GENERATIVE ADVERSARIAL NETWORKS

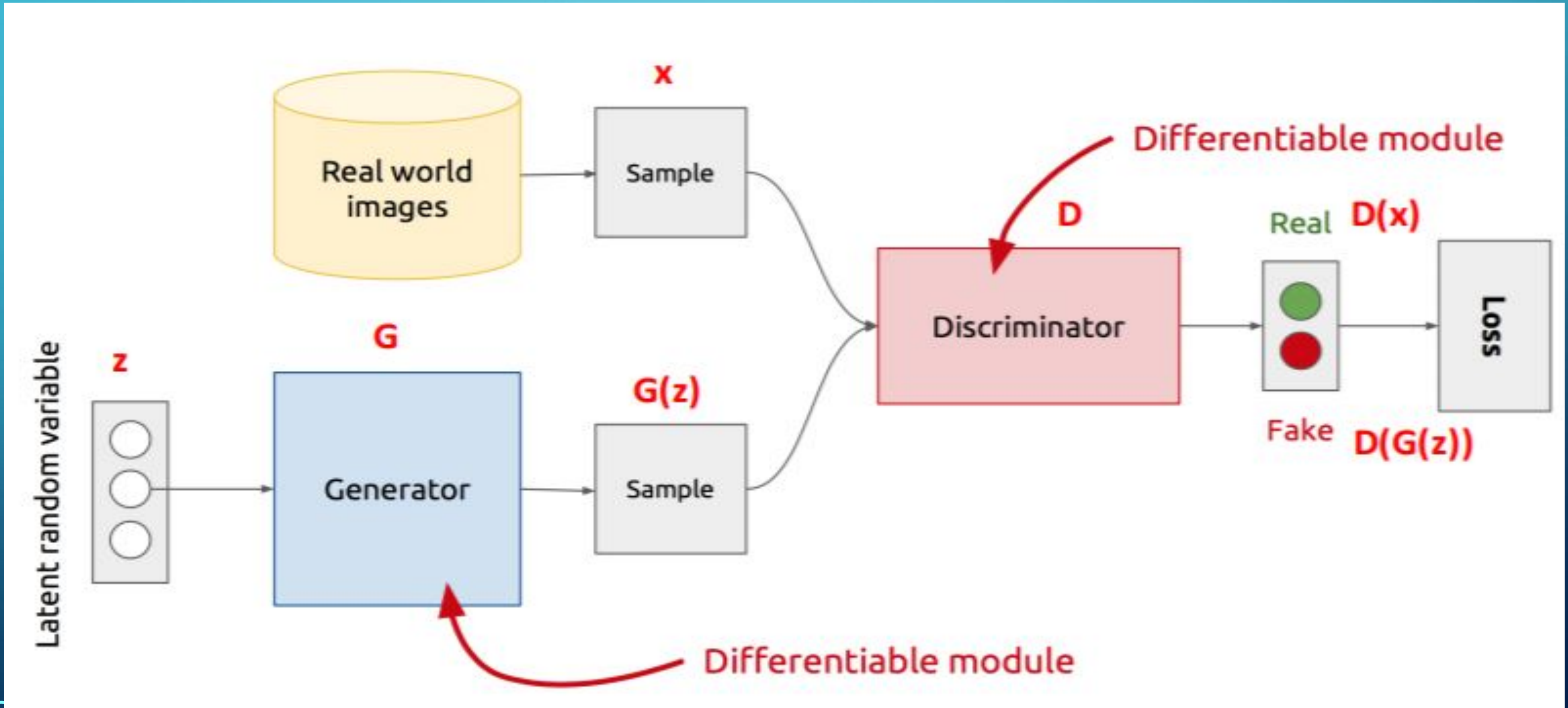
~PRANEETH V.

~ANKIT KR. GUPTA.

OBJECTIVES OF THE PROJECT.

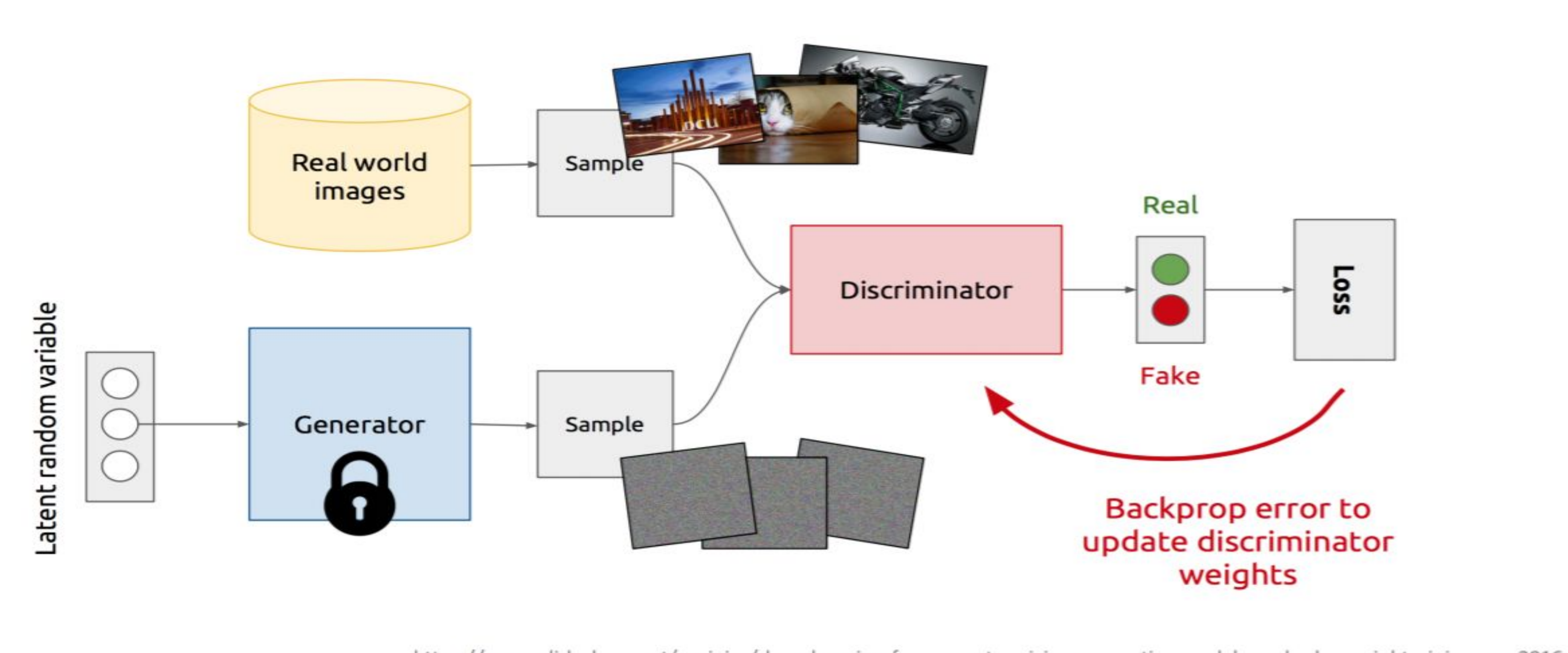
- Develop a GAN for sampling points from a function.
- Develop a GAN to generate images from MNIST handwritten digit dataset.

