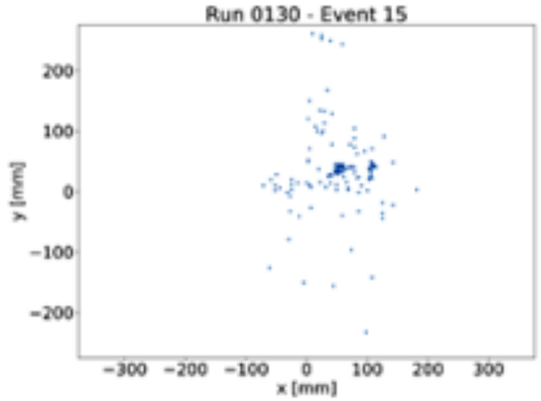
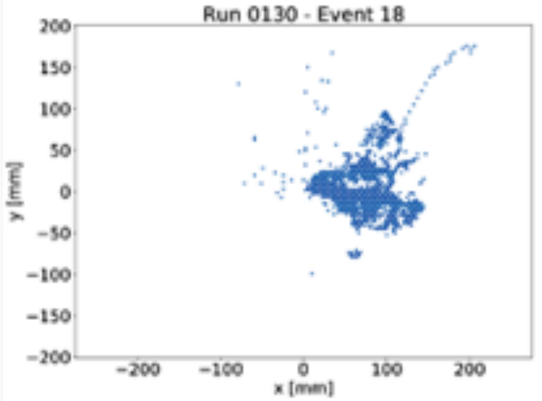
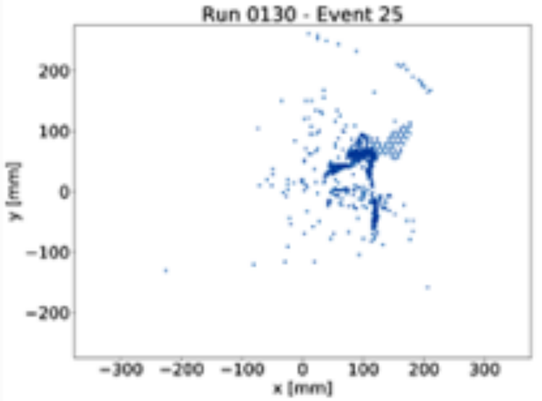
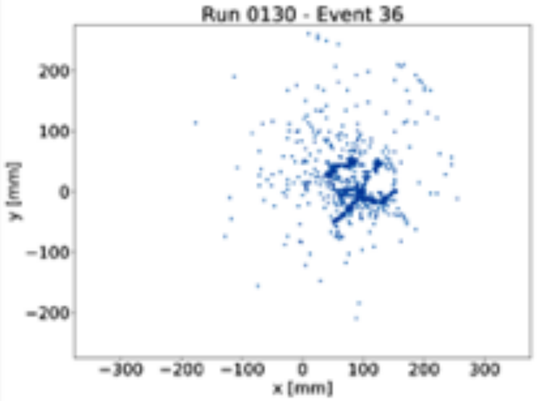
GENERATIVE MODELS

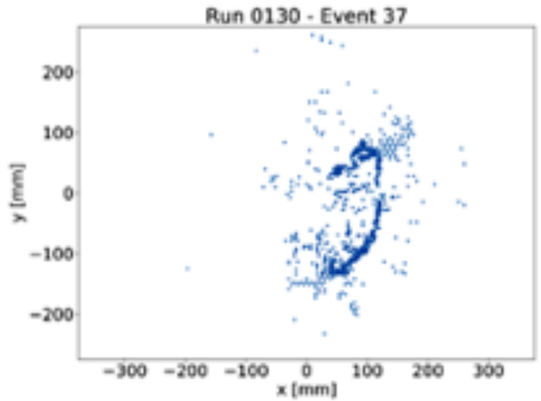
AUTOENCODERS

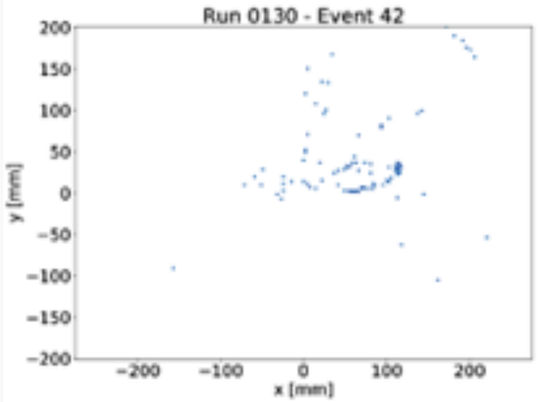

Run 0130 - Event 15
y [mm]
x [mm]

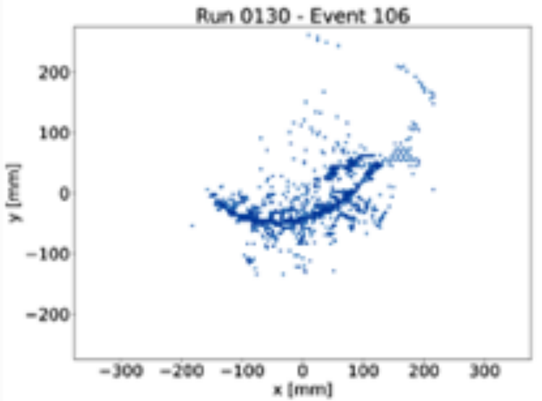

Run 0130 - Event 18
y [mm]
x [mm]

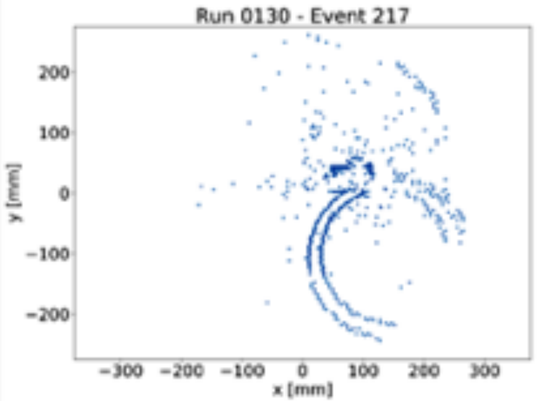

Run 0130 - Event 25
y [mm]
x [mm]

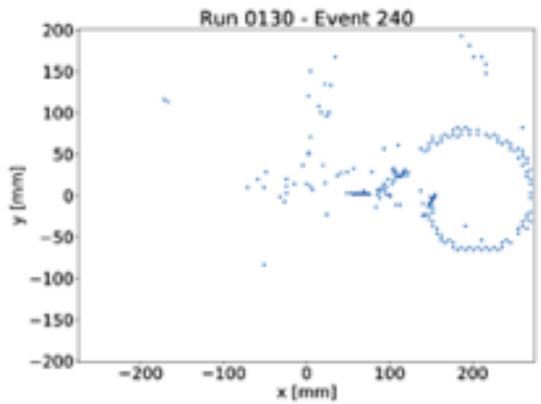

Run 0130 - Event 36
y [mm]
x [mm]

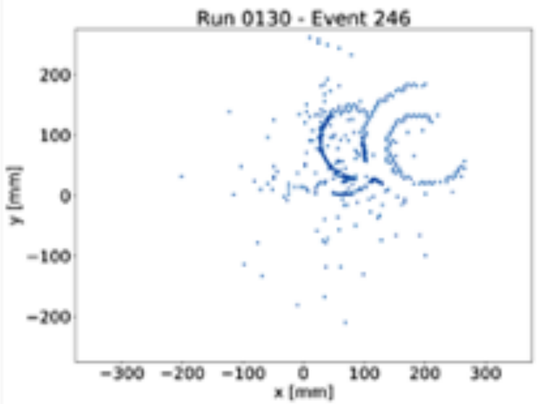

Run 0130 - Event 37
y [mm]
x [mm]

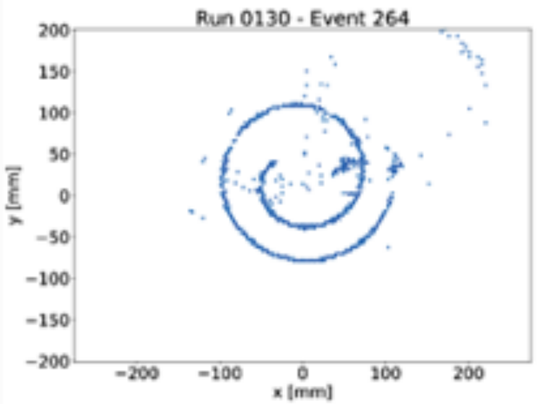

Run 0130 - Event 42
y [mm]
x [mm]

