, renderer)
   3031     for spine in self.spines.values():
   3032         artists.remove(spine)
-> 3034 self._update_title_position(renderer)
   3036 if not self.axison:
   3037     for _axis in self._axis_map.values():

File ~/.local/lib/python3.10/site-packages/matplotlib/axes/_base.py:2988, in _AxesBase._update_title_position(self, renderer)
   2986     _log.debug('top of Axes not in the figure, so title not moved')
   2987     return
-> 2988 if title.get_window_extent(renderer).ymin < top:
   2989     _, y = self.transAxes.inverted().transform((0, top))
   2990     title.set_position((x, y))

File ~/.local/lib/python3.10/site-packages/matplotlib/text.py:956, in Text.get_window_extent(self, renderer, dpi)
    951     raise RuntimeError(
    952         "Cannot get window extent of text w/o renderer. You likely "
    953         "want to call 'figure.draw_without_rendering()' first.")
    955 with cbook._setattr_cm(self.figure, dpi=dpi):
--> 956     bbox, info, descent = self._get_layout(self._renderer)
    957     x, y = self.get_unitless_position()
    958     x, y = self.get_transform().transform((x, y))

File ~/.local/lib/python3.10/site-packages/matplotlib/text.py:373, in Text._get_layout(self, renderer)
    370 ys = []
    372 # Full vertical extent of font, including ascenders and descenders:
--> 373 _, lp_h, lp_d = _get_text_metrics_with_cache(
    374     renderer, "lp", self._fontproperties,
    375     ismath="TeX" if self.get_usetex() else False, dpi=self.figure.dpi)
    376 min_dy = (lp_h - lp_d) * self._linespacing
    378 for i, line in enumerate(lines):

File ~/.local/lib/python3.10/site-packages/matplotlib/text.py:69, in _get_text_metrics_with_cache(renderer, text, fontprop, ismath, dpi)
     66 """Call ``renderer.get_text_width_height_descent``, caching the results."""
     67 # Cached based on a copy of fontprop so that later in-place mutations of
     68 # the passed-in argument do not mess up the cache.
---> 69 return _get_text_metrics_with_cache_impl(
```

```
     70     weakref.ref(renderer), text, fontprop.copy(), ismath, dpi)

File ~/.local/lib/python3.10/site-packages/matplotlib/text.py:77, in _get_
text_metrics_with_cache_impl(renderer_ref, text, fontprop, ismath, dpi)
     73 @functools.lru_cache(4096)
     74 def _get_text_metrics_with_cache_impl(
     75         renderer_ref, text, fontprop, ismath, dpi):
     76     # dpi is unused, but participates in cache invalidation (via t
he renderer).
---> 77     return renderer_ref().get_text_width_height_descent(text, font
prop, ismath)

File ~/.local/lib/python3.10/site-packages/matplotlib/backends/backend_ag
g.py:220, in RendererAgg.get_text_width_height_descent(self, s, prop, isma
th)
    216     ox, oy, width, height, descent, font_image = \
    217         self.mathtext_parser.parse(s, self.dpi, prop)
    218     return width, height, descent
--> 220 font = self._prepare_font(prop)
    221 font.set_text(s, 0.0, flags=get_hinting_flag())
    222 w, h = font.get_width_height()  # width and height of unrotated st
ring

File ~/.local/lib/python3.10/site-packages/matplotlib/backends/backend_ag
g.py:254, in RendererAgg._prepare_font(self, font_prop)
    250 def _prepare_font(self, font_prop):
    251     """
    252     Get the `.FT2Font` for *font_prop*, clear its buffer, and set
its size.
    253     """
--> 254     font = get_font(_fontManager._find_fonts_by_props(font_prop))
    255     font.clear()
    256     size = font_prop.get_size_in_points()

File ~/.local/lib/python3.10/site-packages/matplotlib/font_manager.py:155
4, in get_font(font_filepaths, hinting_factor)
   1551 if hinting_factor is None:
   1552     hinting_factor = mpl.rcParams['text.hinting_factor']
-> 1554 return _get_font(
   1555     # must be a tuple to be cached
   1556     paths,
   1557     hinting_factor,
   1558     _kerning_factor=mpl.rcParams['text.kerning_factor'],
   1559     # also key on the thread ID to prevent segfaults with multi-th
reading
   1560     thread_id=threading.get_ident()
   1561 )

