Graph Generation

Antonio Longa[1,2]

MobS[1] Lab, Fondazione Bruno Kessler,Trento, Italy
SML[2] Lab, University of Trento, Italy

TABLE OF CONTENTS

01 Generative Adversarial Networks (GANs)

Goal: generate fake objects (e.g. images) similar to real ones
Idea: play an adversarial game with two agents

I. Goodfellow et al., *Generative Adversarial Nets*. in Proc. of NIPS, 2014, pp. 2672--2680.

01 Generative Adversarial Networks (GANs)

Goal: generate fake objects (e.g. images) similar to real ones
Idea: play an adversarial game with two agents

↓

Generator: maps noise z to a fake object x
Discriminator: maps object x to probability of real/fake
Game: The generator tries to fool the discriminator
The discriminator tries to detect the fake objects

I. Goodfellow et al., *Generative Adversarial Nets*. in Proc. of NIPS, 2014, pp. 2672--2680.

01 Generative Adversarial Networks (GANs)

Goal: generate fake objects (e.g. images) similar to real ones
Idea: play an adversarial game with two agents

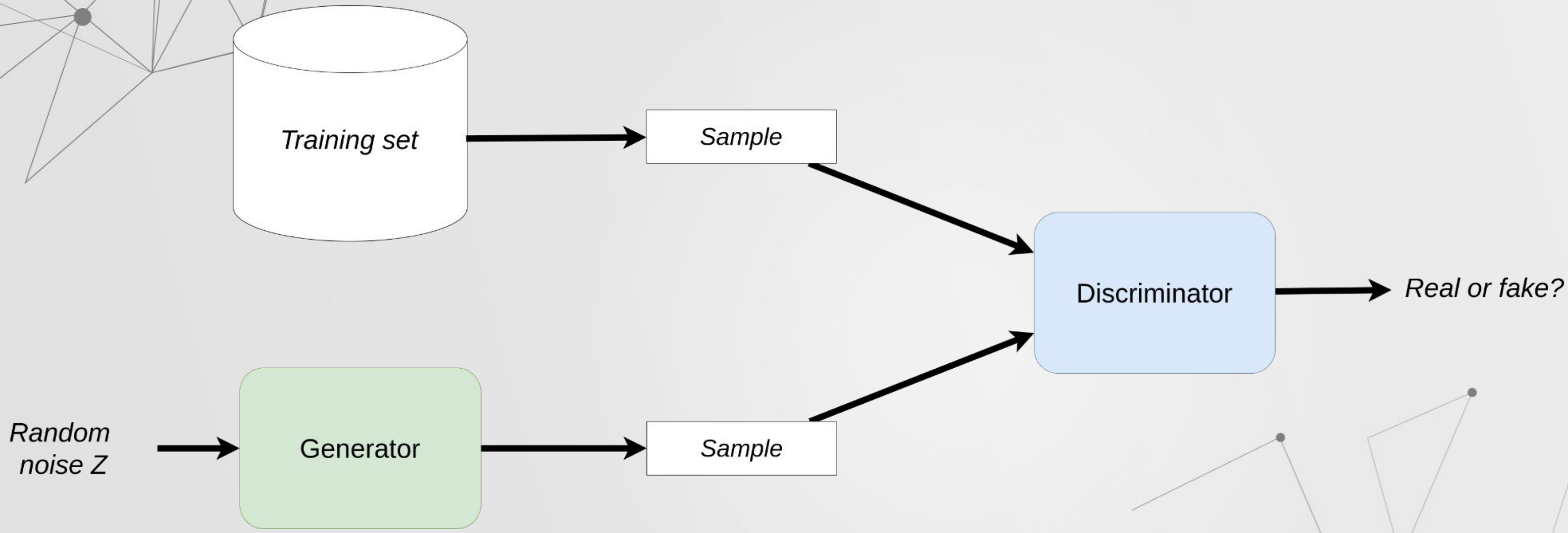
↓

Generator: maps noise z to a fake object x
Discriminator: maps object x to probability of real/fake
Game: The generator tries to fool the discriminator
The discriminator tries to detect the fake objects