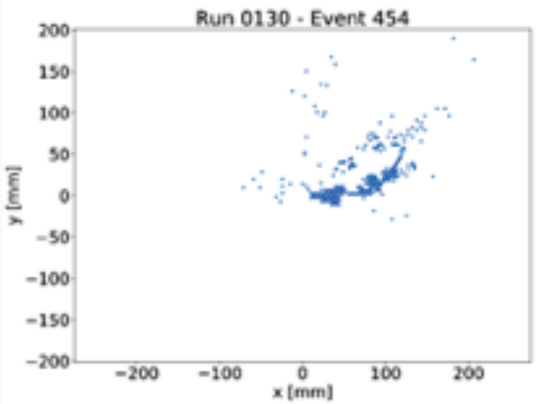

Run 0130 - Event 106
y [mm]
x [mm]

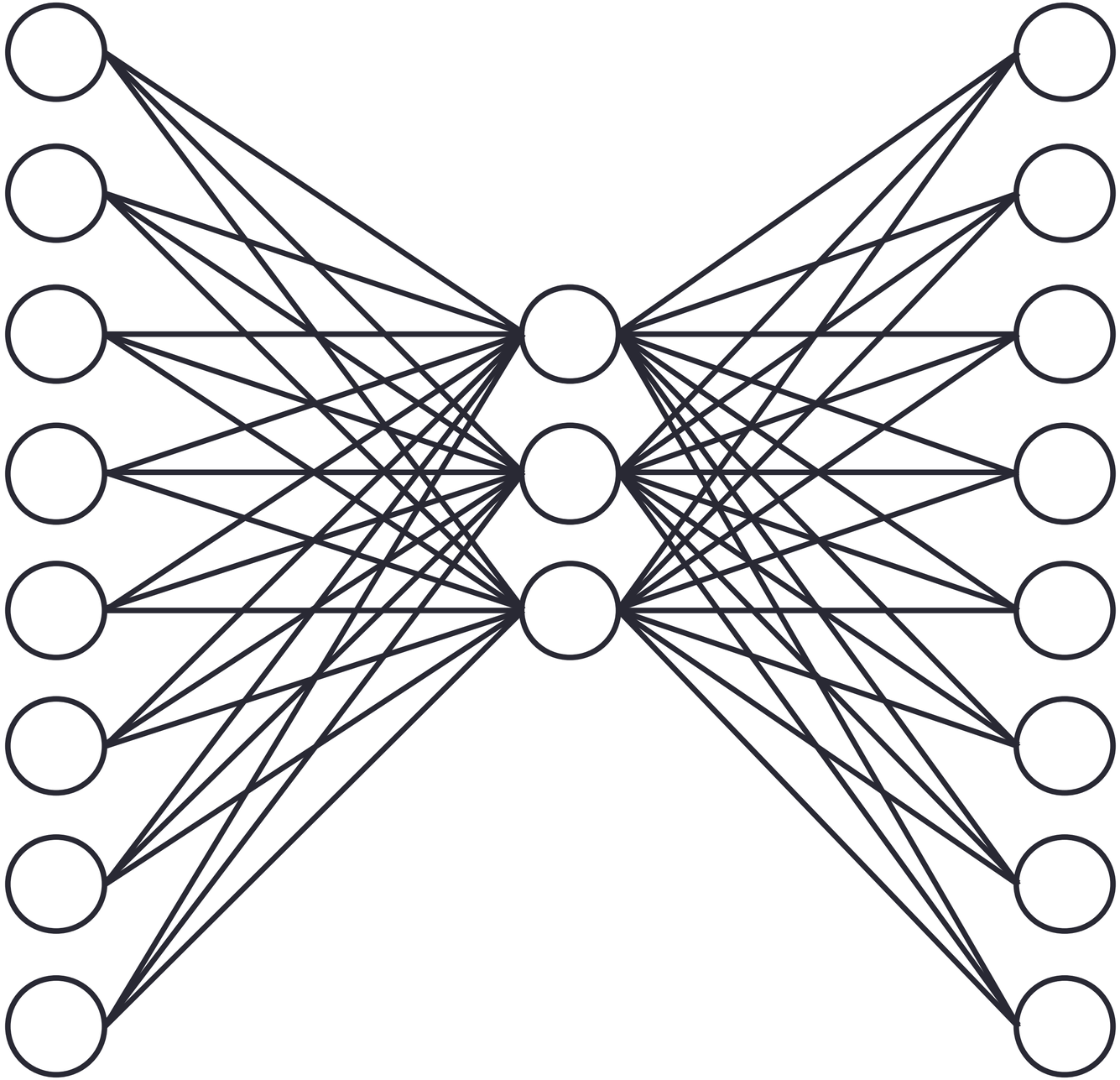

Run 0130 - Event 217
y [mm]
x [mm]


Run 0130 - Event 240
y [mm]
x [mm]

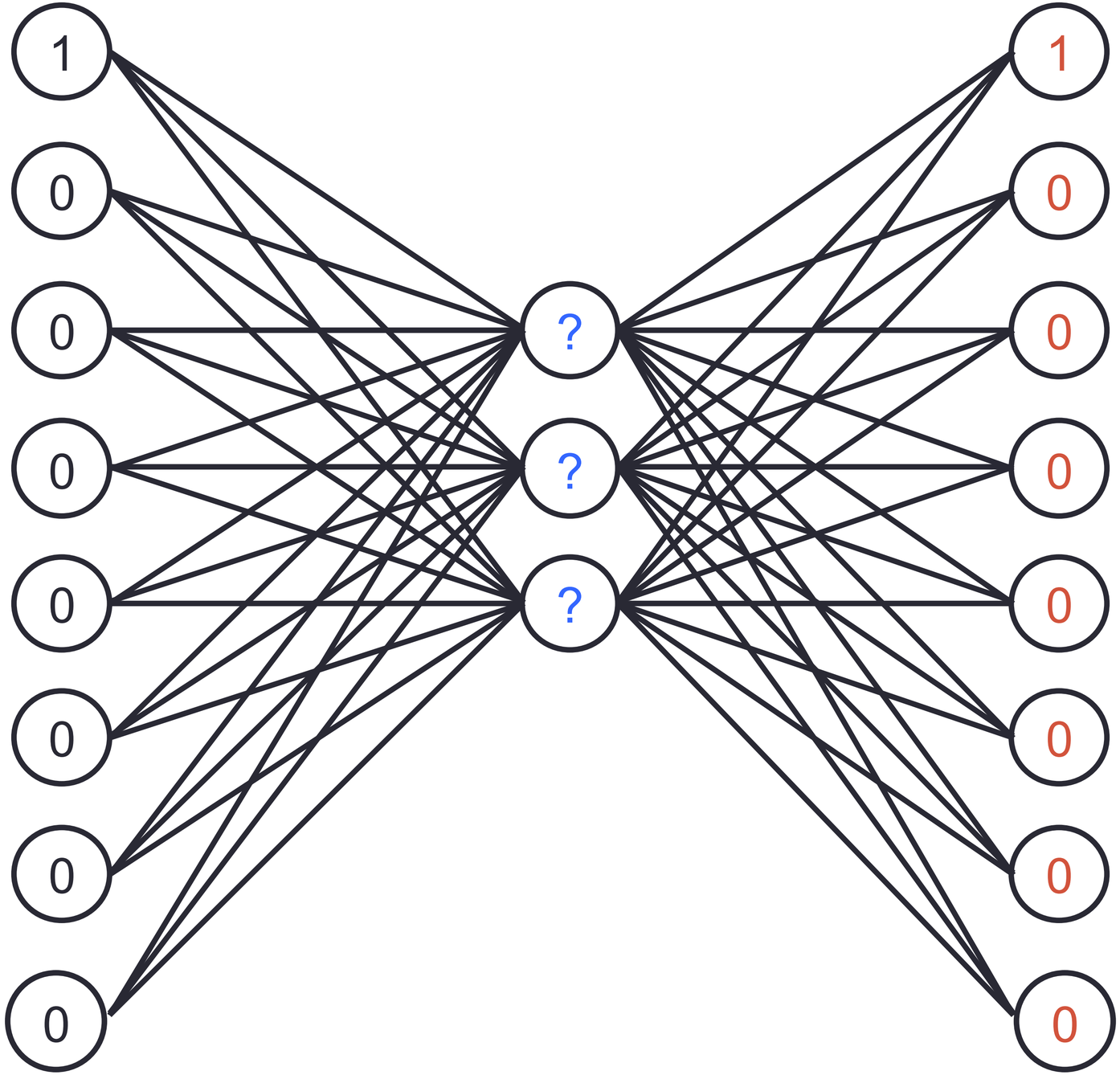

Run 0130 - Event 246
y [mm]
x [mm]