File ~/.local/lib/python3.10/site-packages/matplotlib/font_manager.py:149
6, in _get_font(font_filepaths, hinting_factor, _kerning_factor, thread_i
d)
   1493 @lru_cache(64)
   1494 def _get_font(font_filepaths, hinting_factor, *, _kerning_factor,
thread_id):
   1495     first_fontpath, *rest = font_filepaths
-> 1496     return ft2font.FT2Font(
   1497         first_fontpath, hinting_factor,
   1498         _fallback_list=[
   1499             ft2font.FT2Font(
   1500                 fpath, hinting_factor,
   1501                 _kerning_factor=_kerning_factor
```

```
            )
            for fpath in rest
        ],
        _kerning_factor=_kerning_factor
    )
```

```
FileNotFoundError: [Errno 2] No such file or directory: '/data/scratch/inp
ut/koreanfont/NanumBrush.ttf'
```

```
<Figure size 1200x375 with 5 Axes>
```

Define Neural Networks and Optimizers

Name	Sort
Generator	UNet
Discriminator	ResNet
Optimizer of Generator	Adam
Optimizer of Disciminator	Adam

In [71]:
```
class UNetDown(nn.Module):
    def __init__(self, in_channels, out_channels, normalize=True, dropout=0.
        super().__init__()

        layers = [nn.Conv2d(in_channels, out_channels, 4, stride=2, padding=

        if normalize:
            layers.append(nn.InstanceNorm2d(out_channels)),

        layers.append(nn.LeakyReLU(0.2))

        if dropout:
            layers.append(nn.Dropout(dropout))

        self.down = nn.Sequential(*layers)

    def forward(self, x):
        x = self.down(x)
        return x
```

In [72]:
```
class UNetUp(nn.Module):
    def __init__(self, in_channels, out_channels, dropout=0.0):
        super().__init__()

        layers = [
            nn.ConvTranspose2d(in_channels, out_channels,4,2,1,bias=False),
            nn.InstanceNorm2d(out_channels),
            nn.LeakyReLU()
        ]

        if dropout:
            layers.append(nn.Dropout(dropout))

        self.up = nn.Sequential(*layers)

    def forward(self,x,skip):
        x = self.up(x)
        x = torch.cat((x,skip),1)
        return x
```

In [73]:
```
class GeneratorUNet(nn.Module):
    def __init__(self, in_channels=1, out_channels=1):
        super().__init__()

        self.down1 = UNetDown(in_channels, 64, normalize=False)
        self.down2 = UNetDown(64,128)
        self.down3 = UNetDown(128,256)
        self.down4 = UNetDown(256,512,dropout=0.5)
        self.down5 = UNetDown(512,512,dropout=0.5)
        self.down6 = UNetDown(512,512,dropout=0.5)
        self.down7 = UNetDown(512,512,dropout=0.5)
        self.down8 = UNetDown(512,512,normalize=False,dropout=0.5)

        self.up1 = UNetUp(512,512,dropout=0.5)
        self.up2 = UNetUp(1024,512,dropout=0.5)
        self.up3 = UNetUp(1024//2,512,dropout=0.5)
        self.up4 = UNetUp(1024,512,dropout=0.5)
        self.up5 = UNetUp(1024,256)
        self.up6 = UNetUp(512,128)
        self.up7 = UNetUp(256,64)
        self.up8 = nn.Sequential(
            nn.ConvTranspose2d(128,out_channels,4,stride=2,padding=1),
            nn.Tanh()
        )

    def forward(self, x):
        d1 = self.down1(x)
        d2 = self.down2(d1)
        d3 = self.down3(d2)
        d4 = self.down4(d3)
        d5 = self.down5(d4)
        d6 = self.down6(d5)
        u1 = d6
        u2 = self.up3(u1,d5)
        u3 = self.up4(u2,d4)
        u4 = self.up5(u3,d3)
        u5 = self.up6(u4,d2)
        u6 = self.up7(u5,d1)
        u7 = self.up8(u6)

        return u7
```

In [74]:
```
class BasicBlock(nn.Module):
    expansion = 1

    def __init__(self, in_planes, planes, stride=1):
        super(BasicBlock, self).__init__()
        self.relu = nn.ReLU(False)

        self.conv1 = nn.Conv2d(
            in_planes, planes, kernel_size=3, stride=stride, padding=1, bias
        self.bn1 = nn.BatchNorm2d(planes)
        self.conv2 = nn.Conv2d(planes, planes, kernel_size=3,
                               stride=1, padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(planes)

        self.shortcut = nn.Sequential()
        if stride != 1 or in_planes != self.expansion*planes:
            self.shortcut = nn.Sequential(
                nn.Conv2d(in_planes, self.expansion*planes,
                          kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(self.expansion*planes)
            )