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

I. Goodfellow et al., *Generative Adversarial Nets*. in Proc. of NIPS, 2014, pp. 2672--2680.

01 Generative Adversarial Networks (GANs)



I. Goodfellow et al., *Generative Adversarial Nets*. in Proc. of NIPS, 2014, pp. 2672--2680.

01 Generative Adversarial Networks (GANs)

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

I. Goodfellow et al., *Generative Adversarial Nets*. in Proc. of NIPS, 2014, pp. 2672--2680.

01 Generative Adversarial Networks (GANs)

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

The **Discriminator** wants to **max:**

I. Goodfellow et al., *Generative Adversarial Nets*. in Proc. of NIPS, 2014, pp. 2672--2680.

01 Generative Adversarial Networks (GANs)

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

The **Discriminator** wants to **max:**

- Recall that D(x) is in [0, 1]
- **First term**:
  → large if D(x) is close to 1
  → assign high probability to real objects

I. Goodfellow et al., *Generative Adversarial Nets*. in Proc. of NIPS, 2014, pp. 2672--2680.

01 Generative Adversarial Networks (GANs)

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

The **Discriminator** wants to **max:**

- Recall that D(x) is in [0, 1]
- **First term**:
  → large if D(x) is close to 1
  → assign high probability to real objects
- **Second term**:
  → large if 1-D(G(z) is close to 1
  → large if D(G(z)) is close to 0
  → assign low probability to fake objects

I. Goodfellow et al., *Generative Adversarial Nets*. in Proc. of NIPS, 2014, pp. 2672--2680.

01 Generative Adversarial Networks (GANs)

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

The **Generator** wants to **min:**

I. Goodfellow et al., *Generative Adversarial Nets*. in Proc. of NIPS, 2014, pp. 2672--2680.

01 Generative Adversarial Networks (GANs)

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

The **Generator** wants to **min:**

- **Second term**:
  → small if 1-D(G(z) is close to 0
  → small if D(G(z) is close to 1
  → fool the discriminator into assigning high probability to fake objects

I. Goodfellow et al., *Generative Adversarial Nets*. in Proc. of NIPS, 2014, pp. 2672--2680.

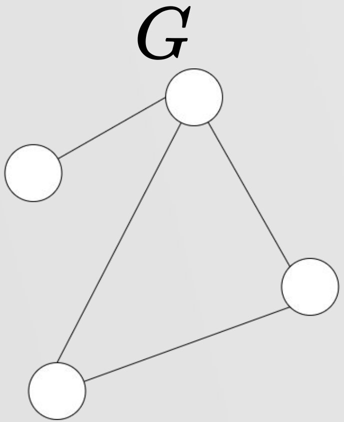
02 Learning social Graph using topologies using GANs

Learning Social Graph Topologies using Generative Adversarial Neural Networks

Sahar Tavakoli[1], Alireza Hajibagheri[1], and Gita Sukthankar[1]

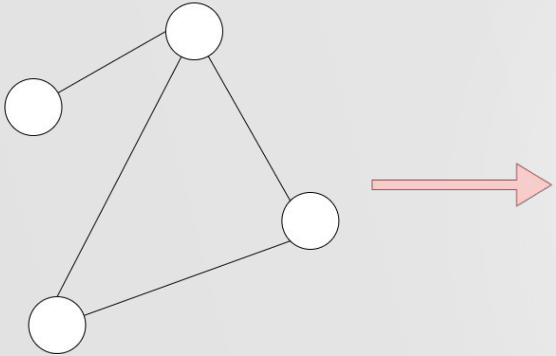
[1]University of Central Florida, Orlando, Florida
sahar@knights.ucf.edu,alireza@eecs.ucf.edu,gitars@eecs.ucf.edu