ARCHITECTURE OF GANS



	Discriminator	Generator
Architecture	Binary Classifier	Multi Layer Perceptron
Input	Real and fake samples	Random noise
Output	Probability that a sample is real	Fake samples

TRAINING THE DISCRIMINATOR

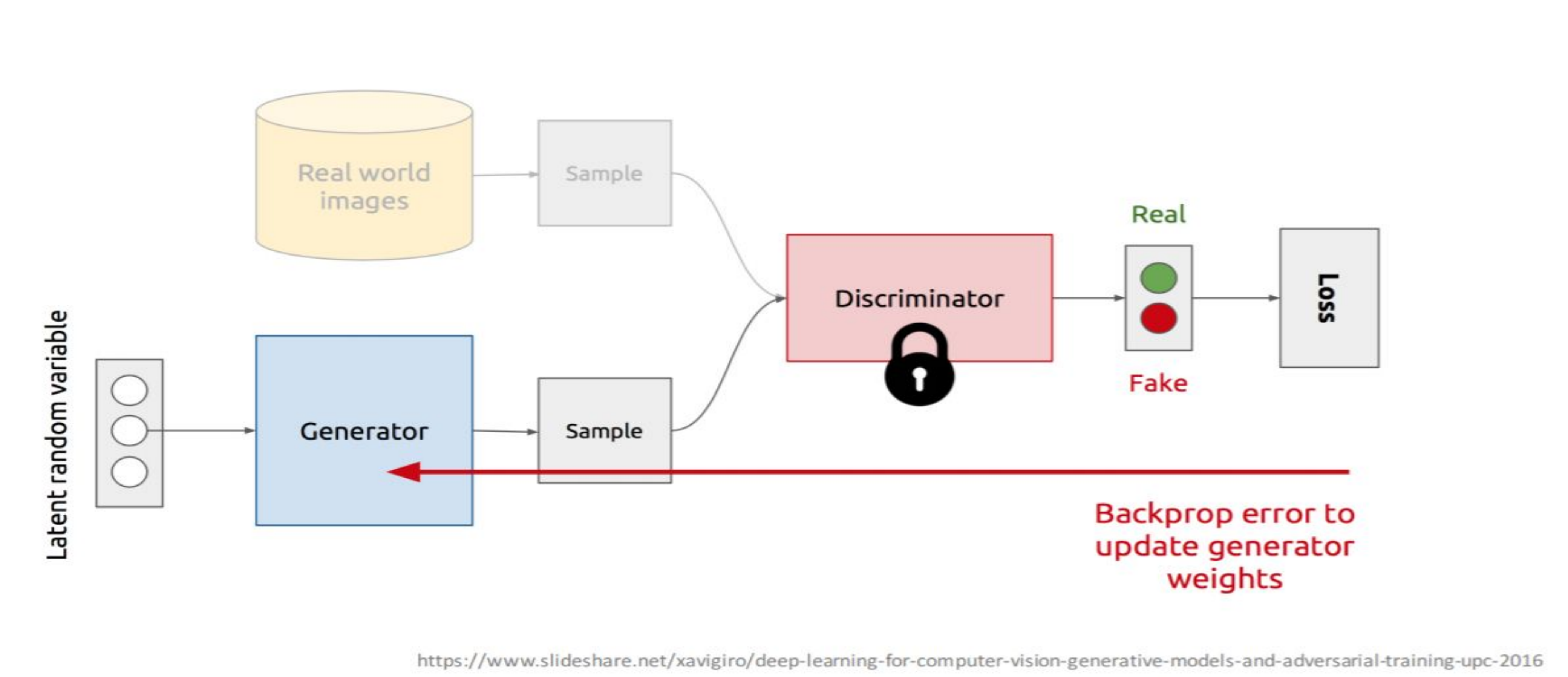


https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

COST FUNCTION-

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$

TRAINING THE GENERATOR



