Fashion trend prediction & style inspiration using object localization and GAN

I-Tsun Cheng and Yang Soo Yoon

Problem

Fashion industry faces a constantly changing consumer taste in fashion as more fashionistas and teenagers of the younger generation look for new outfits to wear daily.

Since large fashion companies need continuous huge profits to be able to continue survive in the industry, they need to come up with new designs that will generate their income and satisfy the demanding and erratic tastes of the consumers.

Our Aim

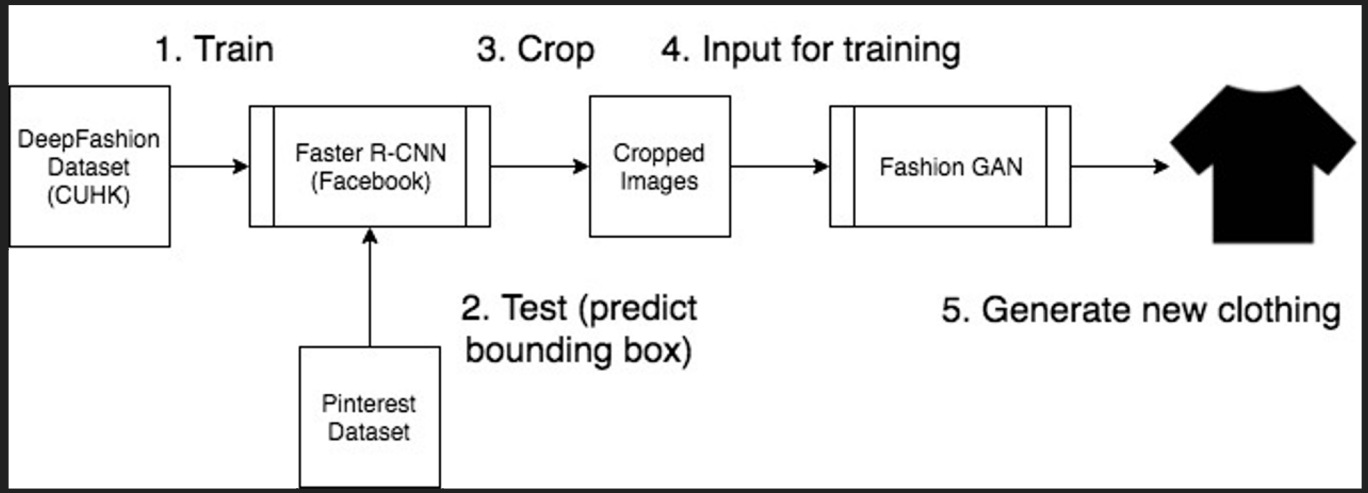
To generate new designs using existing patterns that can be found online

Our model consists of two parts:

Part 1: Clothes Detection Model

Part 2: GAN

Overall structure of the project



DeepFashion Dataset

800,000 diverse fashion images ranging from well-posed shop images to unconstrained consumer photos.

Each image in this dataset is labeled with 50 categories, 1,000 descriptive attributes, bounding box and clothing landmarks.