    def forward(self, x):
        out = self.relu(self.bn1(self.conv1(x)))
        out = self.bn2(self.conv2(out))
        out += self.shortcut(x)
        out = self.relu(out)
        return out
```

In [75]:
```
class Discriminator(nn.Module):
    def __init__(self, block, num_blocks, num_classes=1):
        super(Discriminator, self).__init__()
        self.in_planes = 64

        self.conv1 = nn.Conv2d(1, 64, kernel_size=3, stride=1, padding=1, bi
        self.bn1 = nn.BatchNorm2d(64)
        self.layer1 = self._make_layer(block, 64, num_blocks[0], stride=1)
        self.layer2 = self._make_layer(block, 128, num_blocks[1], stride=2)
        self.layer3 = self._make_layer(block, 256, num_blocks[2], stride=2)
        self.layer4 = self._make_layer(block, 512, num_blocks[3], stride=2)
        self.linear1 = nn.Linear(1*1*512*block.expansion, 1024)
        self.linear2 = nn.Linear(1024, num_classes)

        self.relu = nn.ReLU(False)
        self.sigmoid = nn.Sigmoid()
        self.avg_pool2d = nn.AvgPool2d(16, 16)

    def _make_layer(self, block, planes, num_blocks, stride):
        strides = [stride] + [1]*(num_blocks-1)
        layers = []
        for stride in strides:
            layers.append(block(self.in_planes, planes, stride))
            self.in_planes = planes * block.expansion
        return nn.Sequential(*layers)

    def forward(self, x):
        out = self.relu(self.bn1(self.conv1(x)))
        out = self.layer1(out)
        out = self.layer2(out)
        out = self.layer3(out)
        out = self.layer4(out)
        out = self.avg_pool2d(out)
        out = out.view(out.size(0), -1)
        out = self.linear1(out)
        out = self.linear2(out)
        out = self.sigmoid(out)
        return out
```

Initiate Weights and Biases

In [76]:
```
def weights_init(m):
    classname = m.__class__.__name__
    if type(m) == nn.Conv2d:
        m.weight.data.normal_(0.0, 0.02)
    elif type(m) == nn.BatchNorm2d:
        m.weight.data.normal_(1.0, 0.02)
        m.bias.data.fill_(0)
```

In [77]:
```
Generator = GeneratorUNet().to(device)
Discriminator = Discriminator(BasicBlock, [3, 4, 6, 3]).to(device)

summary_g = Generator.apply(weights_init)
summary_d = Discriminator.apply(weights_init)
```

In [78]:
```
optimizerG = Adam(Generator.parameters(), lr=lrG, betas=(beta1, beta2))
optimizerD = Adam(Discriminator.parameters(), lr=lrD, betas=(beta1, beta2))
```

In [79]:
```
img_list = []
G_loss = []
D_loss = []

FAKE_LABEL = 0.0
REAL_LABEL = 1.0
```

Define Cost Functions

In [80]:
```
l1loss = nn.L1Loss()
bceloss = nn.BCELoss()
```

Cost Function of Generator

$$\mathbf{Loss_G(x,y)} = \sum_{i=1}^{t} \gamma^{i-1} \times \left\{ \lambda_1 \times \mathbf{E}_{x,y_i} \left[\left\| \mathbf{y_i} - \mathbf{G^i(x)} \right\|_1 \right] + \mathbf{E_x} \left[\mathbf{\log(1 - D(G^i(x)))} \right] \right\}$$

t is Time Step. γ is discount factor(GAMMA). λ_1 is L1Lambda.

In [81]:
```
def generator_error(netG, netD, sketch, real, real_label, fake_label, gamma=
    def G_error(G_output, real, D_output):
        return l1loss(G_output, real)*L1lambda + bceloss(D_output, real_labe

    next_input = sketch
    error = None

    real_list = []
    for i in range(TIME_STEP):
        real_list.append(real[:,i,:,:,:])

    for ind, y in enumerate(real_list):
        G_output = netG(next_input)
        next_input = G_output.clone().detach()
        D_output = netD(G_output).view(-1)

        if ind==0:
            error = G_error(G_output, y, D_output)
        else :
            error += (gamma ** ind) * G_error(G_output, y, D_output)

        del G_output, D_output
        gc.collect()
        torch.cuda.empty_cache()