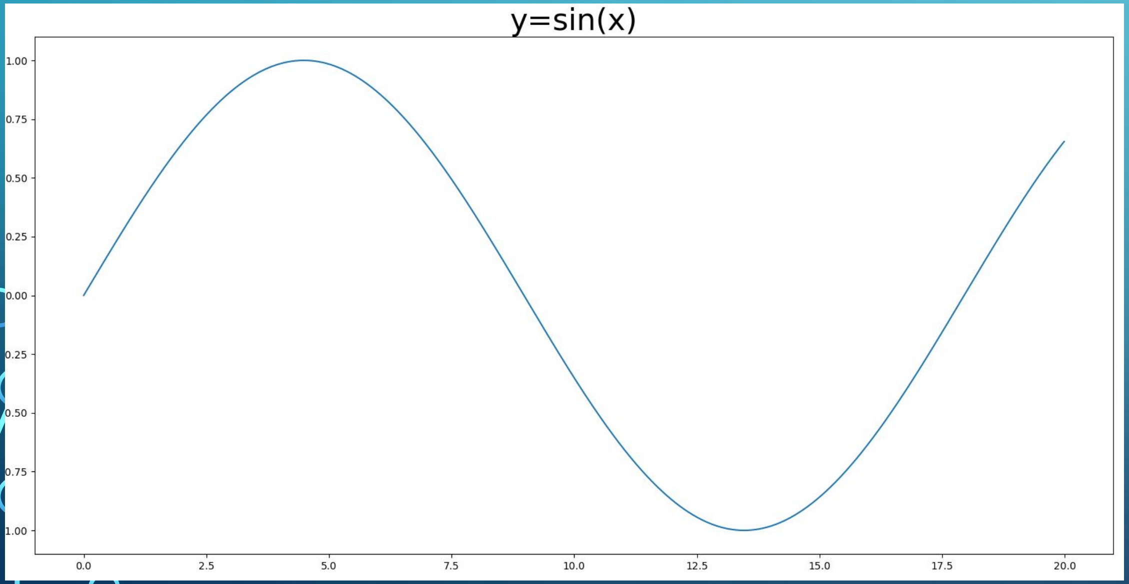
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

COST FUNCTION-

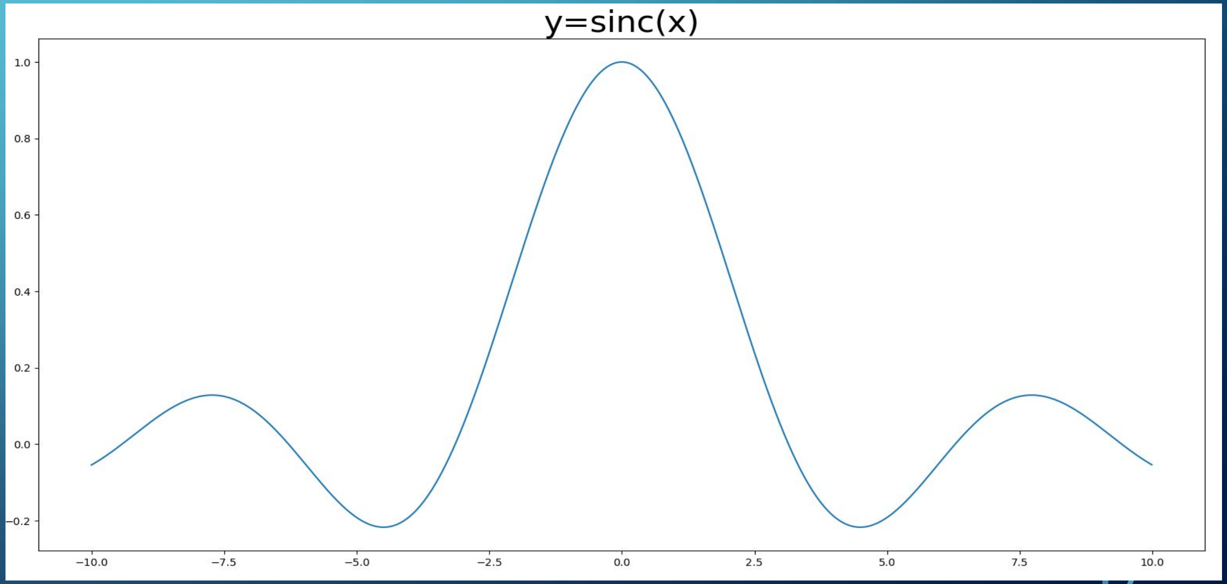
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$$

DEVELOPING A UNIVARIATE GAN

- We experimented with 2 functions y=sin(x) and y=sinc(x) to see whether GAN is able to generate data from these functions.
- **Plot of the functions used :**

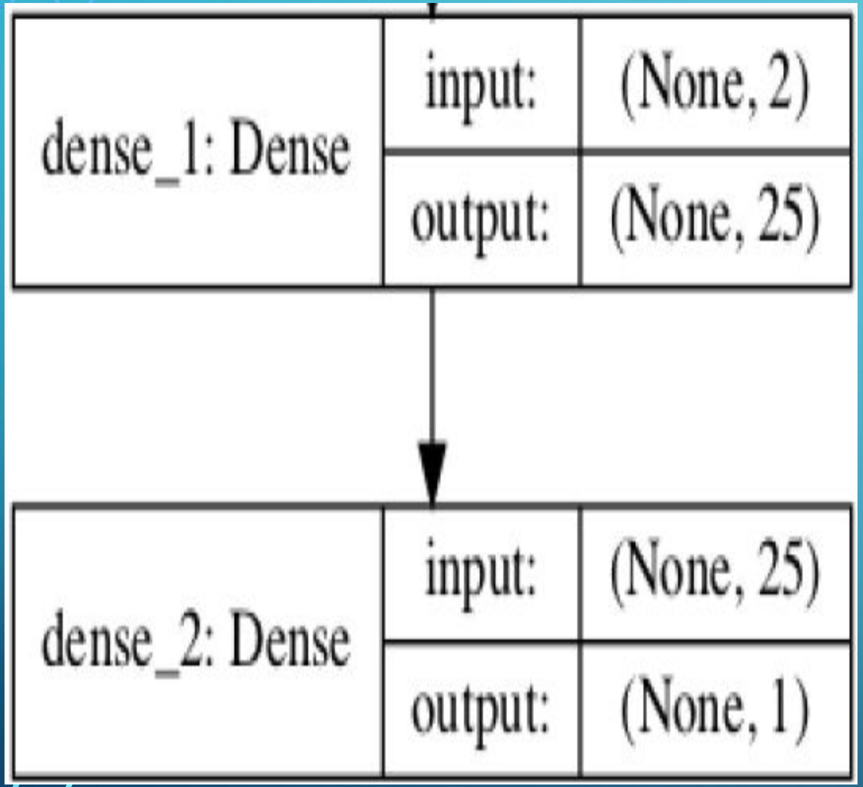


y=sin(x)



y=sinc(x)