02 Learning social Graph using topologies using GANs



02 Learning social Graph using topologies using GANs

G

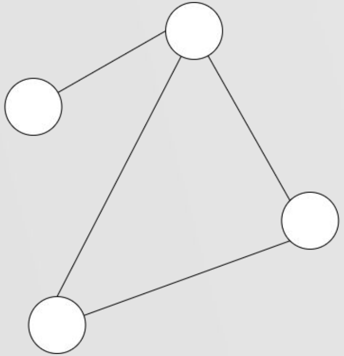


$Adj(G)$

0	1	0	0
1	0	1	1
0	1	0	1
0	1	1	0

02 Learning social Graph using topologies using GANs

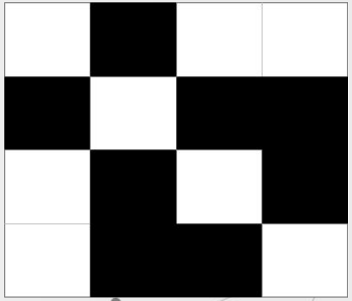
G



$Adj(G)$

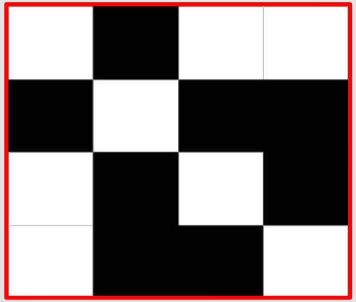
0	1	0	0
1	0	1	1
0	1	0	1
0	1	1	0

Img



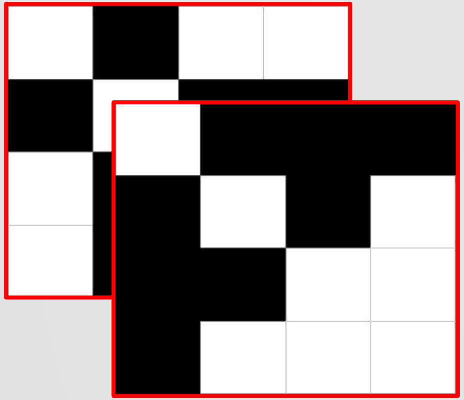
02 Learning social Graph using topologies using GANs

Img



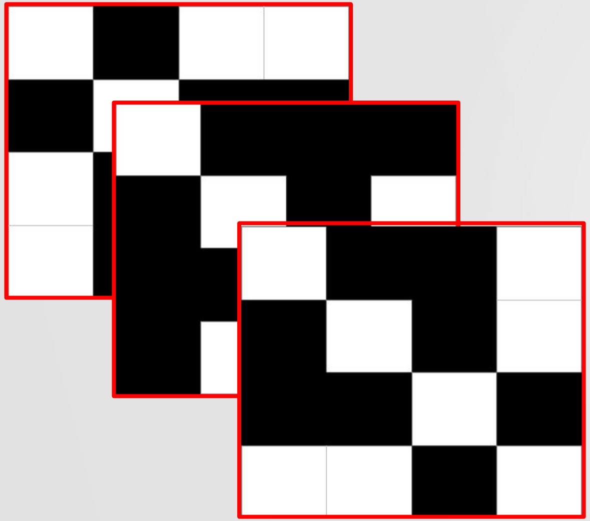
02 Learning social Graph using topologies using GANs