    return error
```

Cost Function of Discriminator

$$\mathbf{Loss_D(x,y) = E_x\,[logD(G(x))] + \frac{1}{t}\sum_{i=1}^{t} E_{y_i}\,[log(1 - D(G(y_i)))]}$$

In [82]:
```
def discriminator_error(netG, netD, sketch, real, real_label, fake_label, av
    output_g = netG(sketch)
    outputs_fake = netD(output_g.detach()).view(-1)
    errD = bceloss(outputs_fake, fake_label)

    del output_g, outputs_fake
    gc.collect()
    torch.cuda.empty_cache()

    for i in range(0, TIME_STEP):
        outputs_real = netD(real[:,i,:,:]).view(-1)
        if avg:
            errD += bceloss(outputs_real, real_label)/TIME_STEP
        else:
            errD += bceloss(outputs_real, real_label)
        del outputs_real
        gc.collect()
        torch.cuda.empty_cache()

    return errD
```

Apply Checkpoint

In [88]:
```
def apply_checkpoint(use_checkpoint=True):
    global Generator, Discriminator, optimizerG, optimizerD, G_Loss, D_Loss,

    if os.path.isdir(OLD_PATH) and use_checkpoint:
        checkpoint = torch.load(OLD_GENERATOR_MODEL)
        start_epoch = checkpoint['epoch']
        Generator.load_state_dict(checkpoint['model_state_dict'])
        optimizerG.load_state_dict(checkpoint['optimizer_state_dict'])

        checkpoint = torch.load(OLD_DISCRIMINATOR_MODEL)
        start_epoch = checkpoint['epoch']
        Discriminator.load_state_dict(checkpoint['model_state_dict'])
        optimizerD.load_state_dict(checkpoint['optimizer_state_dict'])

        with open(OLD_G_LOSS, 'rb') as f:
            G_loss = pickle.load(f)

        with open(OLD_D_LOSS, 'rb') as f:
            D_loss = pickle.load(f)

        print('Continue training. (Epoch : {0})'.format(start_epoch))
    else :
        print('Begin training newly.')
```

Define Train Function

In [89]:
```
nowepoch = 0
strange_error_limit = 10
strange_error_num = 0

def fit(device, num_epochs=1000):
    global nowepoch
    iters = 0
    for epoch in range(start_epoch+1, num_epochs+start_epoch+1):
        nowepoch = epoch
        print("< EPOCH{0} >".format(epoch))
        result = train_one_epoch(device, train_dataloader, Generator, Discri
        if not result:
            return


def train_one_epoch(device, dataloader, netG, netD, optimizerG, optimizerD,
    global nowpath, strange_error_num, strange_error_limit
    with torch.autograd.set_detect_anomaly(True):
        for i, data in enumerate(dataloader):
            sketch, real = data
            sketch, real = sketch.to(device), real.to(device)

            b_size = sketch.size(0)
            real_label = torch.full((b_size,), REAL_LABEL, dtype=torch.float
            fake_label = torch.full((b_size,), FAKE_LABEL, dtype=torch.float

            #Train Discriminator
            netG.eval()
            netD.train()
            netD.zero_grad()

            errD = discriminator_error(netG, netD, sketch, real, real_label,

            log('Complete calcuating of Discriminator')
            errD.backward()
            log('Complete backprogration of Discriminator')
            optimizerD.step()
            log('Complete stepping OptimizerD')

            #Train Generator
            netG.train()
            netD.eval()
            netG.zero_grad()


            errG = generator_error(netG, netD, sketch, real, real_label, fak

            log('Complete calcuating of Generator')
            errG.backward()
            log('Complete backprogration of Genereator')
            optimizerG.step()
            log('Complete stepping OptimizerG')

            del b_size, real_label, fake_label, sketch, real
            gc.collect()
            torch.cuda.empty_cache()

            #Log
            if i % 1 == 0:
                print('[%d/%d][%d/%d]\tLoss_G: %.4f\tLoss_D: %.4f'
                      % (epoch, num_epochs, i, len(dataloader),
```

```
                        errG.item(), errD.item()))



            G_loss.append(errG.item())
            D_loss.append(errD.item())

            del errG, errD
            gc.collect()
            torch.cuda.empty_cache()

            iters += 1
    return True
```

Train

In [90]:
```
apply_checkpoint(use_checkpoint=USE_CHECKPOINT)
```