DISCRIMINATOR MODEL



dense_1: Dense	input:	(None, 2)
	output:	(None, 25)

dense_2: Dense	input:	(None, 25)
	output:	(None, 1)

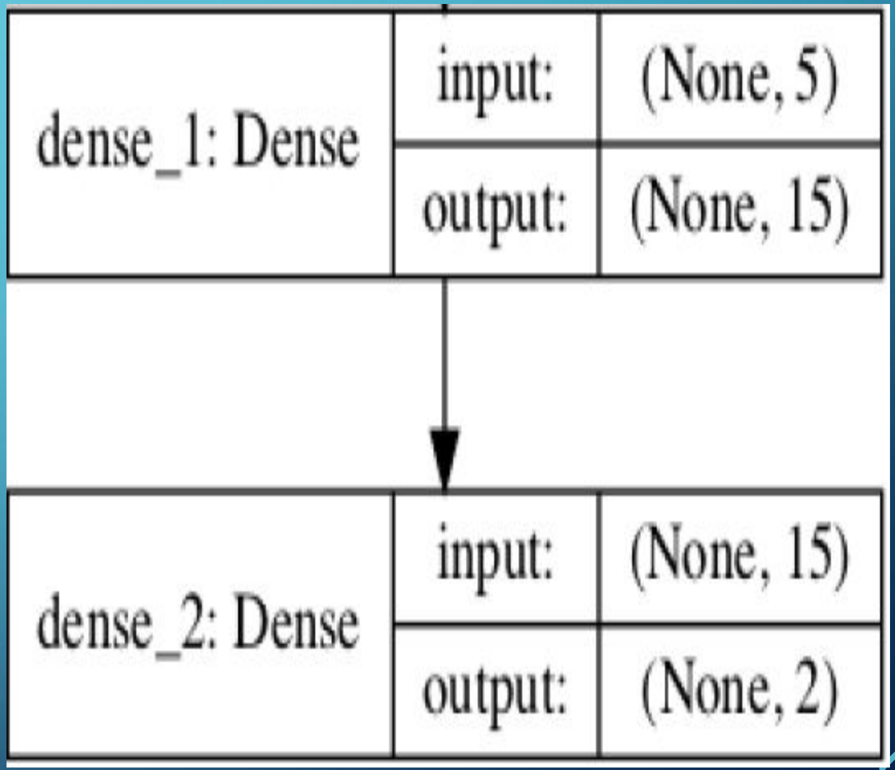
<--- input layer--->

<- hidden layer1->

<- hidden layer1->

<-- output layer-->