Run 0130 - Event 264
y [mm]
x [mm]

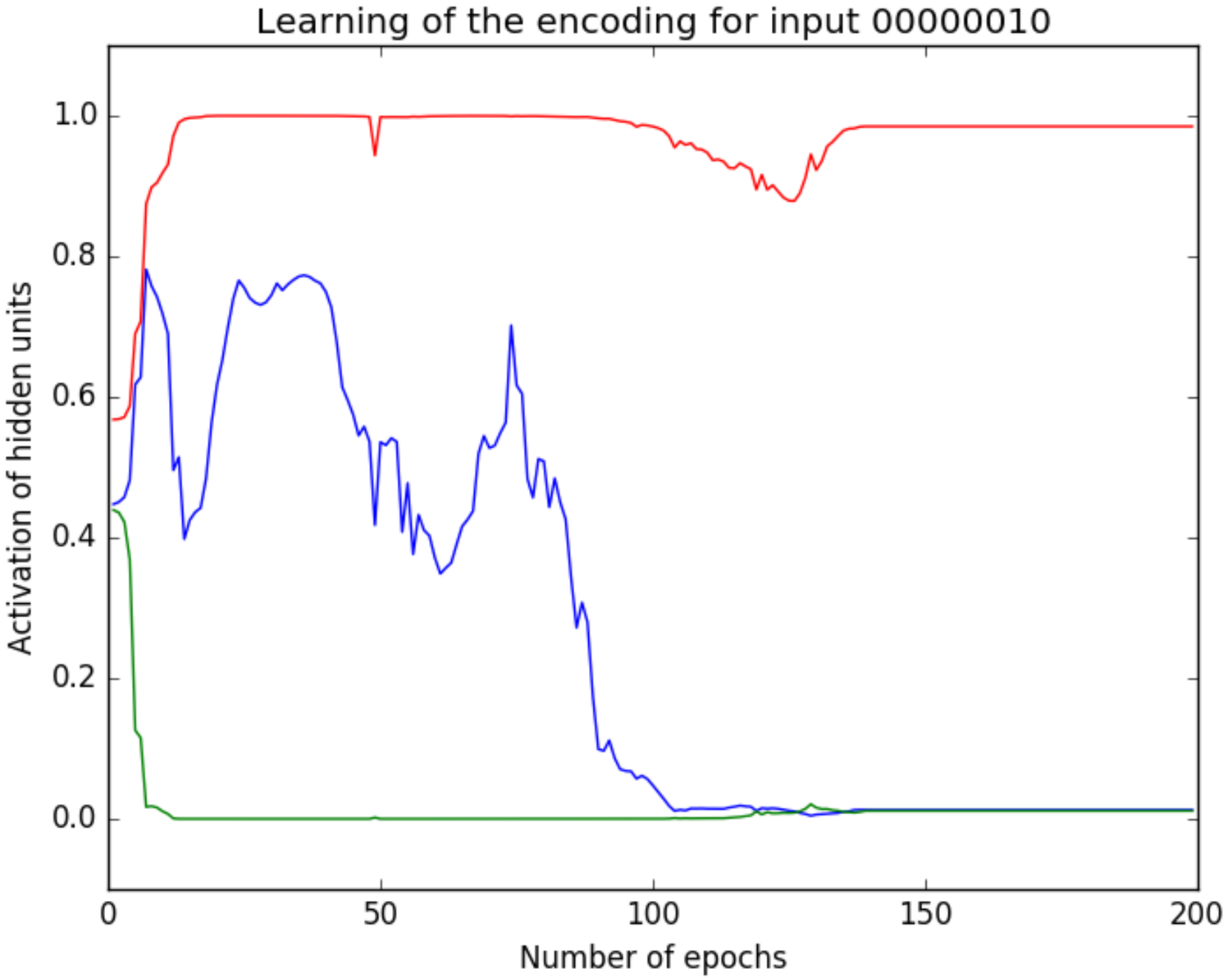

Run 0130 - Event 454
y [mm]
x [mm]



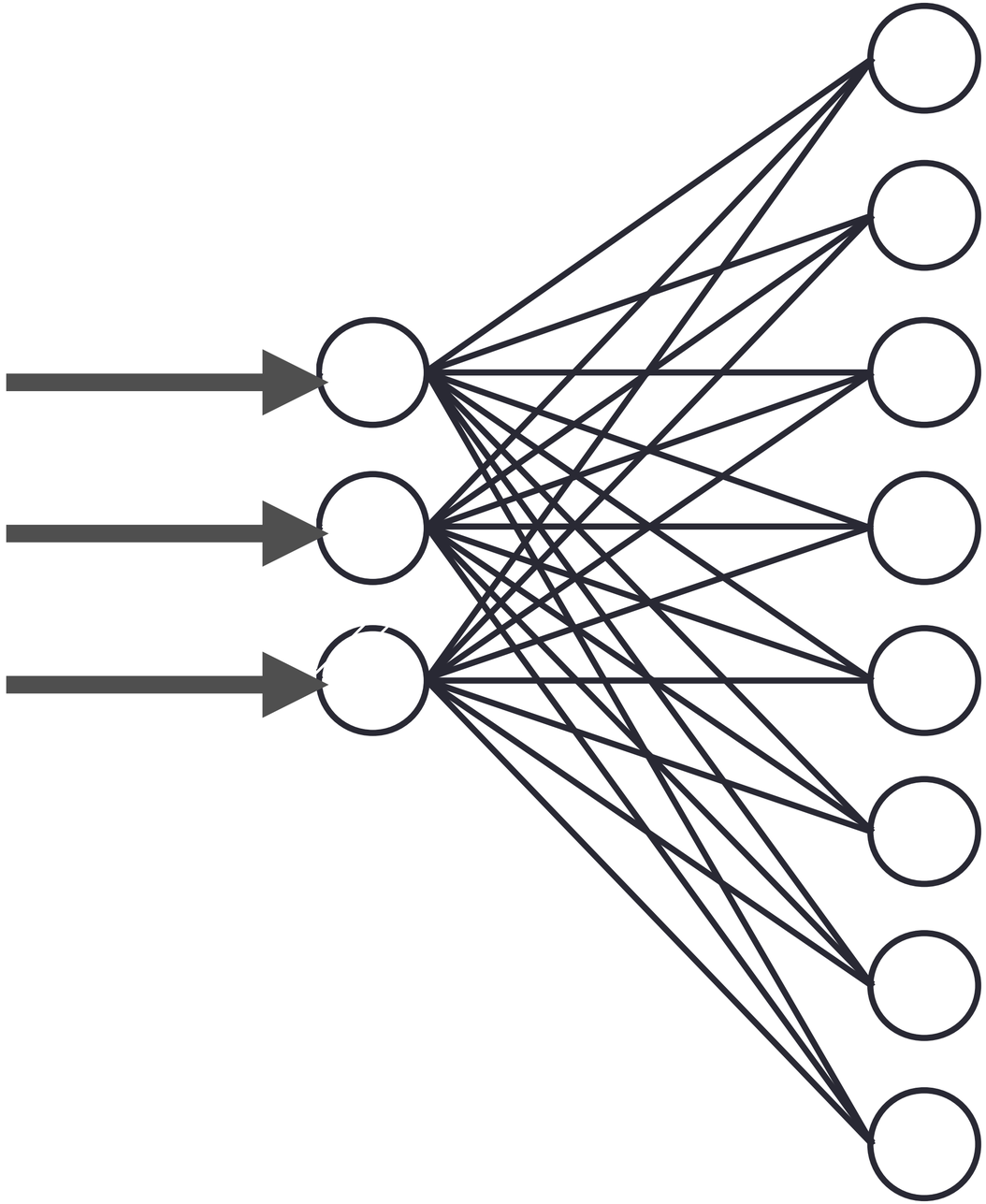
Input	Output
10000000	10000000
01000000	01000000
00100000	00100000
00010000	00010000
00001000	00001000
00000100	00000100