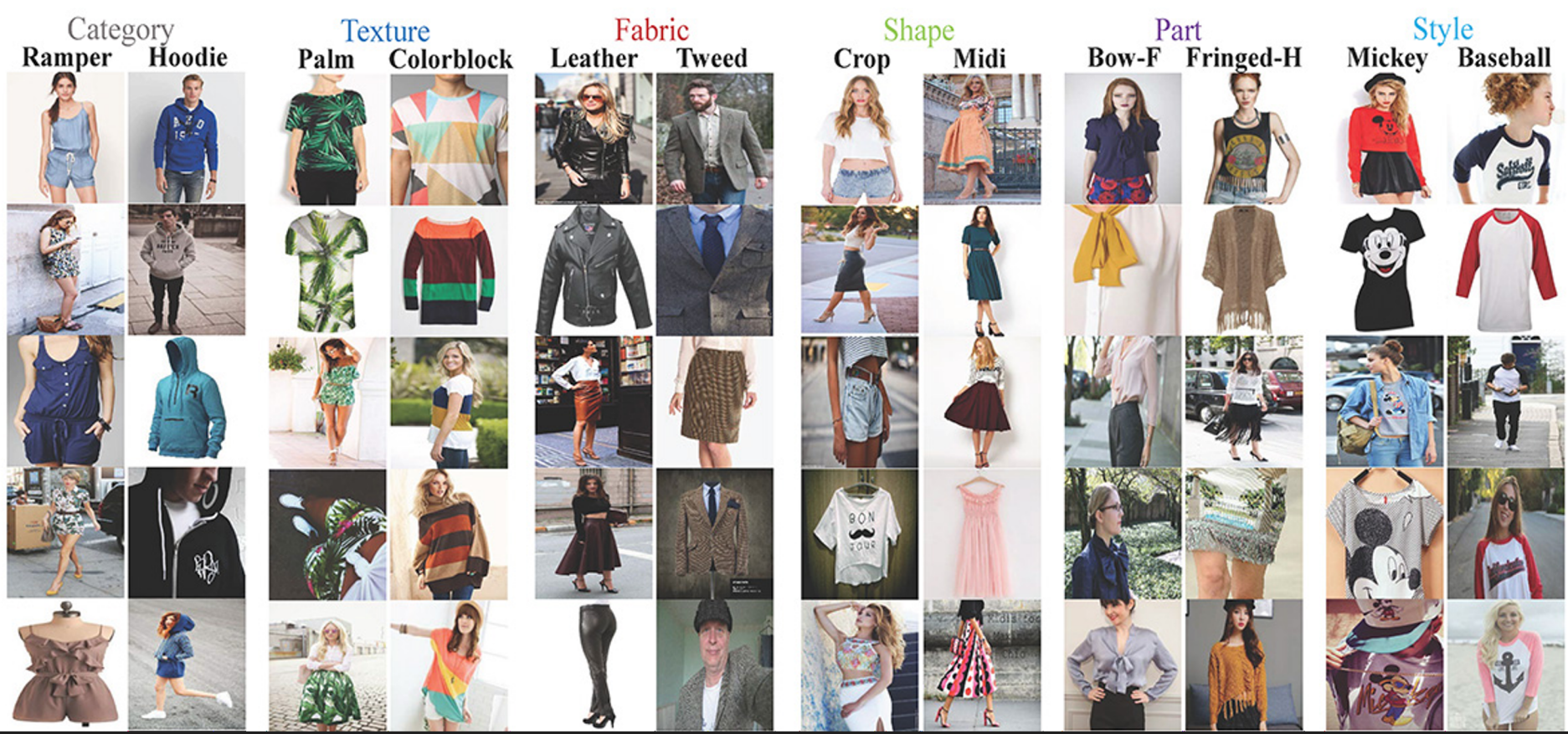
Four benchmarks are developed using the DeepFashion database, including Attribute Prediction, Consumer-to-shop Clothes Retrieval, In-shop Clothes Retrieval, and Landmark Detection.