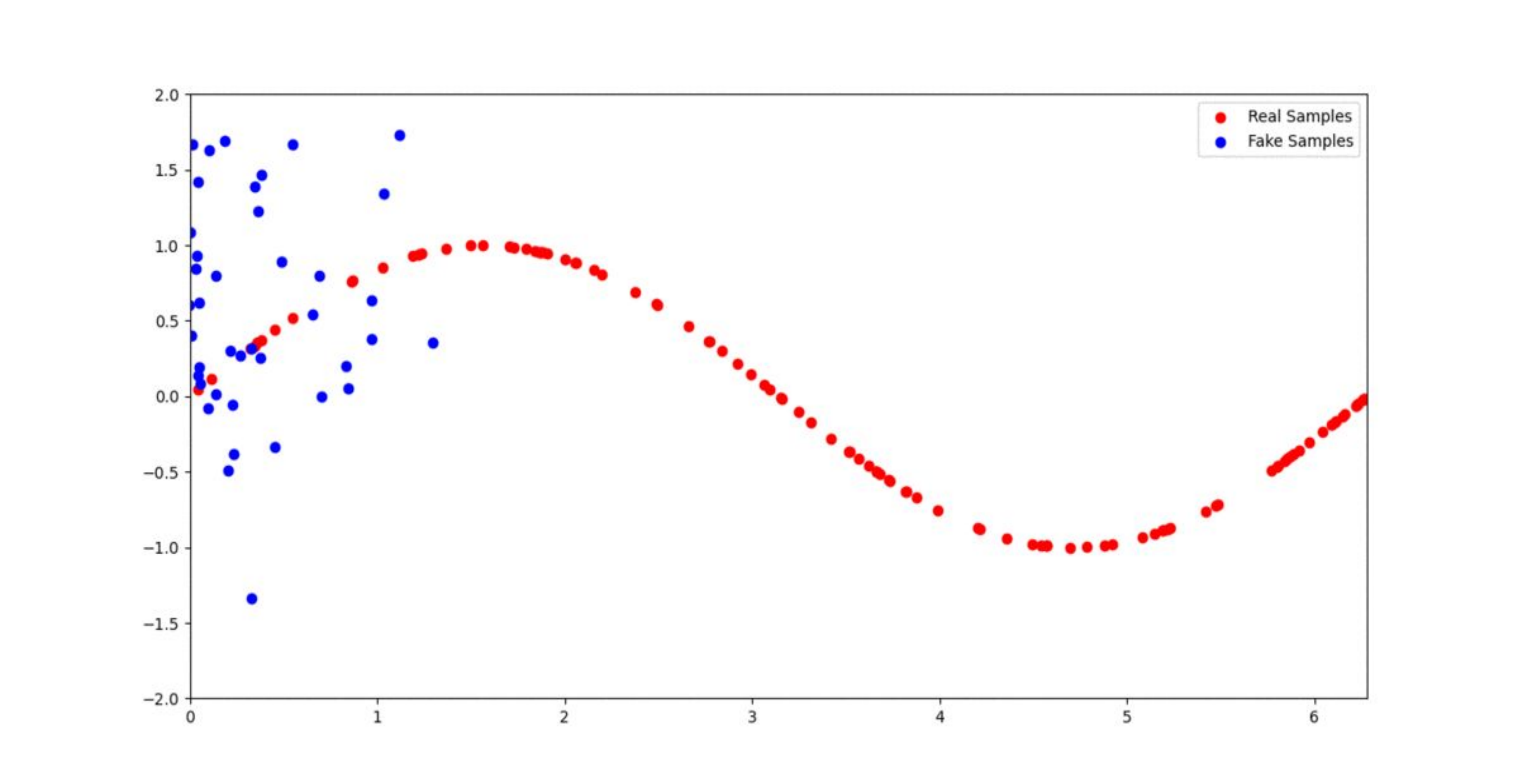
GENERATOR MODEL



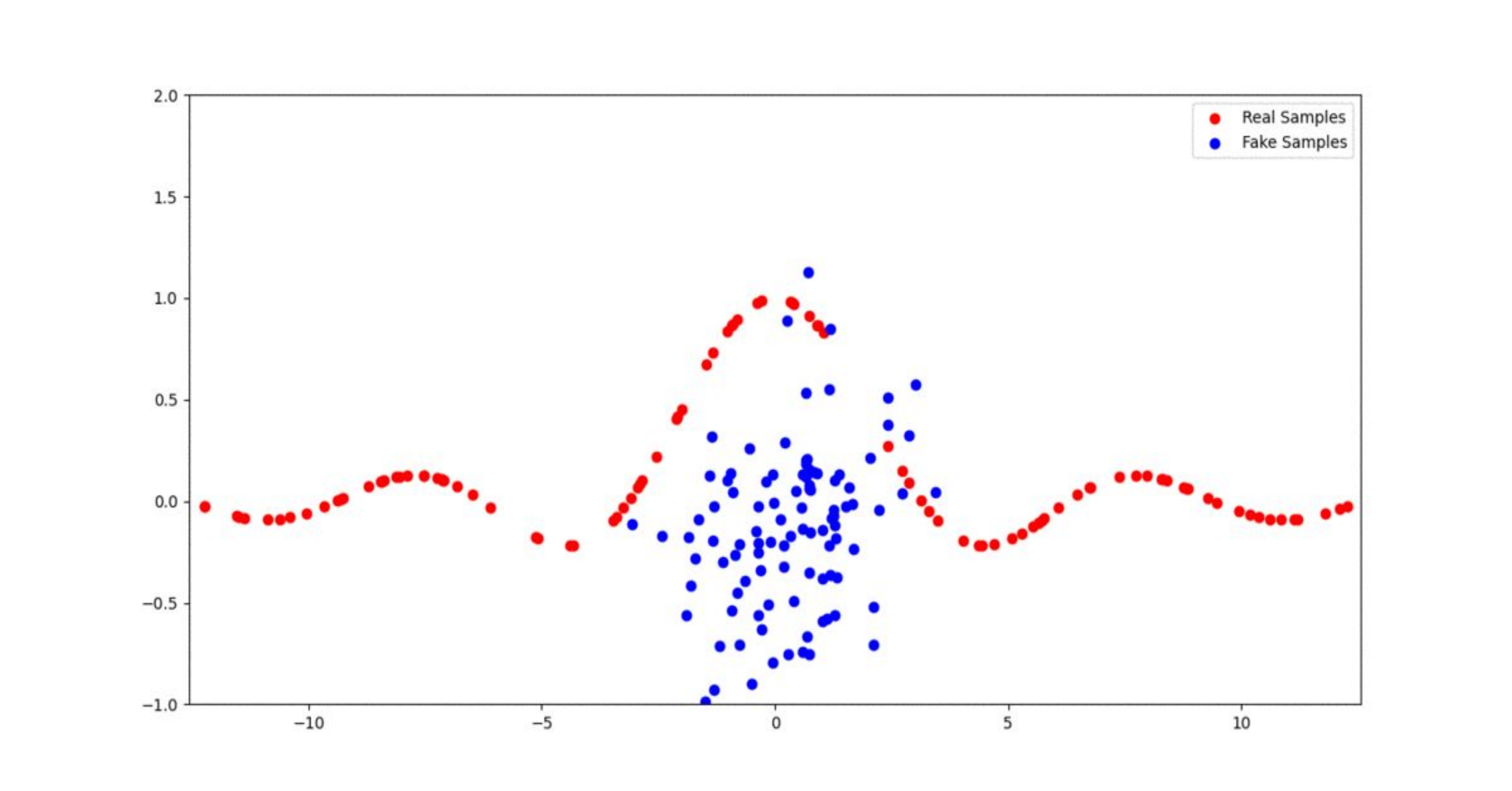
dense_1: Dense	input:	(None, 5)
	output:	(None, 15)

dense_2: Dense	input:	(None, 15)
	output:	(None, 2)

Training Process for Sin Function.