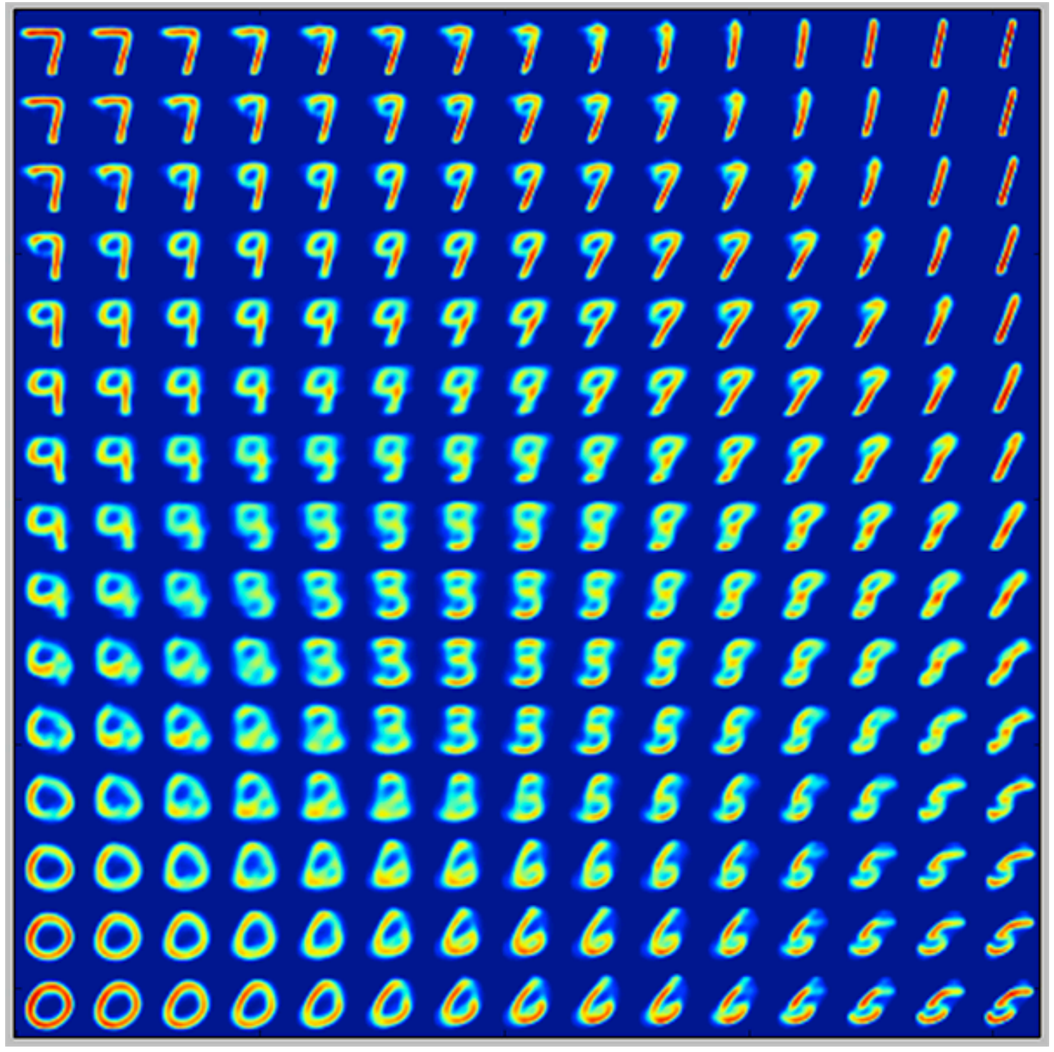

1
0
0
0
0
0
0
0
?
?
?
1
0
0
0
0
0
0
0

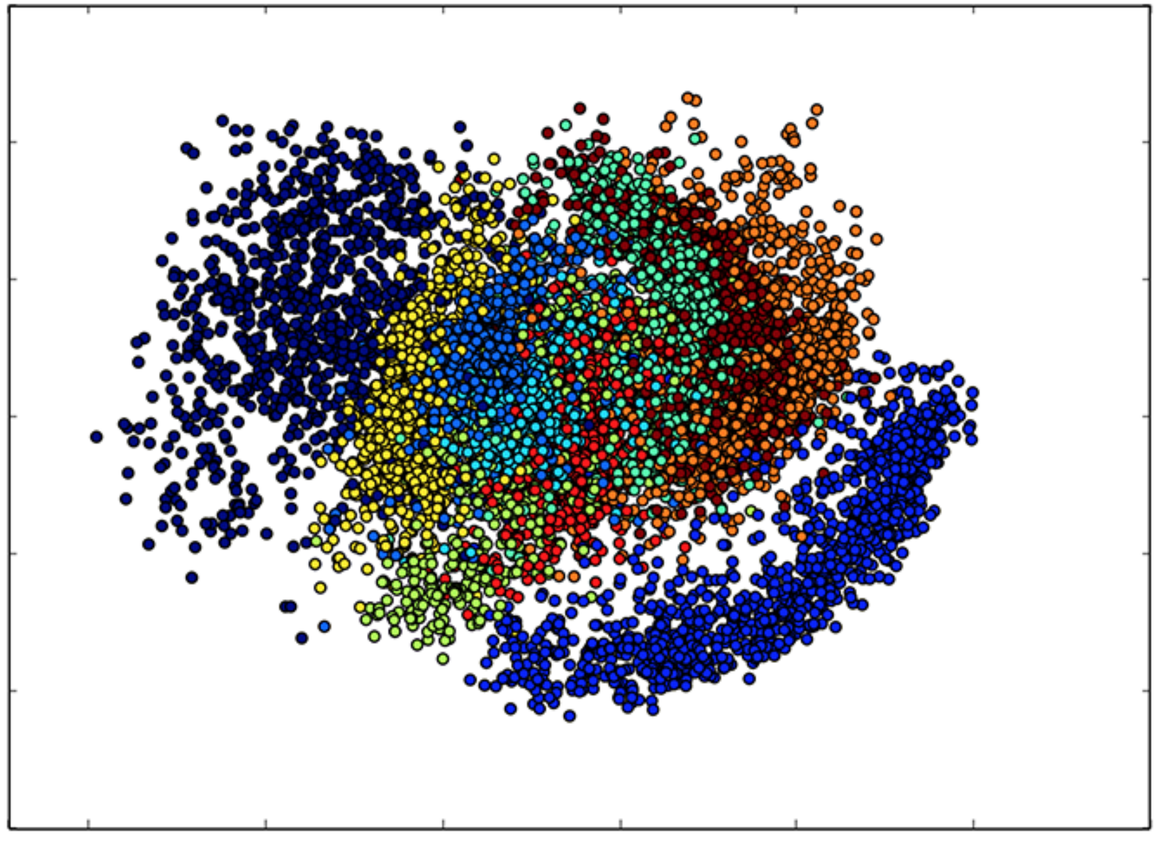
Input	A1	A2	A3	Output
10000000	0.9911	0.9869	0.0093	10000000
01000000	0.9892	0.0095	0.0124	01000000
00100000	0.0094	0.0283	0.0122	00100000
00010000	0.9840	0.9836	0.9900	00010000
00001000	0.0139	0.9904	0.0186	00001000
00000100	0.0128	0.9805	0.9868	00000100


Learning of the encoding for input 00000010
Activation of hidden units
Number of epochs
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200

VARIATIONAL AUTOENCODERS (VAES)





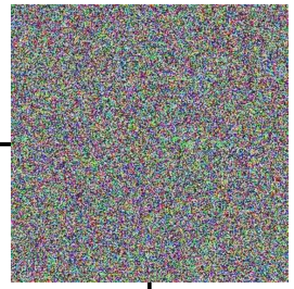


https://blog.keras.io/building-autoencoders-in-keras.html

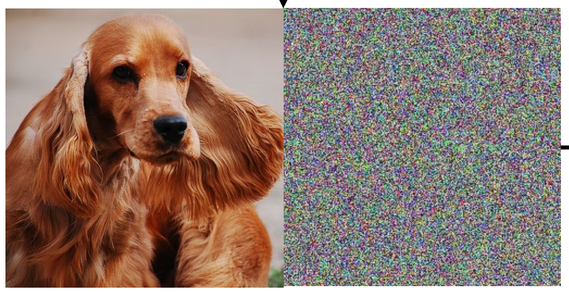
GENERATIVE ADVERSARIAL NETWORKS (GANS)