Img



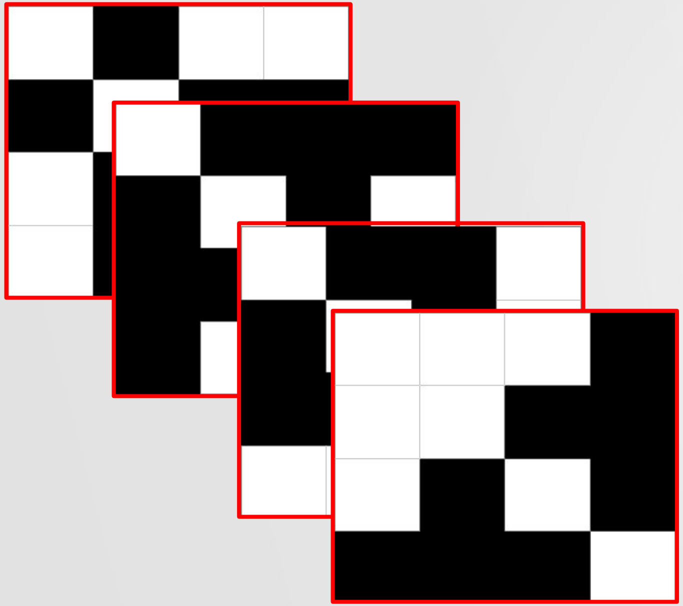
02 Learning social Graph using topologies using GANs

Img

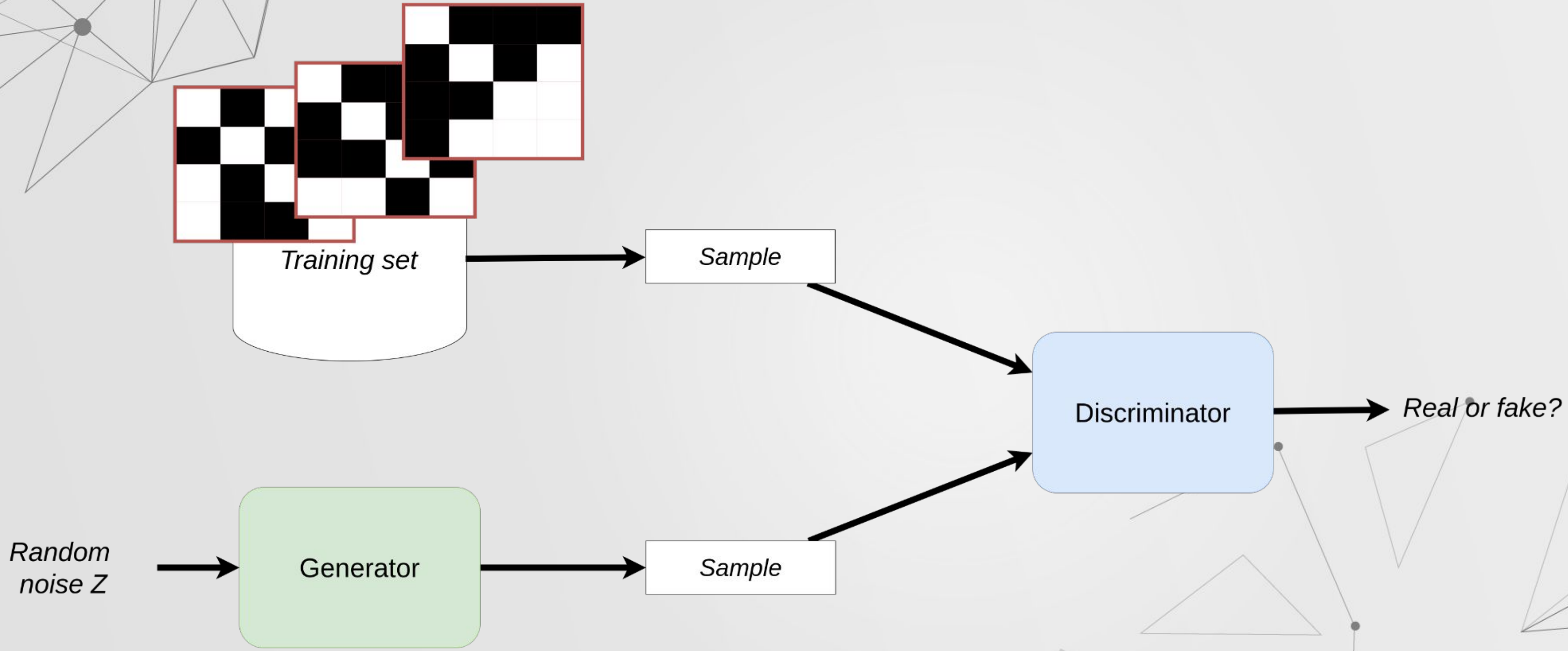


02 Learning social Graph using topologies using GANs

Img



02 Learning social Graph using topologies using GANs



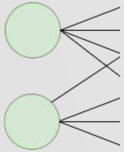
02 Learning social Graph using topologies using GANs