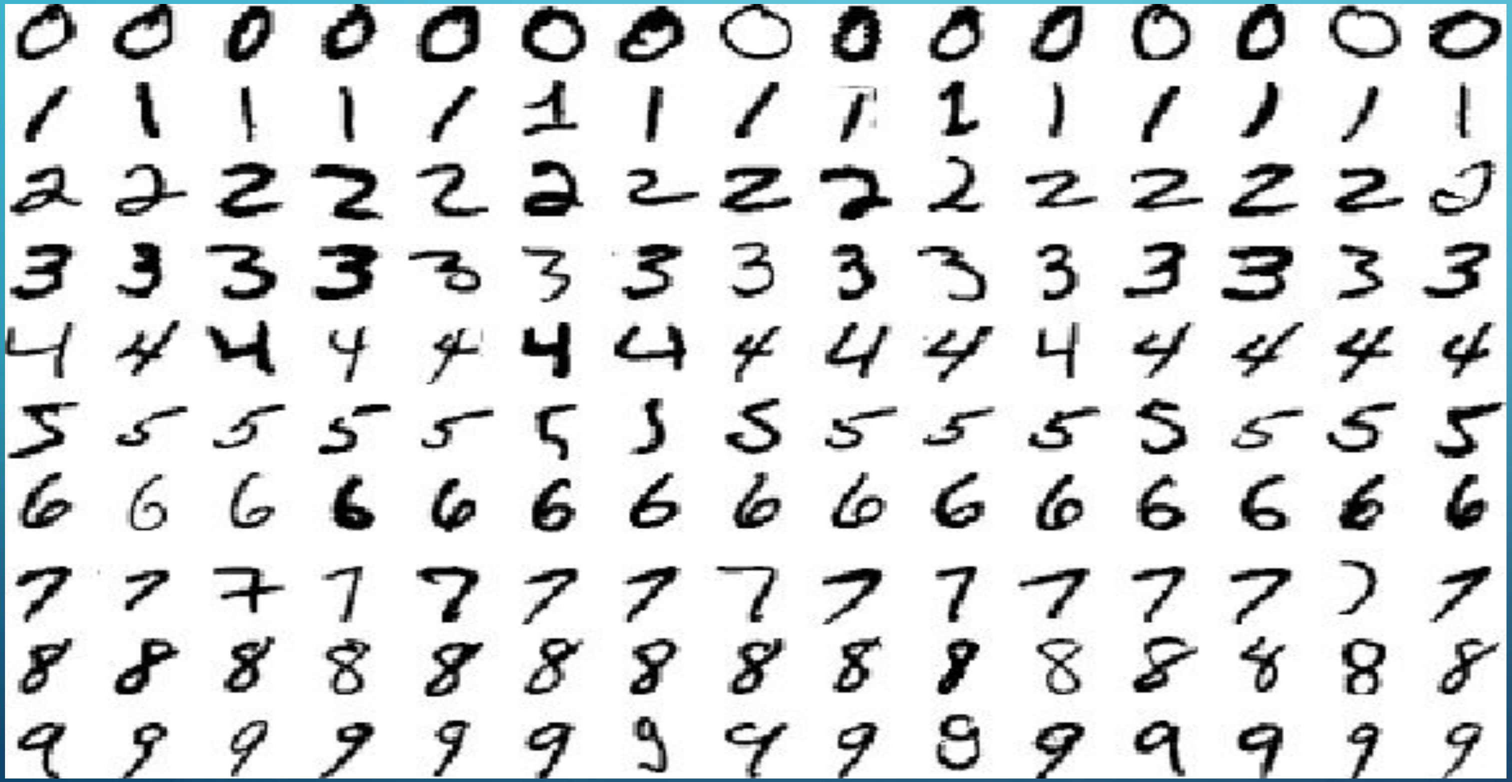


Training Process for Sinc Function.

