Practical Session

Generative Adversarial Networks (GANs)

Applied Artificial Intelligence

David Anghelone - Team STARS

Train DCGAN for generating Manga Faces !

DC GAN : Deep Convolutional Generative Adversarial Network

DATASET
Generator
Discriminator
100 z
Project and reshape
1024
4
4
512
8
8
5
5
Stride 2
CONV 1
256
16
16
Stride 2
5
5
CONV 2
128
32
32
5
5
Stride 2
CONV 3
64
64
5
5
Stride 2
CONV 4
3
G(z)
P(x)

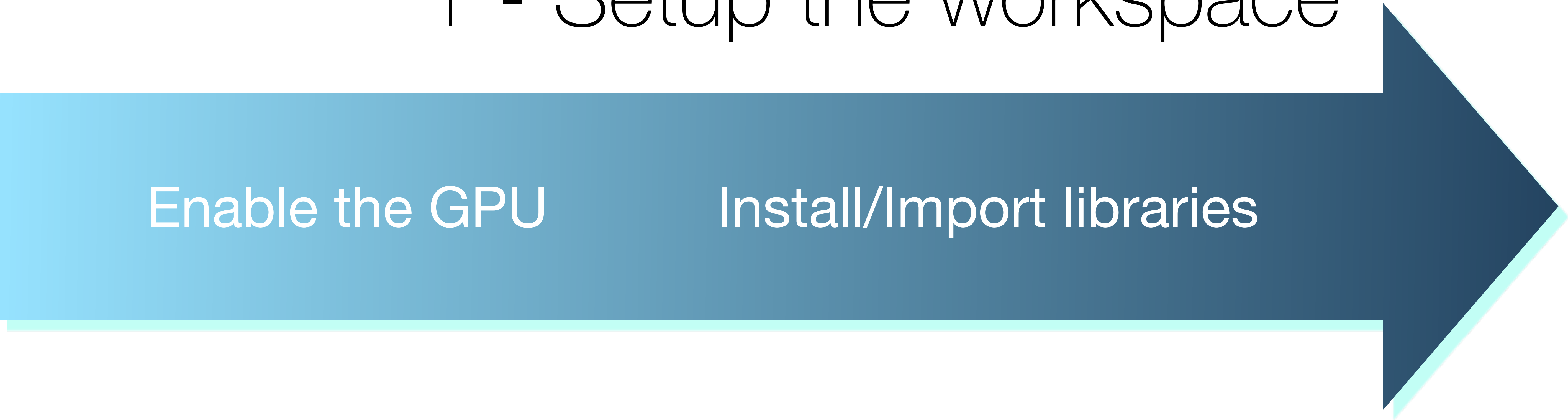
The **Generator** network is able to take random noise and map it into images, such that the **Discriminator** network cannot tell which images came from the *dataset* and which images came from the *generator*.