Category and Attribute Prediction Benchmark

289,222 number of clothes images

50 number of clothing categories, and 1,000 number of clothing attributes

Each image is annotated by bounding box and clothing type

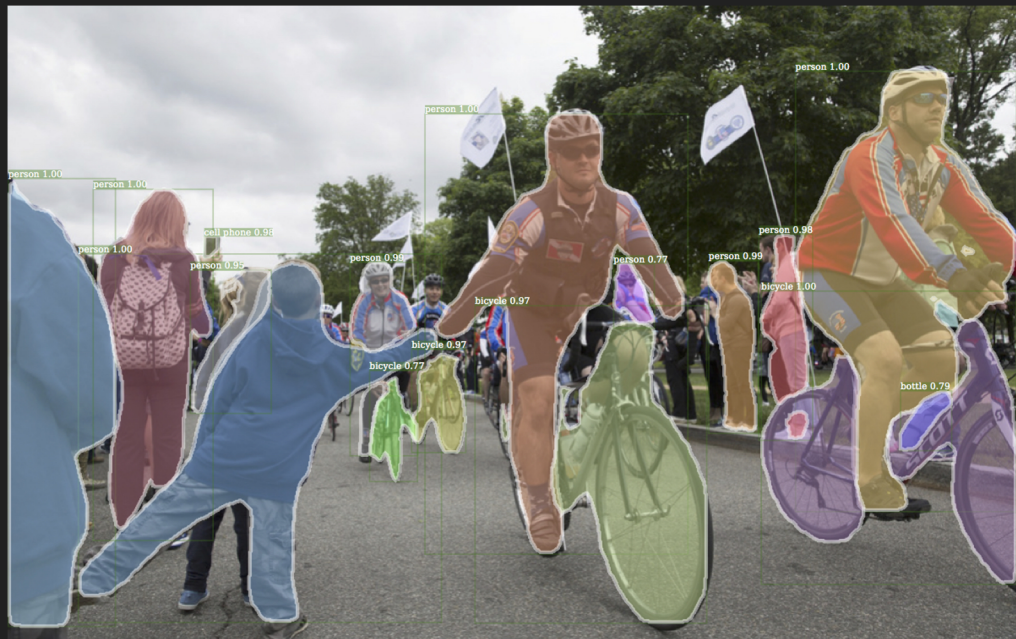

Category
Ramper
Hoodie
Texture
Palm
Colorblock
Fabric
Leather
Tweed
Shape
Crop
Midi
Part
Bow-F
Fringed-H
Style
Mickey
Baseball

Clothes Detection

Facebook’s object detection framework, Detectron

Provides a lot of architecture/models:

- Mask RCNN
- RetinaNet
- Faster R-CNN
- RPN
- Fast R-CNN
- R-FCN



Steps to train

Goal is to predict bounding boxes and category of clothing (full, upper, lower)

1. Change DeepFashion dataset to CoCo-style
2. Setup the configuration file that defines the parameters/settings
3. Train

```yaml
MODEL:
  TYPE: generalized_rcnn
  CONV_BODY: FPN.add_fpn_ResNet50_conv5_body
  NUM_CLASSES: 4
  FASTER_RCNN: True
NUM_GPUS: 1
SOLVER:
  WEIGHT_DECAY: 0.0001
  LR_POLICY: steps_with_decay
  BASE_LR: 0.0025
  GAMMA: 0.1
  MAX_ITER: 360000
  STEPS: [0, 240000, 320000]
FPN:
  FPN_ON: True
  MULTILEVEL_ROIS: True
  MULTILEVEL_RPN: True
FAST_RCNN:
  ROI_BOX_HEAD: fast_rcnn_heads.add_roi_2mlp_head
  ROI_XFORM_METHOD: RoIAlign
  ROI_XFORM_RESOLUTION: 7
  ROI_XFORM_SAMPLING_RATIO: 2
TRAIN:
  WEIGHTS: https://dl.fbaipublicfiles.com/detectron/ImageNetPretrained/MSRA/R-50.pkl
  DATASETS: ('coco_deepfashion3_train',)
  SCALES: (800,)
  MAX_SIZE: 1333
  BATCH_SIZE_PER_IM: 512
  RPN_PRE_NMS_TOP_N: 2000  # Per FPN level
TEST:
  DATASETS: ('coco_deepfashion3_val',)
  SCALE: 800
  MAX_SIZE: 1333
  NMS: 0.5
  RPN_PRE_NMS_TOP_N: 1000  # Per FPN level
  RPN_POST_NMS_TOP_N: 1000
OUTPUT_DIR: .
```

Validation Results

Average Precision (AP) @[IoU=0.50:0.95 | area=   all | maxDets=100] = 0.721

Average Precision (AP) @[IoU=0.50 | area=   all | maxDets=100] = 0.922

Average Precision (AP) @[IoU=0.75 | area=   all | maxDets=100] = 0.843

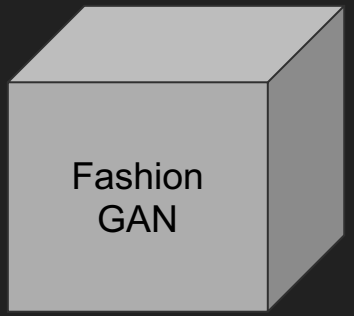
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100] = 0.094

Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100] = 0.466

Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100] = 0.744

Sample predictions

Crop and pass them to GAN for training

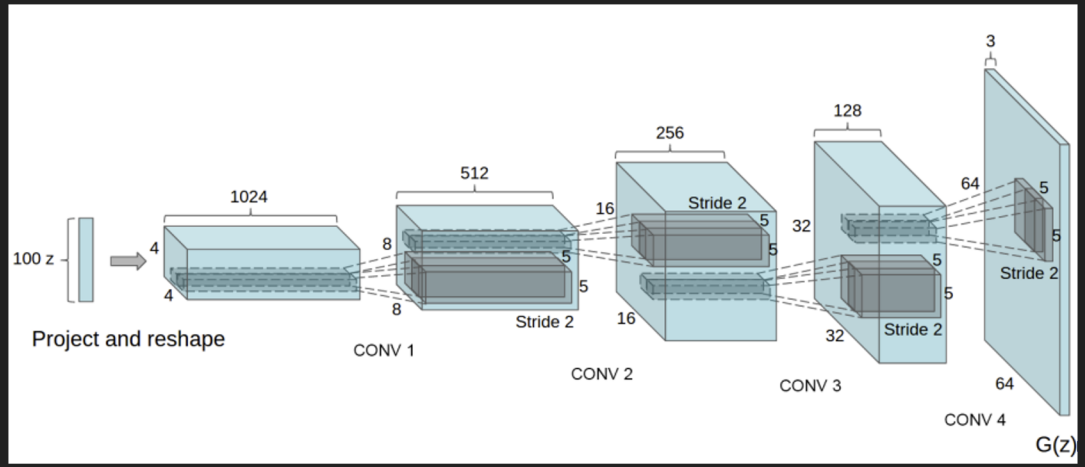


GAN Structure

Generator:

5 Layers, increasing output channel size

Convolutional layer, batch norm layer, relu activation layer



GAN Structure

Discriminator:

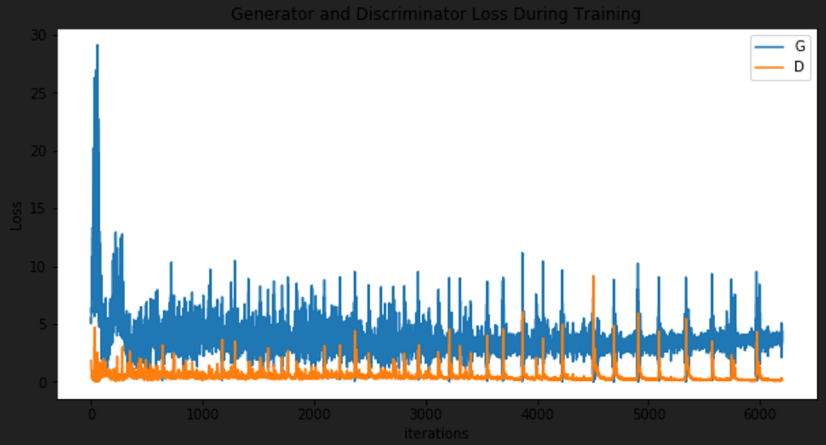
Convolutional layer, batch norm layer, leaky ReLU layer