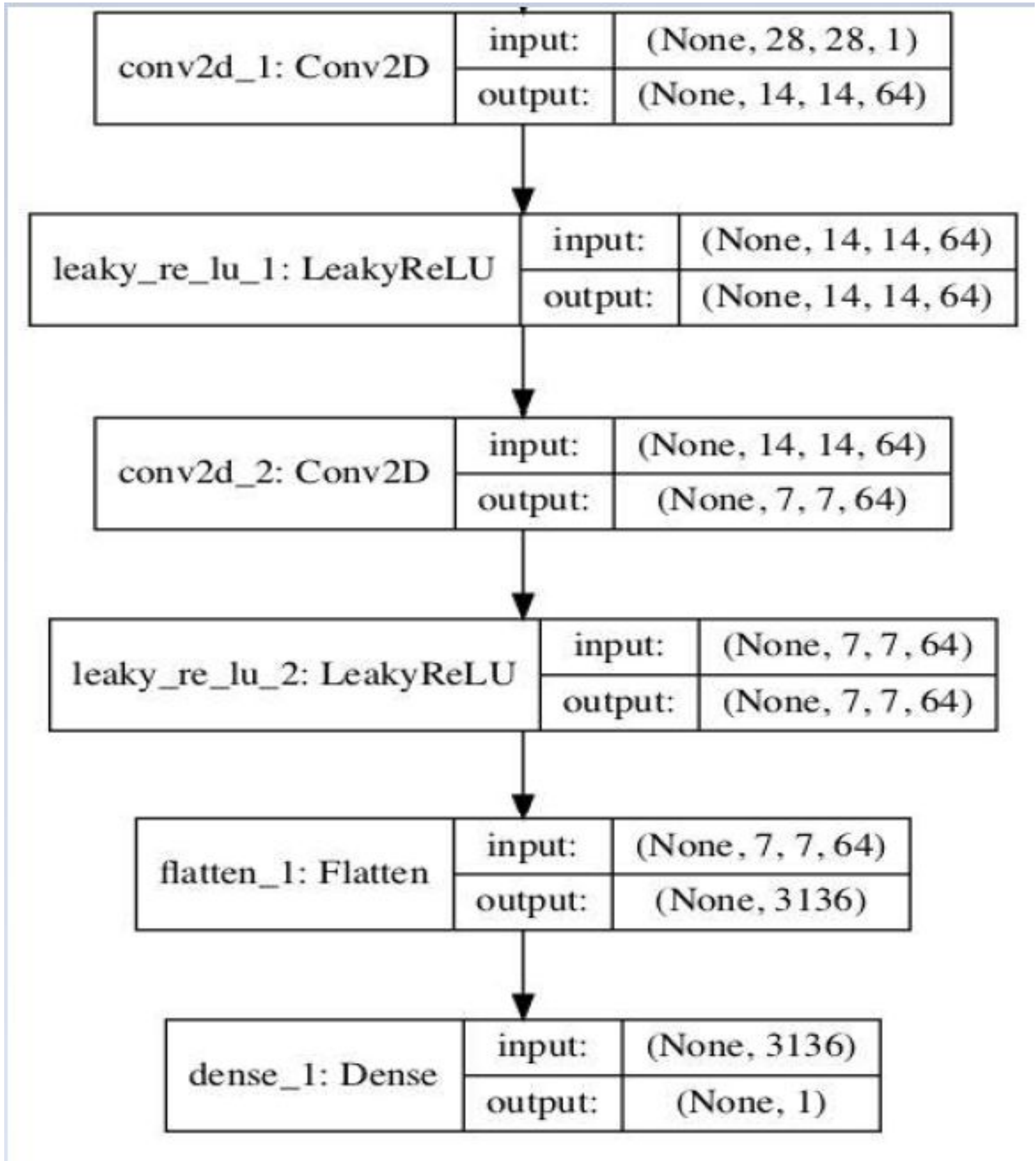


MNIST HANDWRITTEN DIGIT DATASET

- Contains handwritten single digits between 0 and 9 of size (28 x 28).



DISCRIMINATOR MODEL



conv2d_1: Conv2D | input: | (None, 28, 28, 1)
output: | (None, 14, 14, 64)

leaky_re_lu_1: LeakyReLU | input: | (None, 14, 14, 64)
output: | (None, 14, 14, 64)

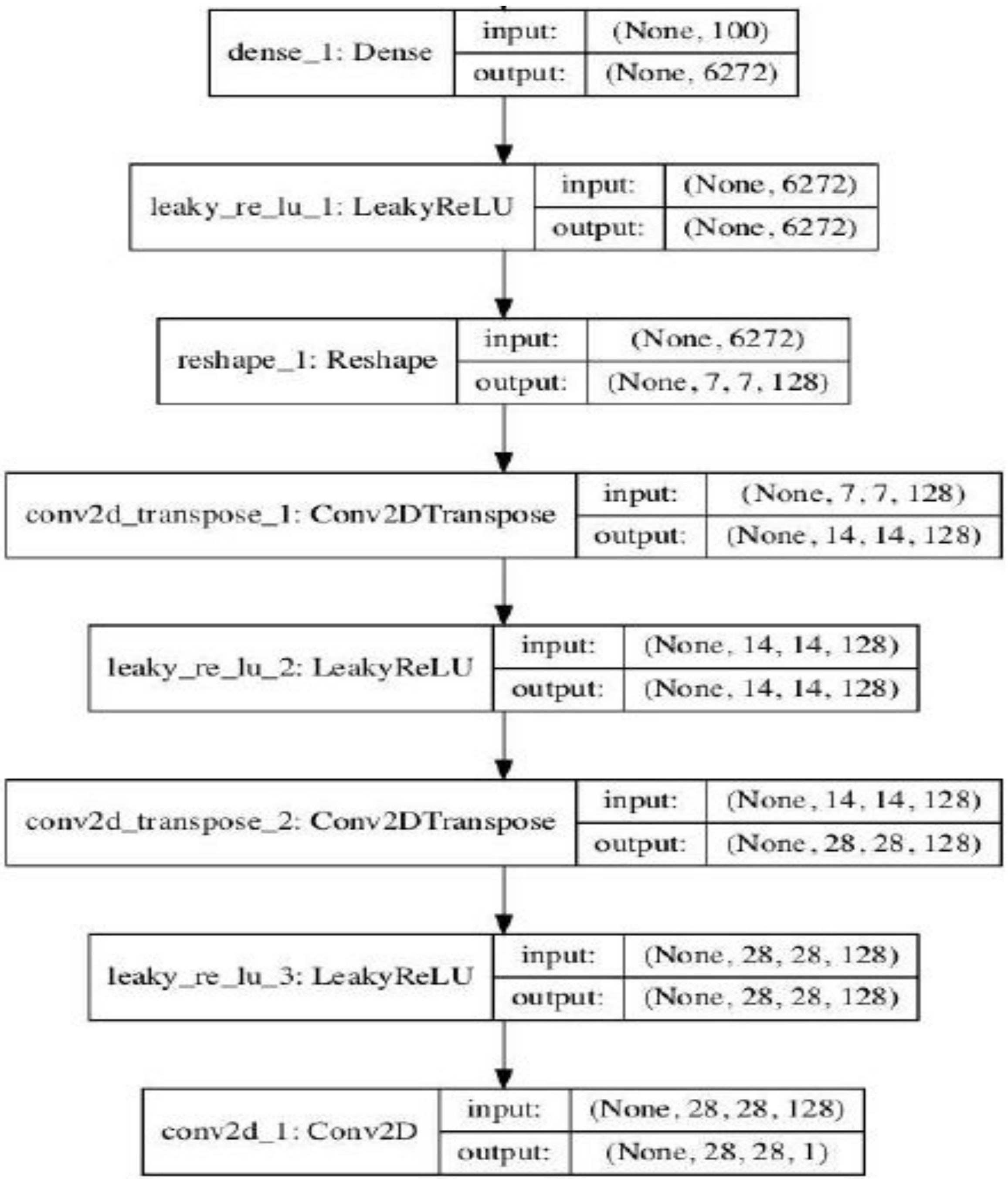
conv2d_2: Conv2D | input: | (None, 14, 14, 64)
output: | (None, 7, 7, 64)

leaky_re_lu_2: LeakyReLU | input: | (None, 7, 7, 64)
output: | (None, 7, 7, 64)

flatten_1: Flatten | input: | (None, 7, 7, 64)
output: | (None, 3136)

dense_1: Dense | input: | (None, 3136)
output: | (None, 1)