Roadline. *A step-by-step instruction*

Google colab

GETTING STARTED WITH GOOGLE COLAB

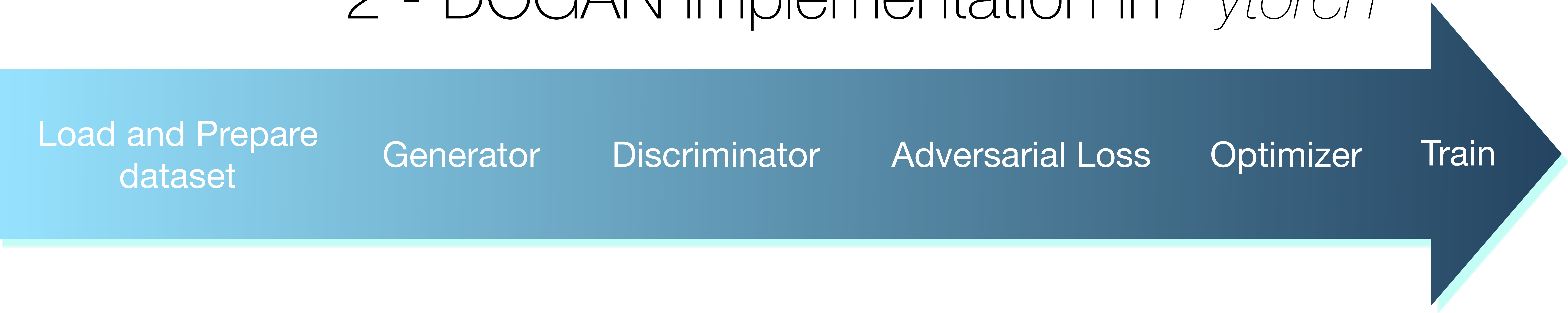
1 - Setup the workspace



Enable the GPU

Install/Import libraries

2 - DCGAN implementation in *Pytorch*



Load and Prepare dataset

Generator

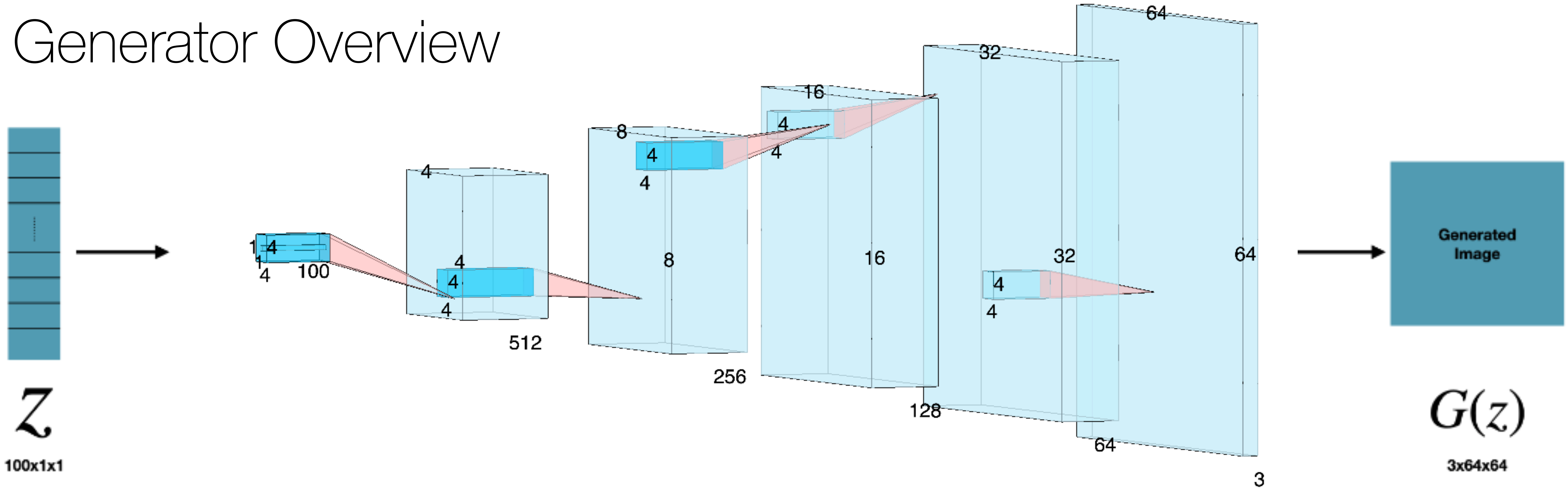
Discriminator

Adversarial Loss

Optimizer

Train

Generator Overview



1st layer:	2nd layer:	3rd layer:	4th layer:	5th layer:
input: **100x1x1**	input: 512x4x4	input: 256x8x8	input: 128x16x16	input: 64x32x32
ConvTransp. batch norm relu	ConvTransp. batch norm relu	ConvTransp. batch norm relu	ConvTransp. batch norm relu	ConvTransp. tanh
output: 512x4x4	output: 256x8x8	output: 128x16x16	output: 64x32x32	output: **3x64x64**