Generated Images

GENERATOR



maximize D(G(z))

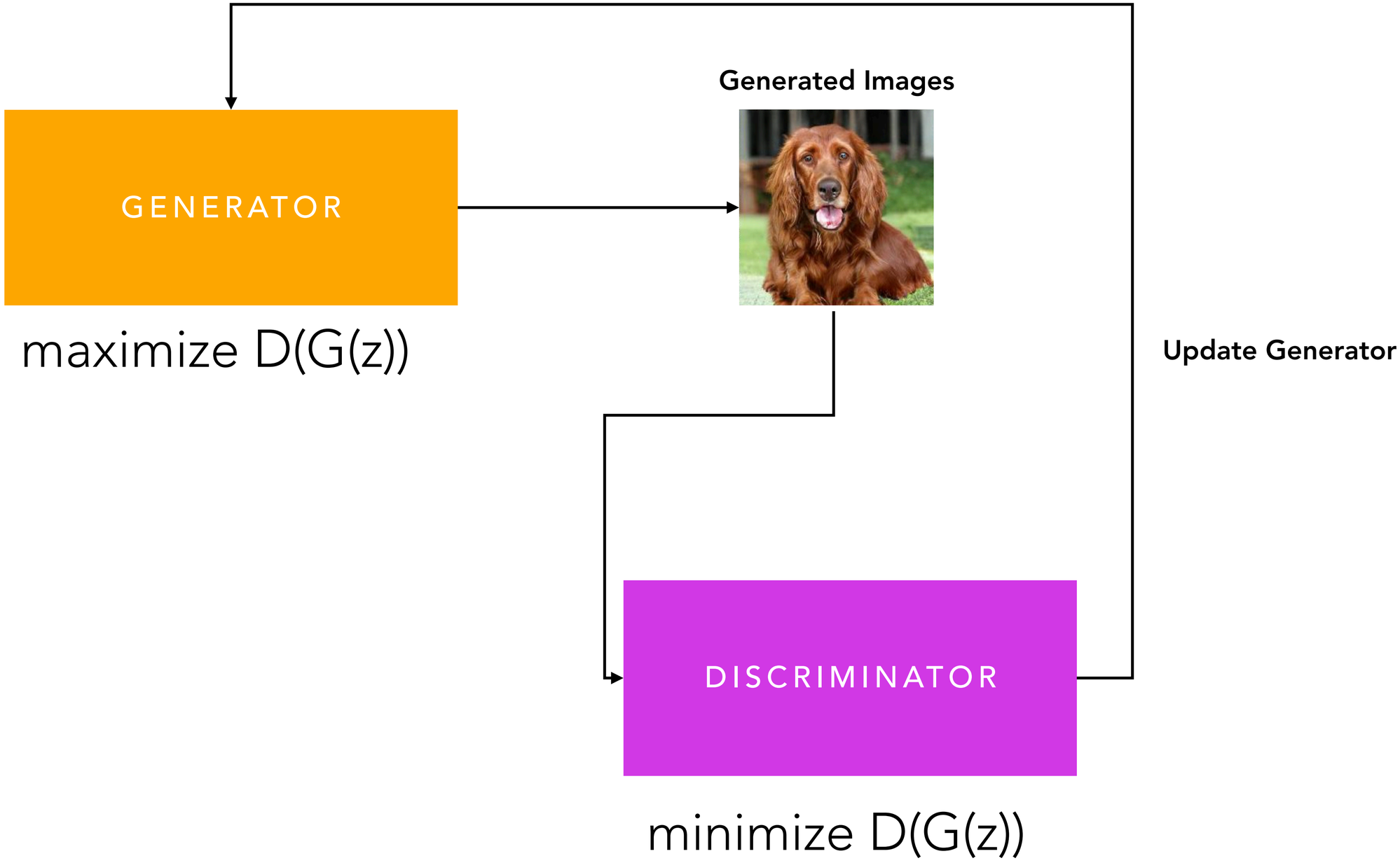


Real and Fake Images

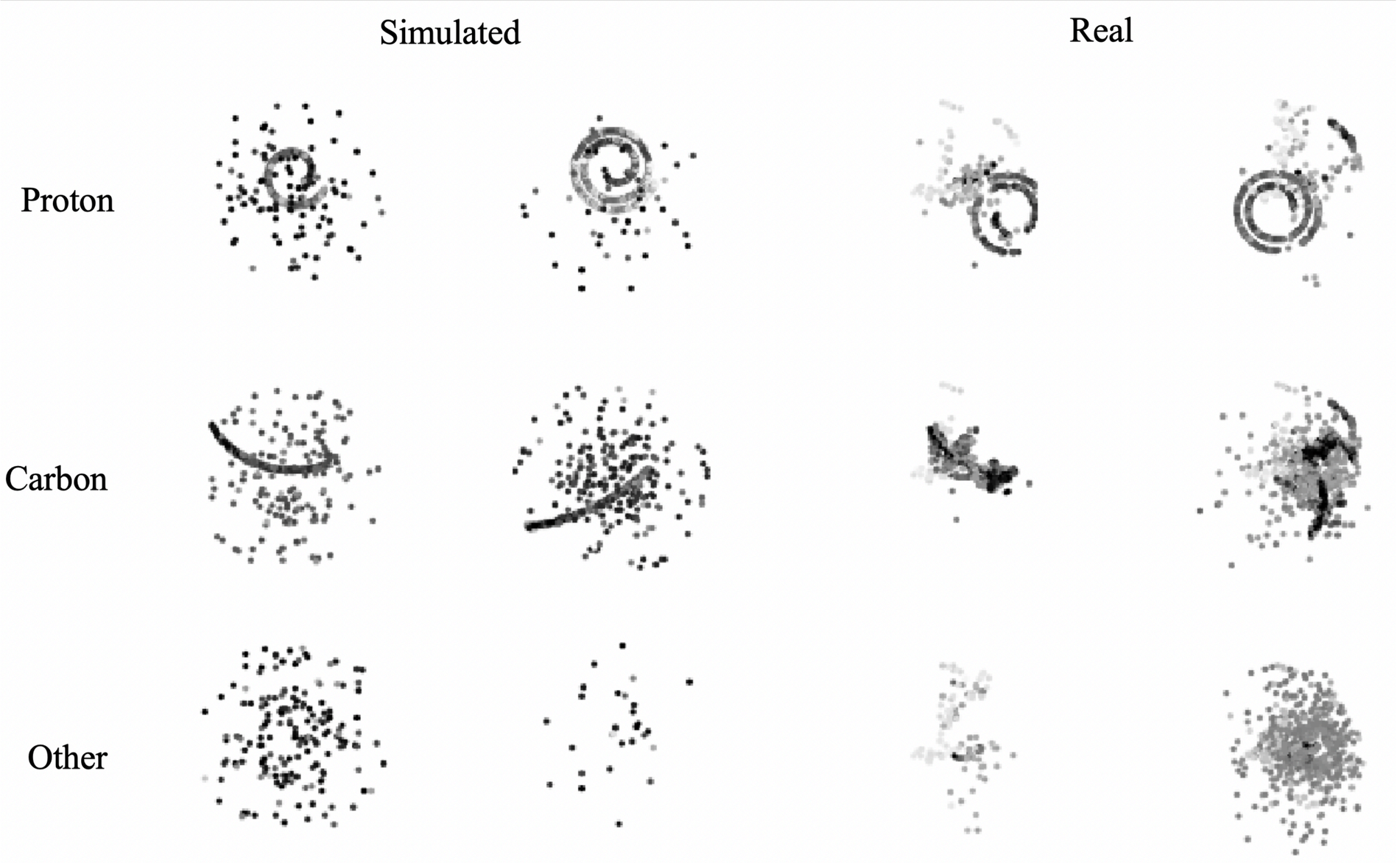
DISCRIMINATOR

minimize D(G(z))