GENERATOR MODEL



dense_1: Dense | input: (None, 100) | output: (None, 6272)

leaky_re_lu_1: LeakyReLU | input: (None, 6272) | output: (None, 6272)

reshape_1: Reshape | input: (None, 6272) | output: (None, 7, 7, 128)

conv2d_transpose_1: Conv2DTranspose | input: (None, 7, 7, 128) | output: (None, 14, 14, 128)

leaky_re_lu_2: LeakyReLU | input: (None, 14, 14, 128) | output: (None, 14, 14, 128)

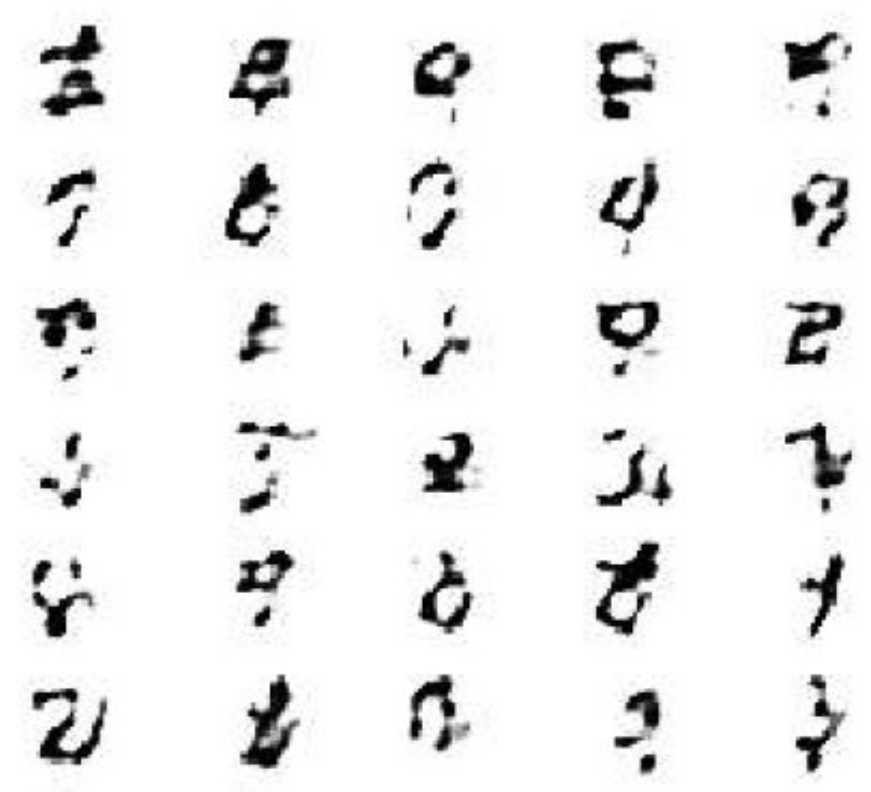
conv2d_transpose_2: Conv2DTranspose | input: (None, 14, 14, 128) | output: (None, 28, 28, 128)

leaky_re_lu_3: LeakyReLU | input: (None, 28, 28, 128) | output: (None, 28, 28, 128)

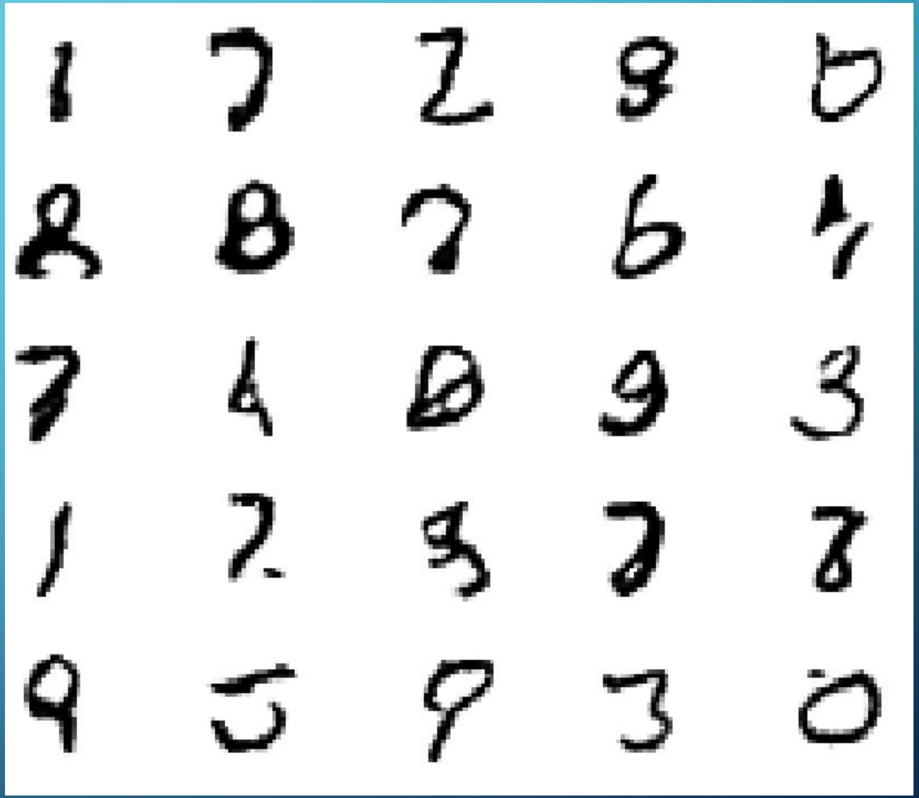
conv2d_1: Conv2D | input: (None, 28, 28, 128) | output: (None, 28, 28, 1)

OUTPUT OF GAN

Result after 10 epochs.



Result after 100 epochs



References

[1] . Ian J Goodfellow et.al. Generative Adversarial Nets.