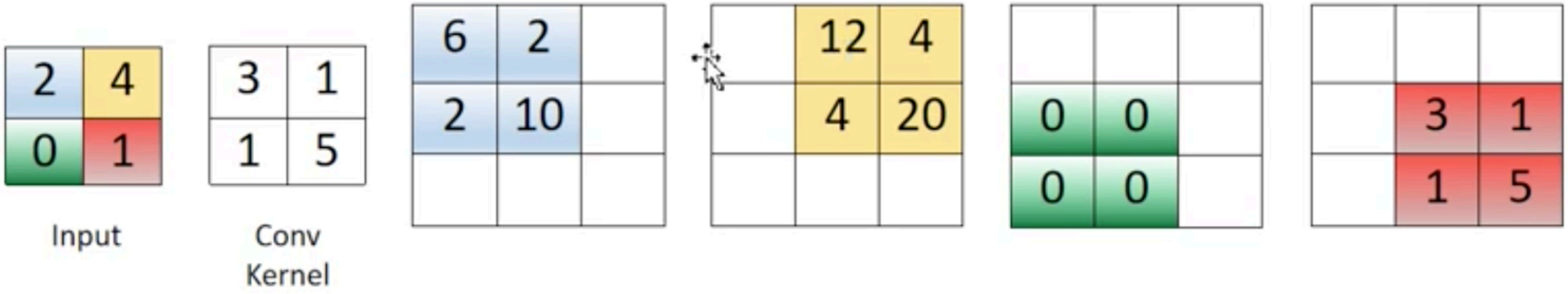
Transposed Convolutions

2x2 convolution, stride of 1 and a pad of 0

4 * 3 = 12
2 * 1 = 2
12 + 2 = 14

6	14	4
2	17	21
0	1	5

Output



2	4
0	1

Input

3	1
1	5

Conv Kernel

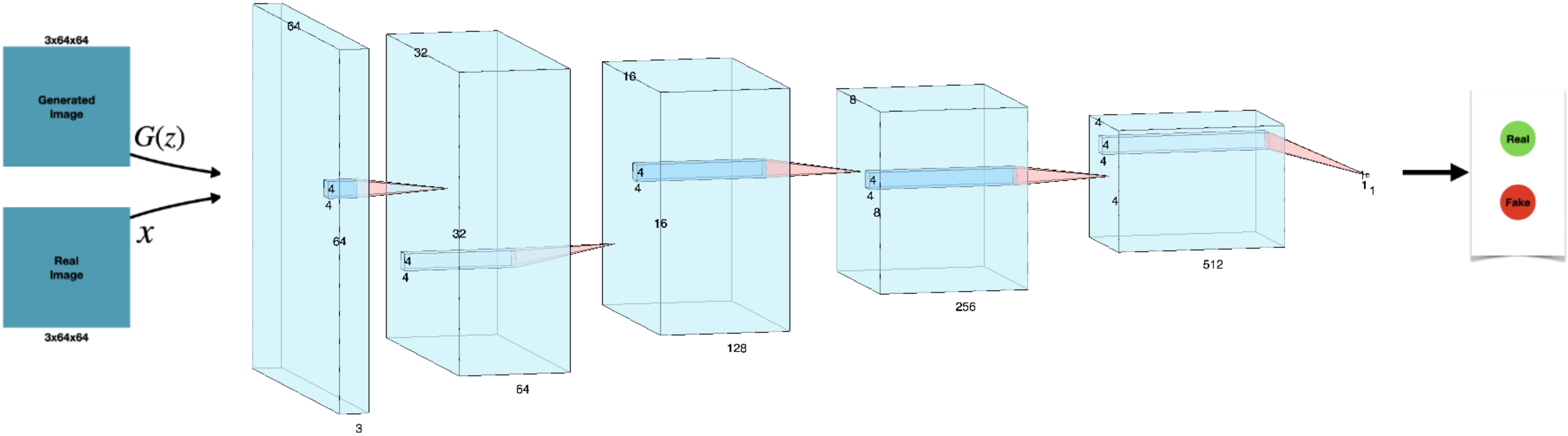
6	2	
2	10	

	12	4
	4	20

0	0	
0	0	

	3	1
	1	5

Discriminator Overview



1st layer:	2nd layer:	3rd layer:	4th layer:	5th layer:
input: **3x64x64**	input: 64x32x32	input: 128x16x16	input: 256x8x8	input: 512x4x4
Convolution Leakyrelu	Convolution batch norm Leakyrelu	Convolution batch norm Leakyrelu	Convolution batch norm Leakyrelu	Convolution sigmoid
output: 64x32x32	output: 128x16x16	output: 256x8x8	output: 512x4x4	output: **1**

Defining the Losses

Since this is a binary classification problem, the ultimate loss function would be ***Binary Cross Entropy***.