```
Continue training. (Epoch : 10)
```

In [ ]:
```
summary = Generator.train()
summary = Discriminator.train()

if all_epochs>0:
    fit(device, num_epochs=all_epochs)

summary = Generator.eval()
summary = Discriminator.eval()
```

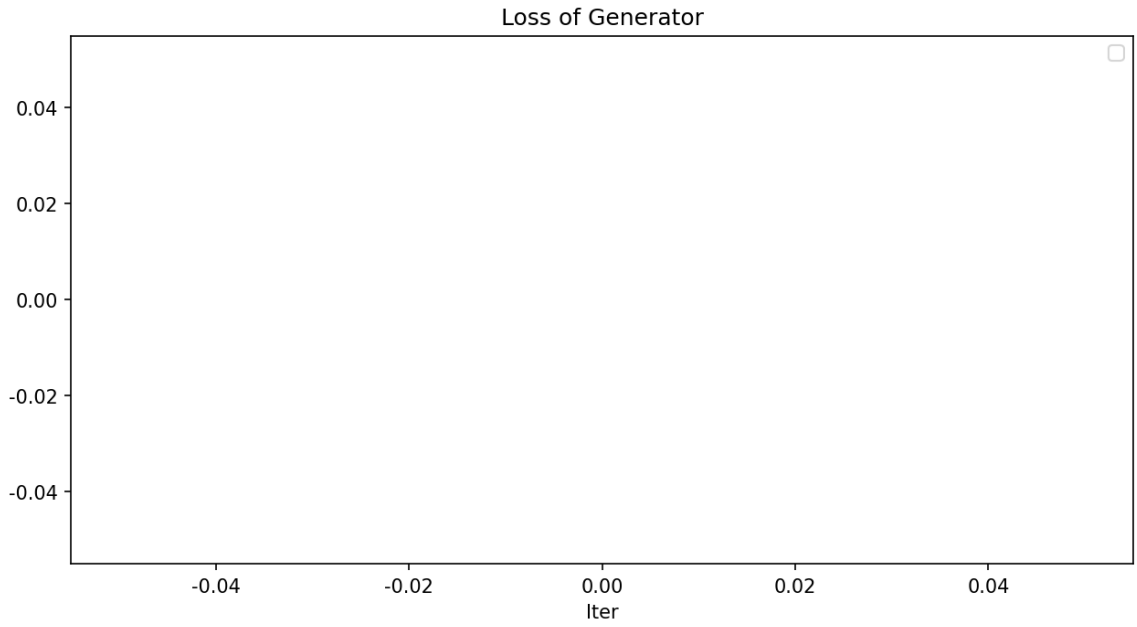
```
[1/10][957/4162]        Loss_G: 22.8949 Loss_D: 0.0000
[1/10][958/4162]        Loss_G: 23.2359 Loss_D: 0.0000
[1/10][959/4162]        Loss_G: 23.1544 Loss_D: 0.0000
[1/10][960/4162]        Loss_G: 23.2580 Loss_D: 0.0000
[1/10][961/4162]        Loss_G: 23.2907 Loss_D: 0.0000
[1/10][962/4162]        Loss_G: 23.2107 Loss_D: 0.0000
[1/10][963/4162]        Loss_G: 23.0020 Loss_D: 0.0000
[1/10][964/4162]        Loss_G: 23.1502 Loss_D: 0.0000
[1/10][965/4162]        Loss_G: 23.2708 Loss_D: 0.0000
[1/10][966/4162]        Loss_G: 22.9630 Loss_D: 0.0000
[1/10][967/4162]        Loss_G: 23.0286 Loss_D: 0.0000
[1/10][968/4162]        Loss_G: 22.9397 Loss_D: 0.0000
[1/10][969/4162]        Loss_G: 22.9144 Loss_D: 0.0000
[1/10][970/4162]        Loss_G: 23.3003 Loss_D: 0.0000
[1/10][971/4162]        Loss_G: 23.9069 Loss_D: 0.0000
[1/10][972/4162]        Loss_G: 23.5267 Loss_D: 0.0000
[1/10][973/4162]        Loss_G: 23.0350 Loss_D: 0.0000
[1/10][974/4162]        Loss_G: 22.9848 Loss_D: 0.0000
[1/10][975/4162]        Loss_G: 23.0512 Loss_D: 0.0000
```

Test

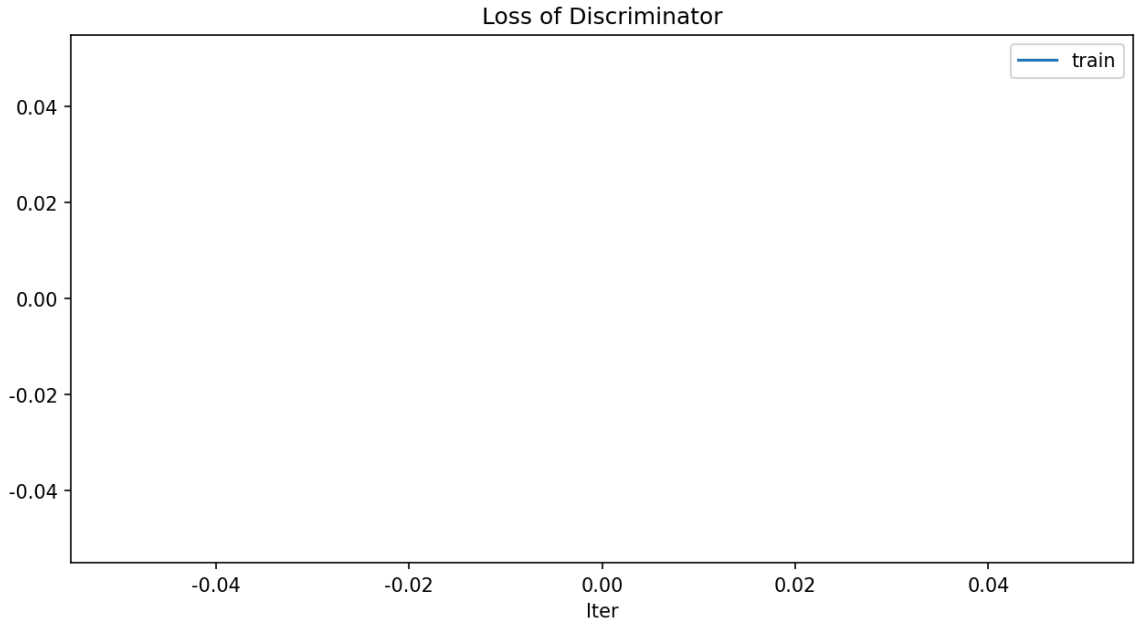
1. Calculate SSIM Score each Time Steps
2. Generate test predicted images.
3. Generate video which consist of series predicted images.

In [91]:
```
plt.figure(figsize=(10,5))
plt.title('Loss of Generator')
plt.plot(G_loss,label="")
plt.xlabel("Iter")
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose la
bel start with an underscore are ignored when legend() is called with no a
rgument.
```



In [92]:
```
plt.figure(figsize=(10,5))
plt.title('Loss of Discriminator')
plt.plot(D_loss,label="train")
plt.xlabel("Iter")
plt.legend()
plt.show()
```



In [93]:
```
def model_predict(model, time, input):
    if time%15==0 and time!=0:
        model.eval()
        num = time//15