Downsampling with strided convolution(best practice) vs Pooling

Batchnorm & Leaky ReLU gradient flow

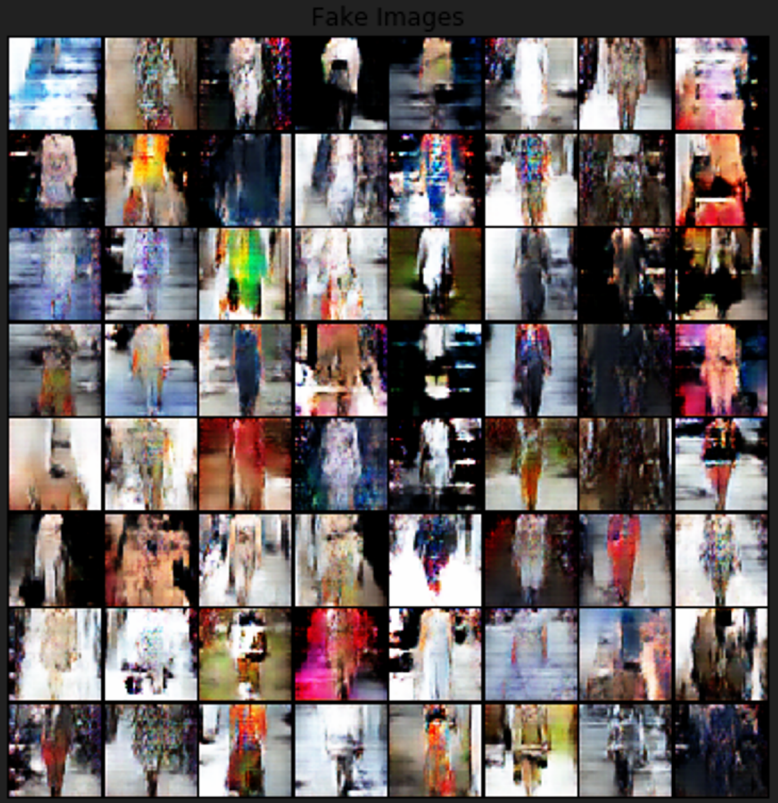
Parameter Comparisons

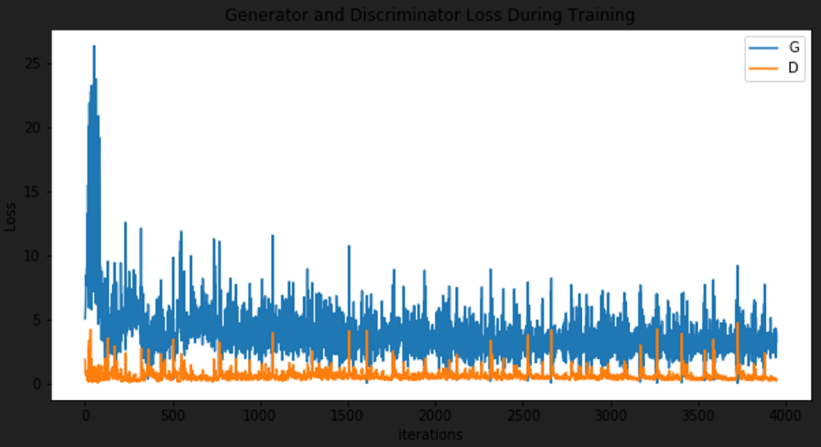
64 x 64



Loss_D: 0.1564 Loss_G: 3.7075
D(x): 0.9183 D(G(z)): 0.0621 / 0.0456

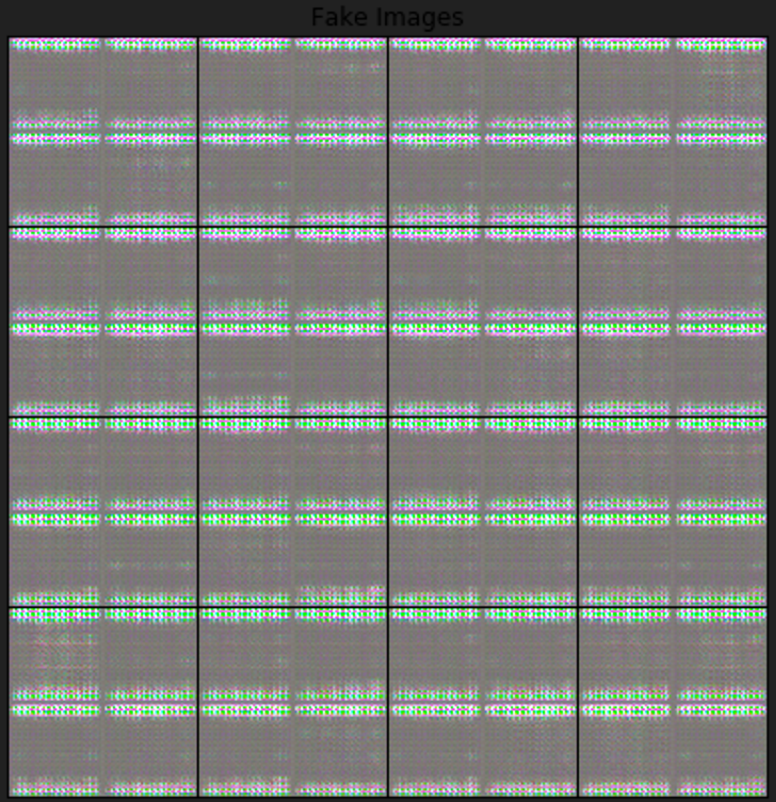
64 x 64 (Original Data)

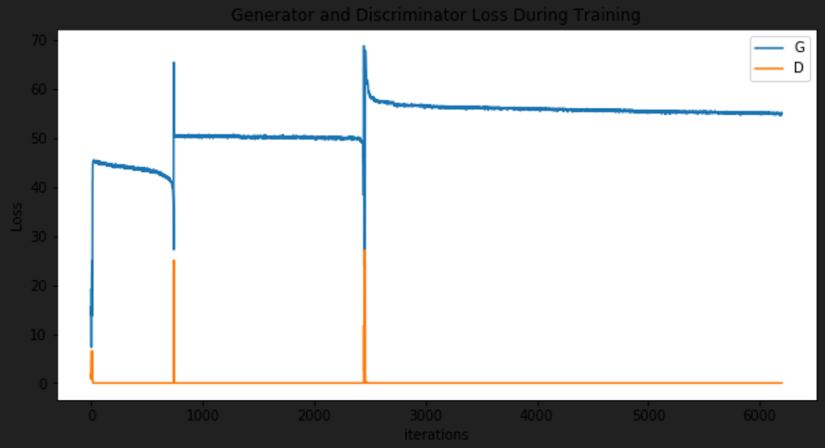