- > However, we will see during the training that this loss is adjusted and applied to both the networks separately in order to optimize their objective.

Discriminator

Wants itself to predict generated outputs as fake, and at the same time, it must predict any real image as real.

Hence, the discriminator trains on a combination of these two following losses :

errD = errD_Real + errD_Fake

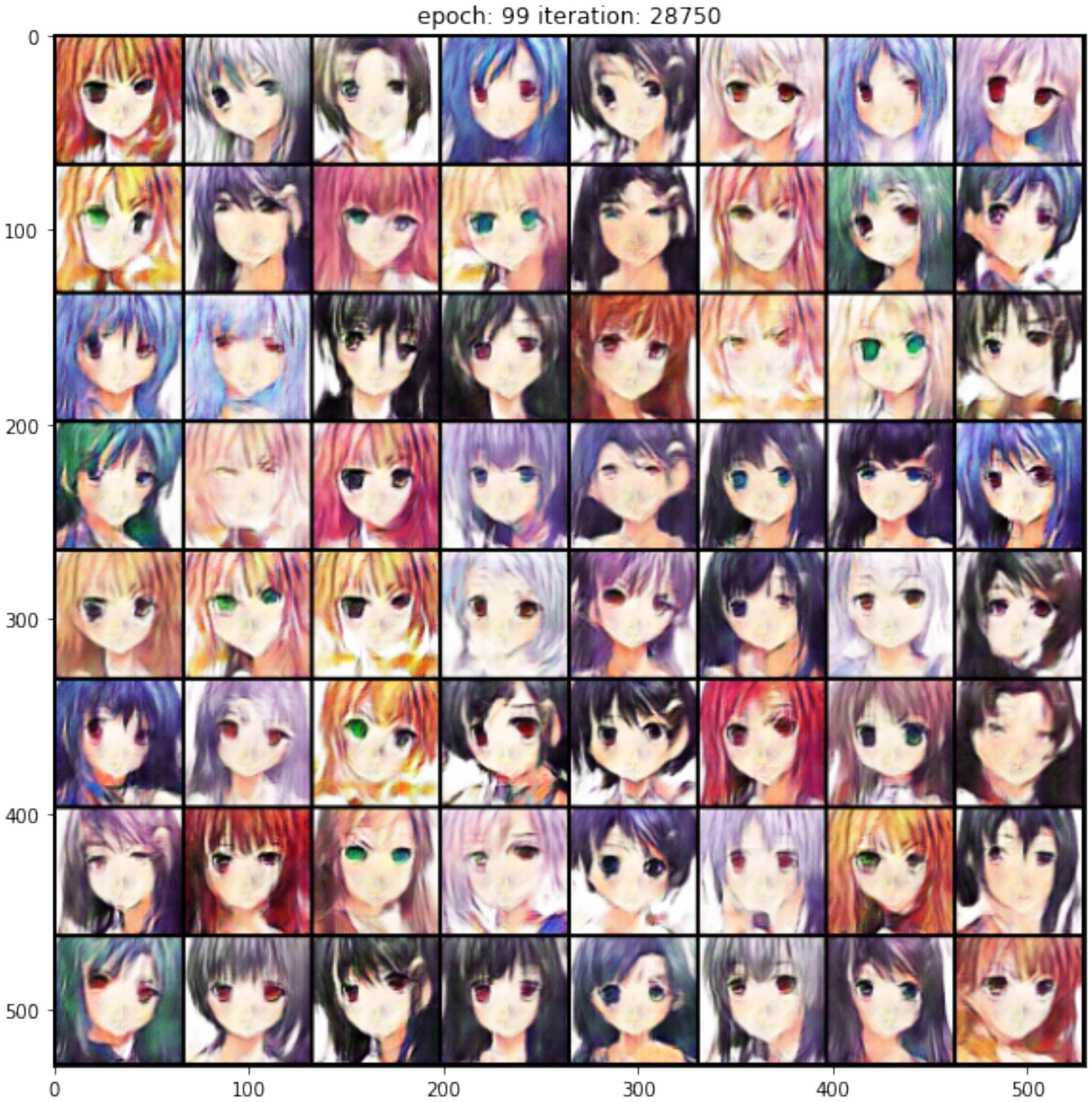
Generator

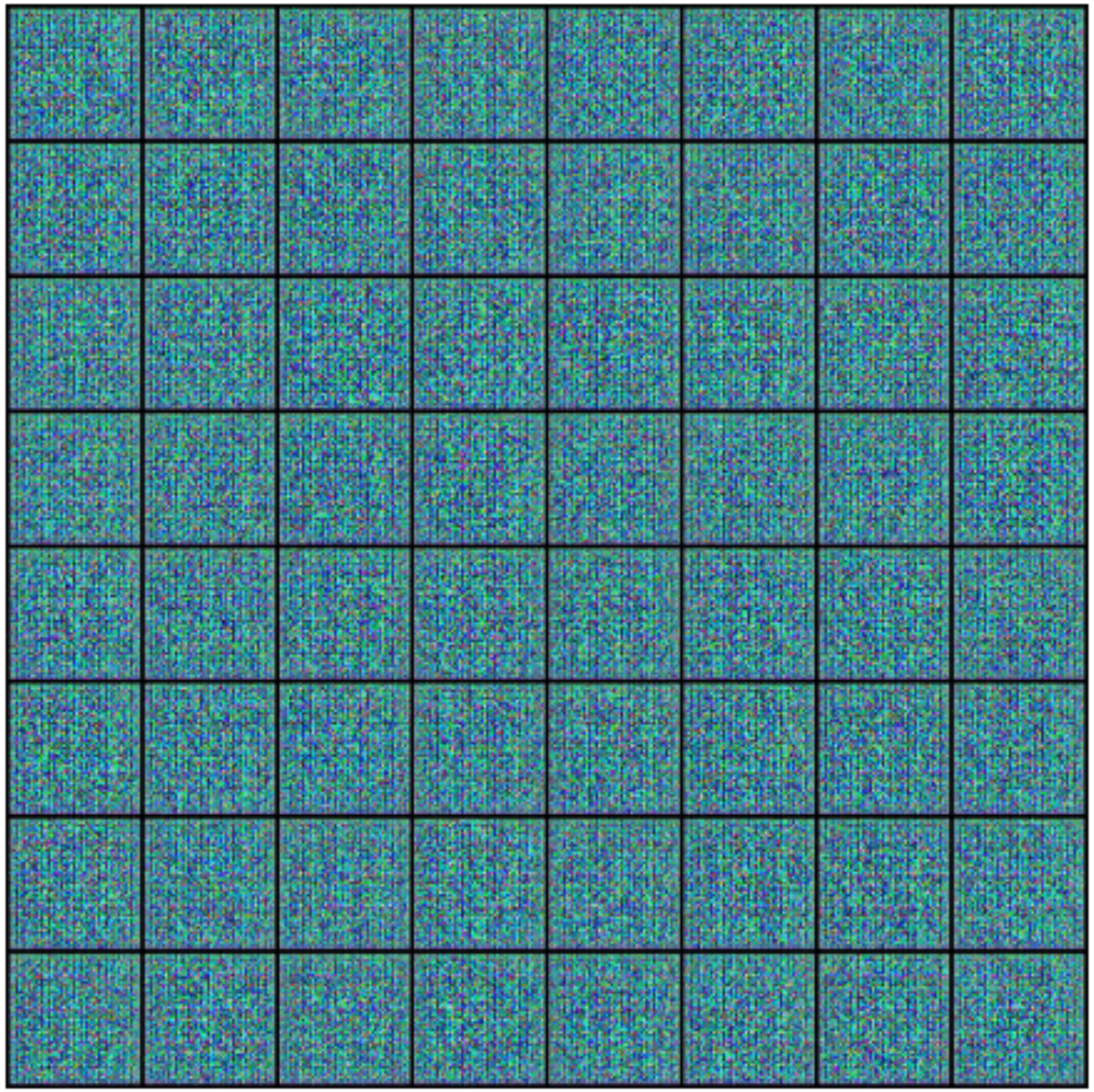
Contrary to the discriminator, the generator is essentially trying ti generate images that the discriminator would approve as real images.