        next_input = input
        for i in range(num):
            next_input = model(next_input).clone().detach()
        return next_input
    else:
        raise ValueError('Please set the time to a multiple of 15.')
```

In [94]:
```
from IQA_pytorch import SSIM, utils

toPILImage = transforms.ToPILImage()
ssim_model = SSIM(channels=1)

def one_time_step_ssim_score(dataloader, model, time_step, num=-1):
    model.eval()
    score = 0
    total = 0
    for ind, (x, y) in enumerate(test_dataloader_bs1_shuffle):
        x, y = x.squeeze(0).to(device), y.squeeze(0).to(device)
        outputG = model_predict(model, time_step*15, x.unsqueeze(0))

        sketch = utils.prepare_image(toPILImage(outputG.squeeze(0))).to(devi
        real = utils.prepare_image(toPILImage(y[time_step-1])).to(device)

        score += ssim_model(sketch, real, as_loss=False).item()
        total += 1

        del x, y, outputG, sketch, real
        gc.collect()
        torch.cuda.empty_cache()

        if num != -1:
            if ind+1 >= num:
                break

    print("SSIM Score of the prediction {0} minutes later : {1}".format(time
    return score/total

for ind in range(1, TEST_TIME_STEP+1):
    one_time_step_ssim_score(test_dataloader_bs1_shuffle, Generator, ind, nu
```

```
SSIM Score of the prediction 15 minutes later : 0.9561838684976101
SSIM Score of the prediction 30 minutes later : 0.9272922943532467
SSIM Score of the prediction 45 minutes later : 0.9059086436629296
SSIM Score of the prediction 60 minutes later : 0.8918968372046947
SSIM Score of the prediction 75 minutes later : 0.8797581914663315
SSIM Score of the prediction 90 minutes later : 0.8700819456875324
```

< SSIM Score of normal Pix2Pix > *(check "Pix2Pix (Compared to OracleGAN) (https://www.kaggle.com/lapl04/pix2pix-compared-to-oraclegan)")*

Prediction	SSIM Score
prediction 15 minutes later	0.788236691981554
prediction 30 minutes later	0.6658438920378685
prediction 45 minutes later	0.5865897158235311
prediction 60 minutes later	0.5096726692765952
prediction 75 minutes later	0.42021833929419516
prediction 90 minutes later	0.3831377309216186

In [95]:
```
import zipfile