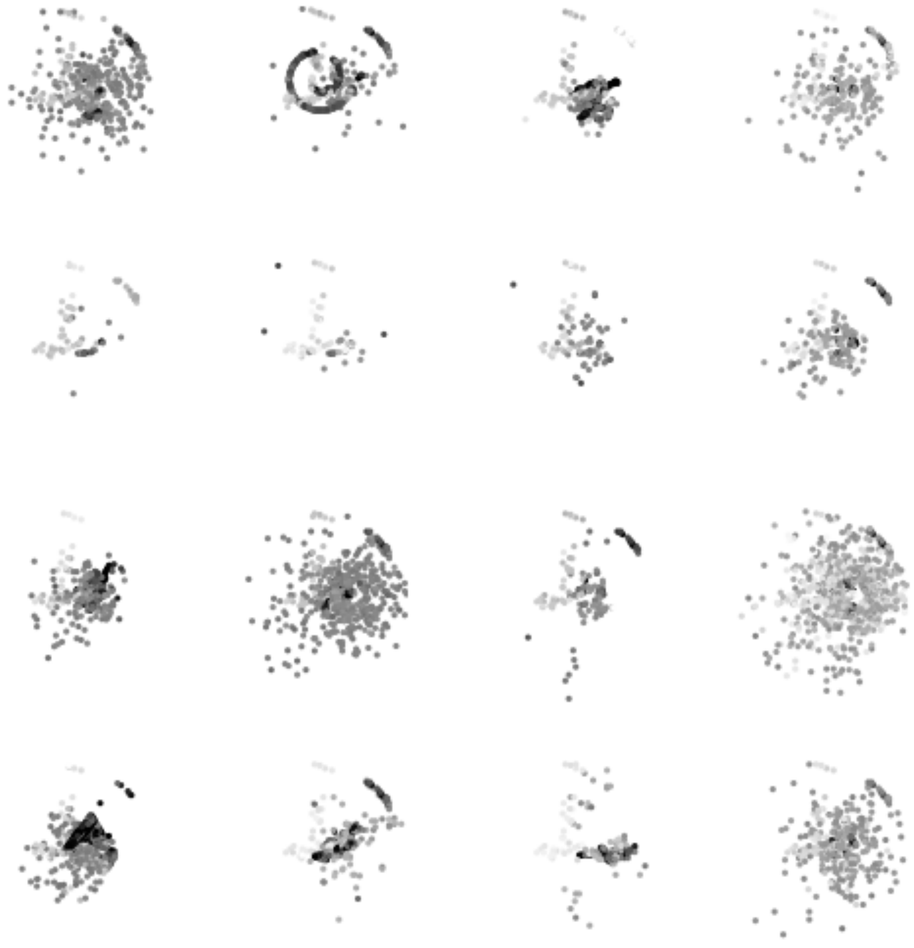
PHOTOS FROM KATRINA S; ANDREW B ET. AL., *BIGGAN*.



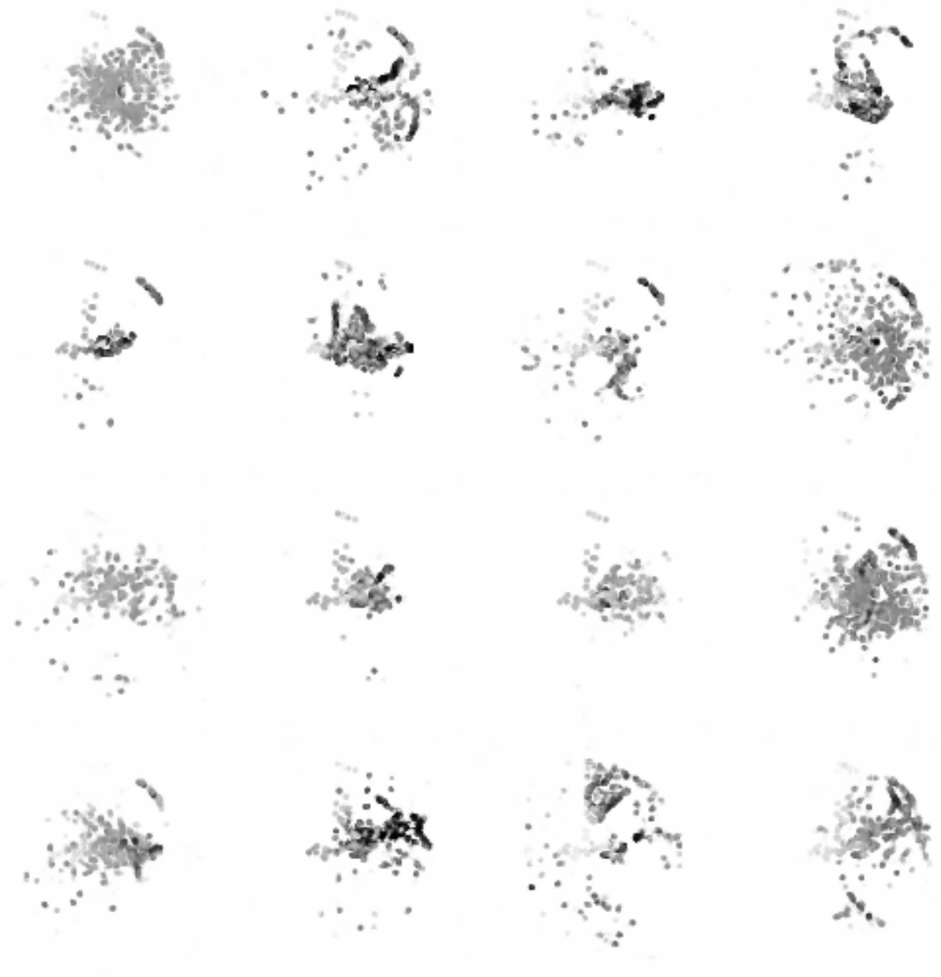
PHOTOS FROM KATRINA S; ANDREW B ET. AL., *BIGGAN*.