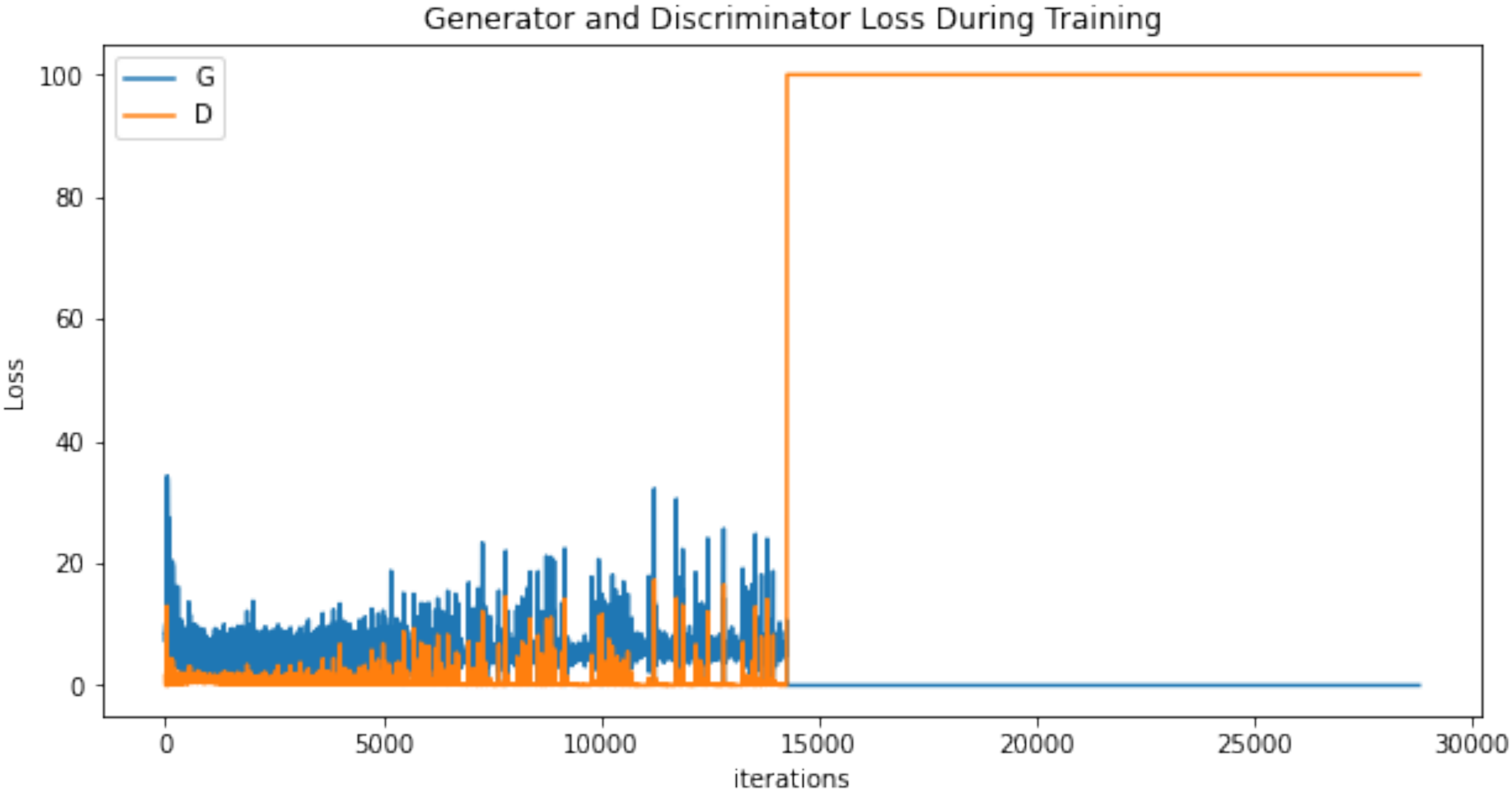
Hence, all the generated images must be predicted as *1* and must be penalized for failing to do so.

Therefore, we train the generator to predict *1*, as the output at the discriminator.

After training, I got the following results






Generator and Discriminator Loss During Training
G
D
Loss
100
80
60
40
20
0
0
5000
10000
15000
20000
25000
30000
iterations

Thank you

« Any sufficiently advanced technology
is indistinguishable from magic »

Arthur C. Clarke

David Anghelone - Team STARS

Inria