y_nums = 40
iter = 0

ai_noseries_ls = []
real_noseries_ls = []

start_ind = 200

for ind, (x, y) in enumerate(test_dataloader_bs1_shuffle):
    if ind < start_ind:
        continue

    iter += 1

    x, y = x.to(device), y[0].to(device)

    outputg = Generator(x).to(cpu_device)

    outputg = outputg*127.5+127.5
    realimage = y*127.5+127.5

    cv2.imwrite('./AI_NOSERIES_Answer{0}.png'.format(ind+1), torch_tensor_to
    cv2.imwrite('./Real_NOSERIES{0}.png'.format(ind+1), torch_tensor_to_plt(

    ai_noseries_ls.append('./AI_NOSERIES_Answer{0}.png'.format(ind+1))
    real_noseries_ls.append('./Real_NOSERIES{0}.png'.format(ind+1))

    if iter > y_nums:
        break

with zipfile.ZipFile("ai_noseries.zip", 'w') as my_zip:
    for i in ai_noseries_ls:
        my_zip.write(i)
    my_zip.close()


with zipfile.ZipFile("real_noseries.zip", 'w') as my_zip:
    for i in real_noseries_ls:
        my_zip.write(i)
    my_zip.close()

for file in (ai_noseries_ls + real_noseries_ls):
    os.remove(file)

print('NOSERIES Images are generated.')
```

```
NOSERIES Images are generated.
```

In [96]:
```
import zipfile

y_nums = 40
iter = 0

ai_series_ls = []
real_series_ls = []

next_input = None
start_ind = 200

for ind, (x, y) in enumerate(test_dataloader_bs1_noshuffle):
    if ind < start_ind:
        continue

    iter += 1

    if ind == start_ind:
        next_input = x.clone().detach().to(device)
        cv2.imwrite('./Input_SERIES.png', torch_tensor_to_plt(next_input.to(

    x, y = x.to(device), y[0].to(device)

    outputg_series = Generator(next_input).to(cpu_device)
    next_input = outputg_series.clone().detach().to(device)

    outputg_series = outputg_series * 127.5 + 127.5
    realimage = y*127.5+127.5

    cv2.imwrite('./AI_SERIES_Answer{0}.png'.format(ind+1), torch_tensor_to_p
    cv2.imwrite('./Real_SERIES{0}.png'.format(ind+1), torch_tensor_to_plt(re

    ai_series_ls.append('./AI_SERIES_Answer{0}.png'.format(ind+1))
    real_series_ls.append('./Real_SERIES{0}.png'.format(ind+1))

    if iter > y_nums:
        break

with zipfile.ZipFile("ai_series.zip", 'w') as my_zip:
    for i in ai_series_ls:
        my_zip.write(i)
    my_zip.close()

with zipfile.ZipFile("real_series.zip", 'w') as my_zip:
    for i in real_series_ls:
        my_zip.write(i)
    my_zip.close()



print('SERIES Images are generated')
```

```
SERIES Images are generated
```

In [97]:
```
v1 = cv2.VideoWriter('oraclegan_series.mp4',cv2.VideoWriter_fourcc(*'DIVX'),
for name in ai_series_ls:
    v1.write(cv2.imread(name))
v1.release()

v2 = cv2.VideoWriter('real_series.mp4',cv2.VideoWriter_fourcc(*'DIVX'), 3, (
for name in real_series_ls:
    v2.write(cv2.imread(name))
v2.release()

print('Videos are generated')
print('video path : "./oraclegan_series.mp4" and "./real_series.mp4"')
```

```
Videos are generated
video path : "./oraclegan_series.mp4" and "./real_series.mp4"

OpenCV: FFMPEG: tag 0x58564944/'DIVX' is not supported with codec id 12 an
d format 'mp4 / MP4 (MPEG-4 Part 14)'
OpenCV: FFMPEG: fallback to use tag 0x7634706d/'mp4v'
OpenCV: FFMPEG: tag 0x58564944/'DIVX' is not supported with codec id 12 an
d format 'mp4 / MP4 (MPEG-4 Part 14)'
OpenCV: FFMPEG: fallback to use tag 0x7634706d/'mp4v'
```

In [98]:
```
urllib.request.urlretrieve('https://storage.googleapis.com/kaggle-script-ver
print('Videos are saved')
print('video path : "./pix2pix_series.mp4"')
```