Simulated
Real
Proton
Carbon
Other

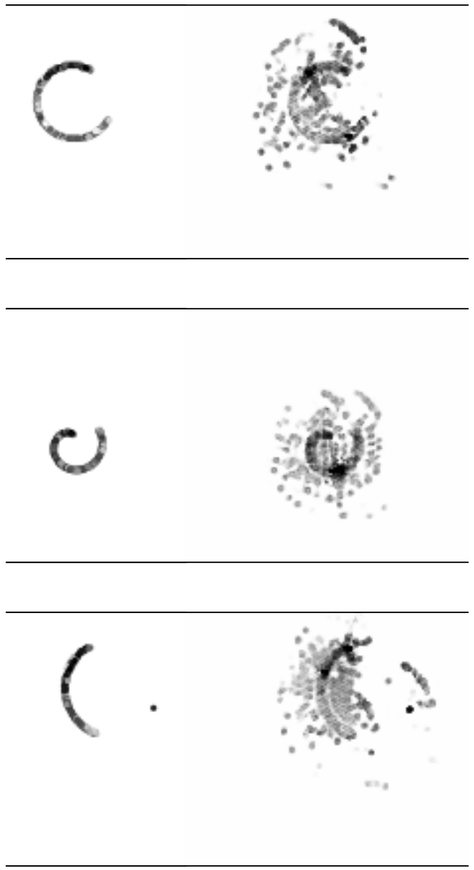
Real



Generated

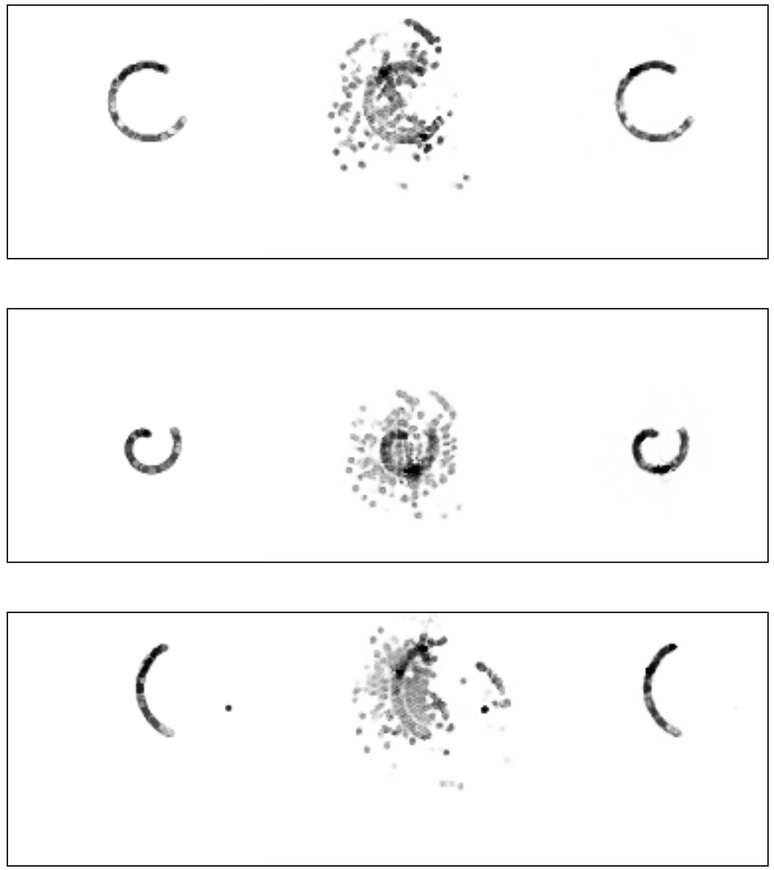


- **Goal:** learn to add realistic noise to a clean, simulated event
  - allow realistic simulation
  - transfer learn with higher accuracy



CYCLEGAN

- **CycleGAN:** translate images from one domain to another
- Can both clean real data AND generate noisy data!

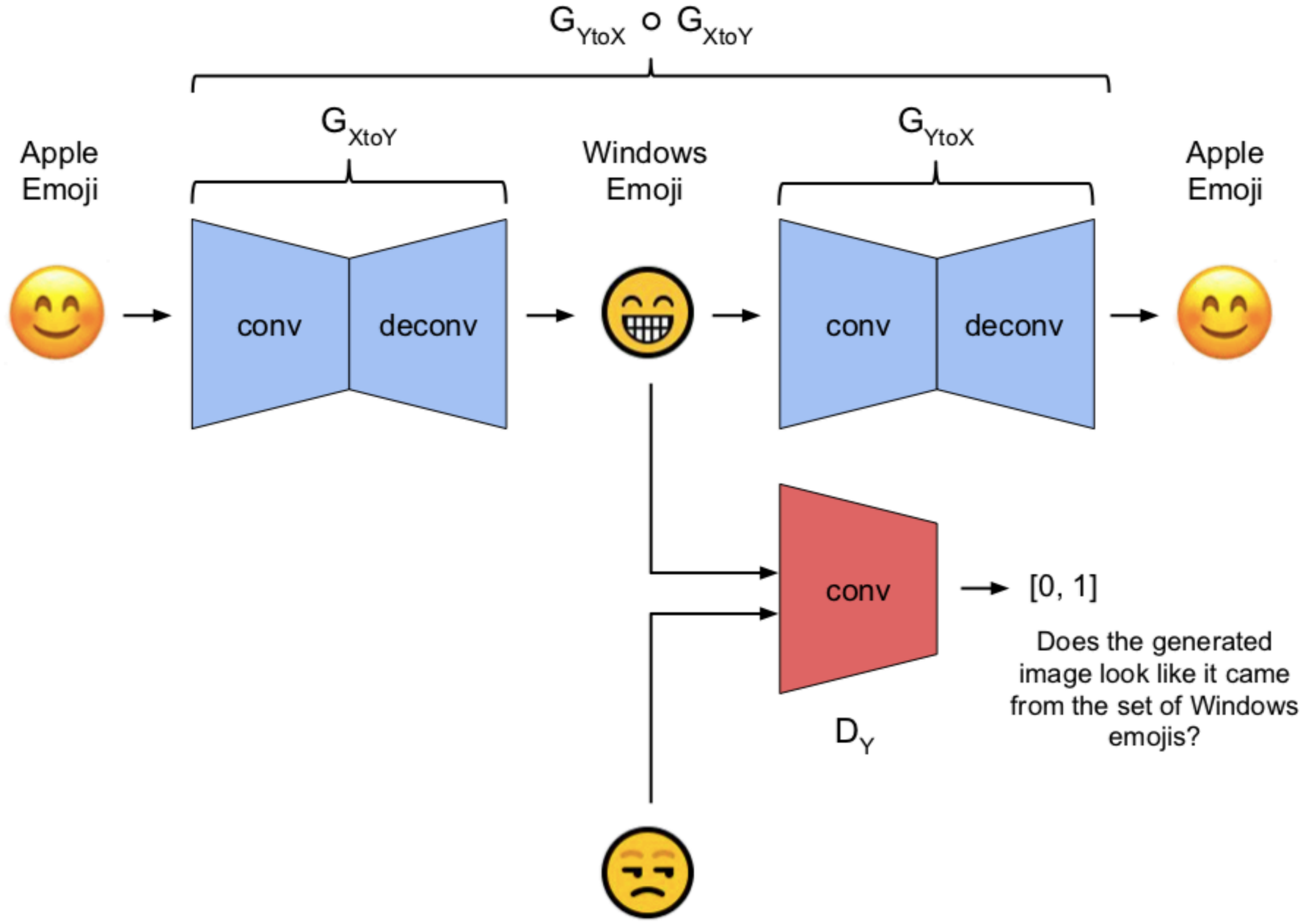


GAN Problems

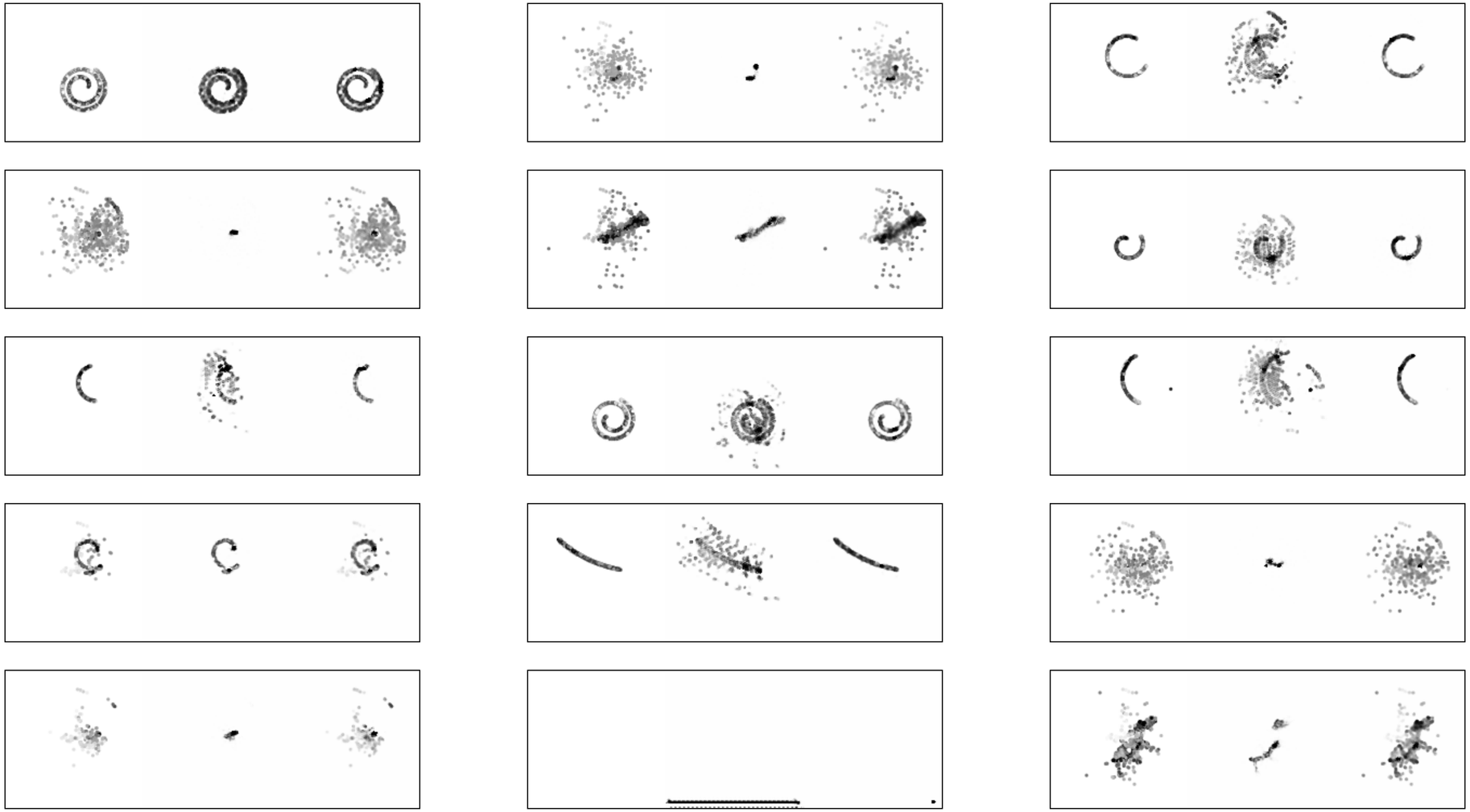
- **Vanishing gradients**
  - If the discriminator behaves badly, the generator does not have accurate feedback and the loss function cannot represent the reality.
  - If the discriminator does a great job, the gradient of the loss function drops too close to zero and the learning becomes super slow or even jammed.
- **Mode collapse**
  - During the training, the generator may collapse to a setting where it always produces same the outputs.
  - Even though the generator might be able to trick the corresponding discriminator, it fails to learn to represent the real-world data and gets stuck in a small space with extremely low variety.