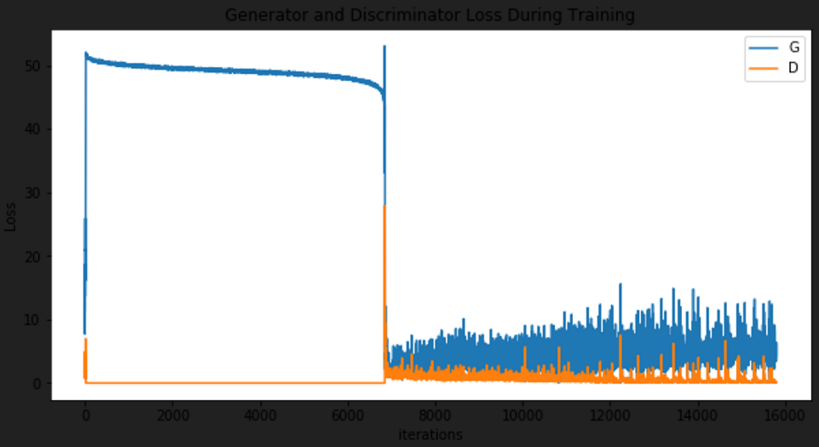

```
Loss_D: 0.3935 Loss_G: 3.2441
D(x): 0.8563 D(G(z)): 0.1848 / 0.0587
```

128 x 128


Fake Images


Generator and Discriminator Loss During Training
G
D
Loss
iterations
0
10
20
30
40
50
60
70
0
1000
2000
3000
4000
5000
6000

128 x 128 (Original Data)



Loss_D: 0.1592 Loss_G: 4.3941
D(x): 0.9530 D(G(z)): 0.0909 / 0.0285

Future Improvements