Generator

INPUT

$$I \in \mathbf{R}^{100}$$

$$I \sim N(0,1)$$



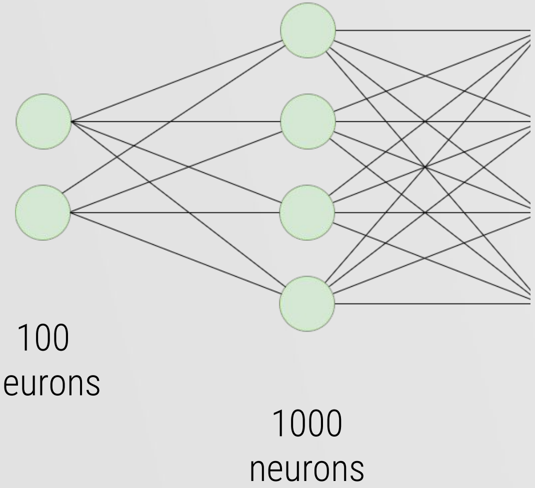
100
neurons

02 Learning social Graph using topologies using GANs

Generator

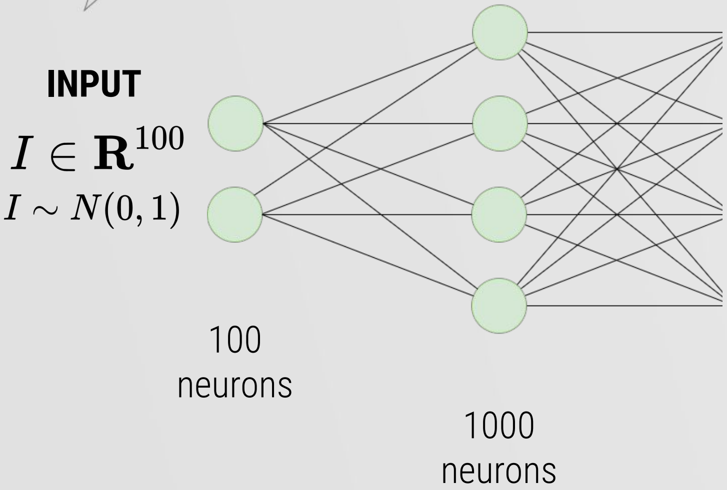
INPUT

$$I \in \mathbf{R}^{100}$$
$$I \sim N(0,1)$$



02 Learning social Graph using topologies using GANs