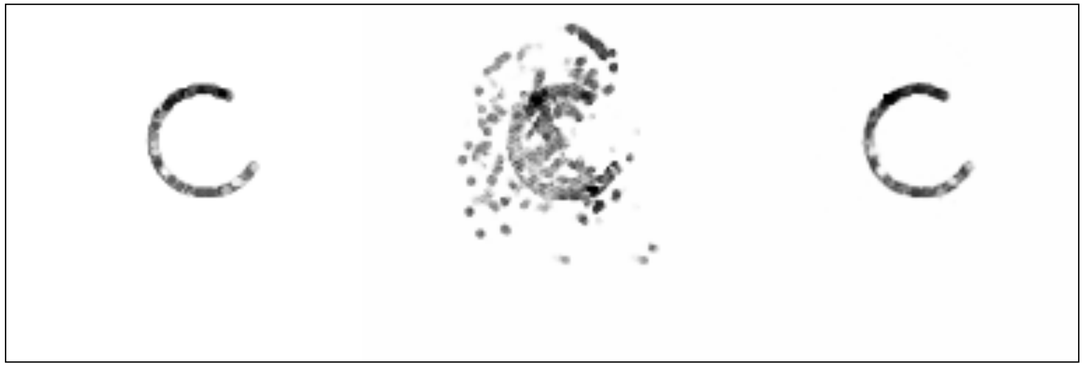
Heuristic Tricks

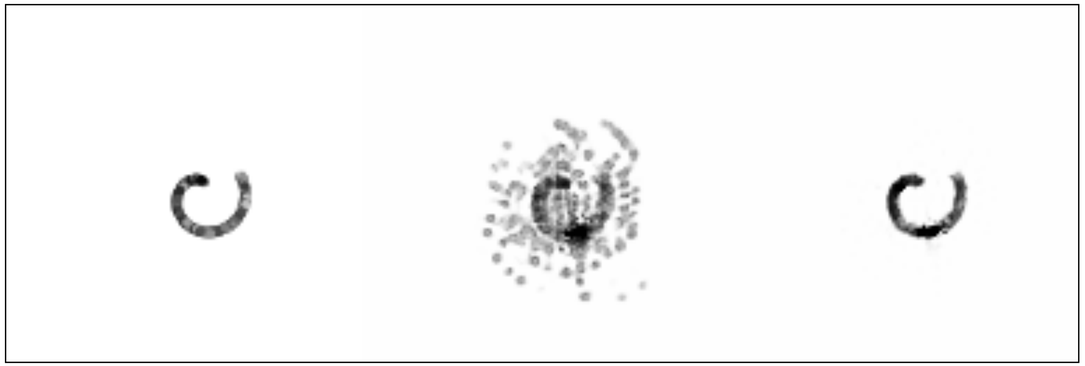
- Normalize the images between -1 and 1.
- Use `tanh` as the last activation in the generator, instead of `sigmoid`.
- Sample points from the latent space using a normal distribution, not a uniform distribution.
- Because GAN training results in a dynamic equilibrium, GANs are likely to get stuck in all sorts of ways. Introducing randomness during training helps prevent this.
  - Use dropout in the discriminator.
  - Add random noise to the labels for the discriminator.
  - Add gaussian noise to every layer of generator.
  - Use dropout in generator in both train and test phase.
- Avoid sparse gradients.
  - Instead of max pooling, use strided convolutions for downsampling.
  - Use a `LeakyReLU` layer instead of a `ReLU` activation (allows small negative values).
- Use a kernel size that's divisible by the stride size whenever using a strided `Conv2DTranspose` or `Conv2D` in both the generator and the discriminator.
- Use batch norm in both generator and discriminator.
- Remove fully-connected hidden layers for deeper architectures.
- Use SGD for discriminator and ADAM for generator.

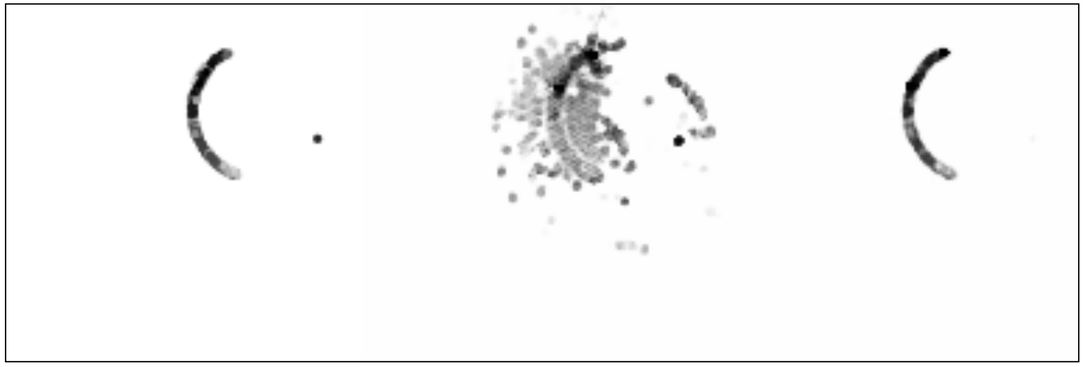

$G_{YtoX} \circ G_{XtoY}$
G_{XtoY}
G_{YtoX}
Apple Emoji
Windows Emoji
Apple Emoji
conv
deconv
conv
deconv
conv
[0, 1]
Does the generated image look like it came from the set of Windows emojis?
D_Y

Original -> Translated -> Reconstructed





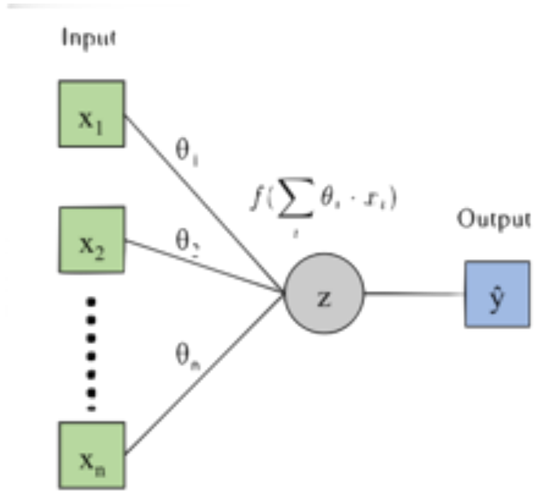




VALIDATION

- Are all generated data physical?
- Does charge distribution of generator match experimental data?
- Are classes “similar”?

Experimental learning task	Precision	Recall	F1
Logistic Regression	0.77	0.67	0.72
FC Neural Network	0.83	0.62	0.71



Transfer learning task	Precision	Recall	F1
Logistic Regression	0.44	0.03	0.05
FC Neural Network	0.78	0.44	0.57

CNNS

Experimental task: accuracy: 0.98	Precision	Recall	F1
Proton	0.97	0.88	0.93
Non-proton	0.96	0.99	0.98

Transfer task: accuracy: 0.95	Precision	Recall	F1
Proton	0.94	0.71	0.81
Non-proton	0.92	0.99	0.95