```
---------------------------------------------------------------------------
HTTPError                                 Traceback (most recent call last)
Cell In[98], line 1
----> 1 urllib.request.urlretrieve('https://storage.googleapis.com/kaggle-script-versions/80102153/output/pix2pix_series.mp4?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=databundle-worker-v2%40kaggle-161607.iam.gserviceaccount.com%2F20211118%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20211118T232151Z&X-Goog-Expires=345599&X-Goog-SignedHeaders=host&X-Goog-Signature=8cb00c08f73986d99bac385f6910658fe1b34ee66e9e9127ba169245b0e063860cefefb9816971bf362075f1183c111df561b01a660841d6207a834dd645b21e501a7fabe15d5f82946bf07286da07bf97b7c3859f83e7c4c28cc5d8f224c353dad78e43c355845c0067fe4658d431940b320f828224c5a2a85542ece1fde70b3845468ab268a69794420f599be28d2ff09ad76142e90185567b00ad6f667bb4e751d4e871640131944954d2ac60349e0df27e0b3cdff9d318a31512538ae7024ae1b9098fa5ac6d358aeb501f533d0bbd94851b756fe34028fe88f2ef335d683d78d276d25c0f0490ed7508a6c9f4878f795ab1ae252d0419f176e1df455528', './pix2pix_series.mp4')
      2 print('Videos are saved')
      3 print('video path : "./pix2pix_series.mp4"')

File /usr/lib/python3.10/urllib/request.py:241, in urlretrieve(url, filename, reporthook, data)
    224 """
    225 Retrieve a URL into a temporary location on disk.
    226
   (...)
    237 data file as well as the resulting HTTPMessage object.
    238 """
    239 url_type, path = _splittype(url)
--> 241 with contextlib.closing(urlopen(url, data)) as fp:
    242     headers = fp.info()
    244     # Just return the local path and the "headers" for file://
    245     # URLs. No sense in performing a copy unless requested.

File /usr/lib/python3.10/urllib/request.py:216, in urlopen(url, data, timeout, cafile, capath, cadefault, context)
    214 else:
    215     opener = _opener
--> 216 return opener.open(url, data, timeout)

File /usr/lib/python3.10/urllib/request.py:525, in OpenerDirector.open(self, fullurl, data, timeout)
    523 for processor in self.process_response.get(protocol, []):
    524     meth = getattr(processor, meth_name)
--> 525     response = meth(req, response)
    527 return response

File /usr/lib/python3.10/urllib/request.py:634, in HTTPErrorProcessor.http_response(self, request, response)
    631 # According to RFC 2616, "2xx" code indicates that the client's
    632 # request was successfully received, understood, and accepted.
    633 if not (200 <= code < 300):
--> 634     response = self.parent.error(
    635         'http', request, response, code, msg, hdrs)
    637 return response

File /usr/lib/python3.10/urllib/request.py:563, in OpenerDirector.error(self, proto, *args)
    561 if http_err:
    562     args = (dict, 'default', 'http_error_default') + orig_args
```

```
--> 563     return self._call_chain(*args)

File /usr/lib/python3.10/urllib/request.py:496, in OpenerDirector._call_ch
ain(self, chain, kind, meth_name, *args)
    494 for handler in handlers:
    495     func = getattr(handler, meth_name)
--> 496     result = func(*args)
    497     if result is not None:
    498         return result

File /usr/lib/python3.10/urllib/request.py:643, in HTTPDefaultErrorHandle
r.http_error_default(self, req, fp, code, msg, hdrs)
    642 def http_error_default(self, req, fp, code, msg, hdrs):
--> 643     raise HTTPError(req.full_url, code, msg, hdrs, fp)

HTTPError: HTTP Error 400: Bad Request
```

In [ ]:
```
for file in (ai_series_ls + real_series_ls):
    os.remove(file)
```

Save Checkpoint

In [ ]:
```
torch.save({
            'epoch': nowepoch,
            'model_state_dict': Generator.state_dict(),
            'optimizer_state_dict': optimizerG.state_dict(),
            }, 'Generator.pth')

torch.save({
            'epoch': nowepoch,
            'model_state_dict': Discriminator.state_dict(),
            'optimizer_state_dict': optimizerD.state_dict(),
            }, 'Discriminator.pth')
```

In [ ]:
```
with open('./gloss.txt', 'wb') as f:
    pickle.dump(G_loss, f)
with open('./dloss.txt', 'wb') as f:
    pickle.dump(D_loss, f)
```

In [ ]:
```

```