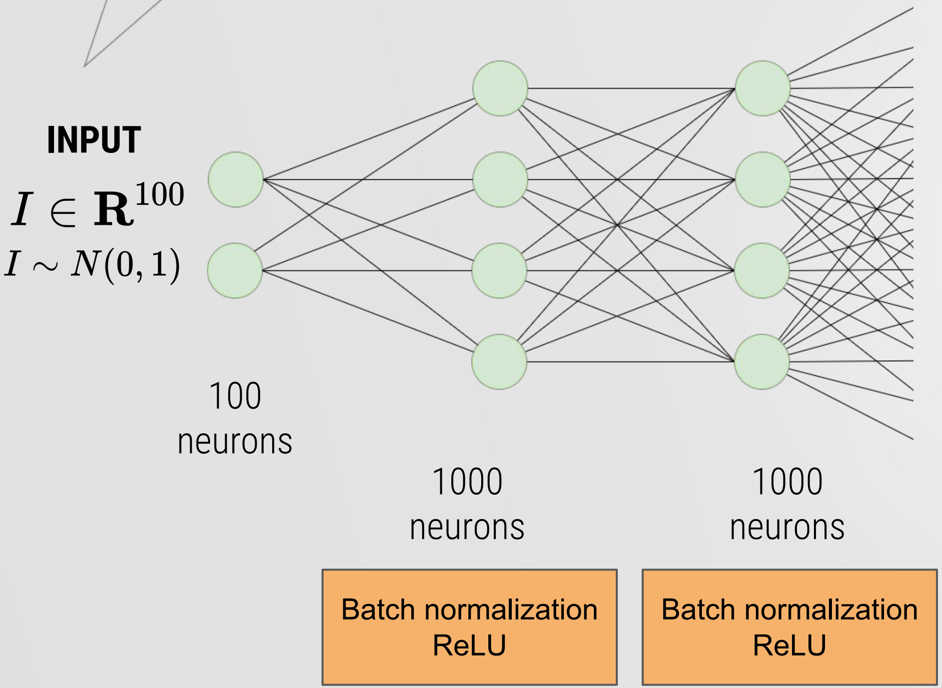
Generator



Batch normalization
ReLU

02 Learning social Graph using topologies using GANs

Generator



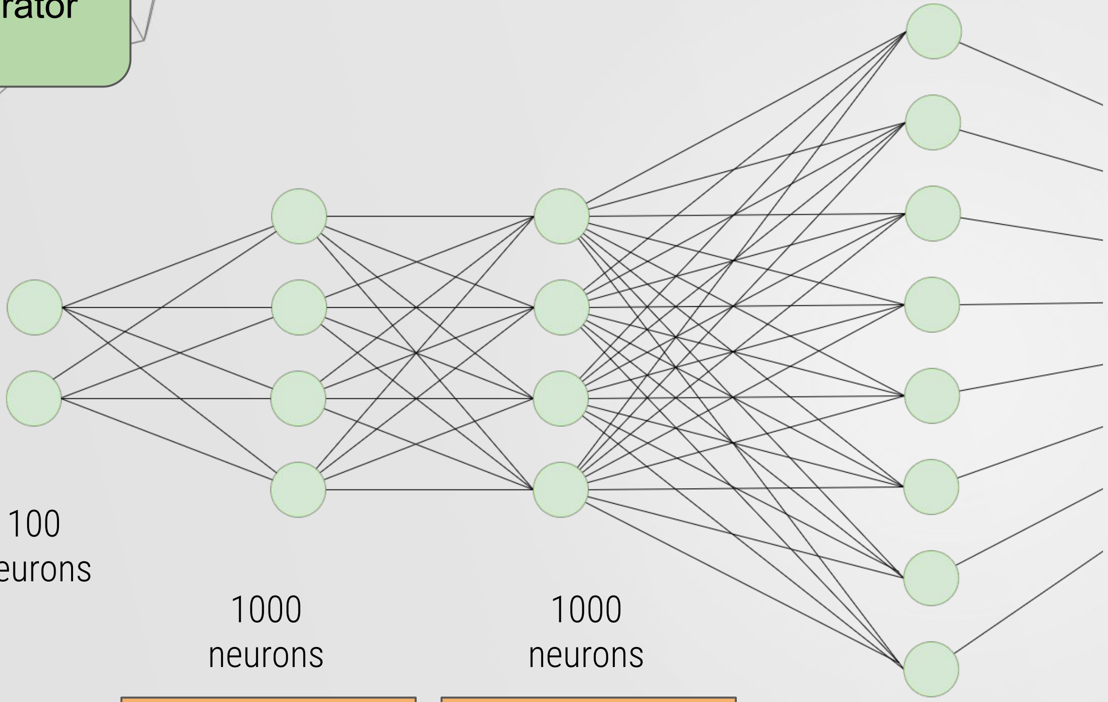
02 Learning social Graph using topologies using GANs

Generator

INPUT

$I \in \mathbf{R}^{100}$

$I \sim N(0,1)$



100 neurons

1000 neurons

1000 neurons

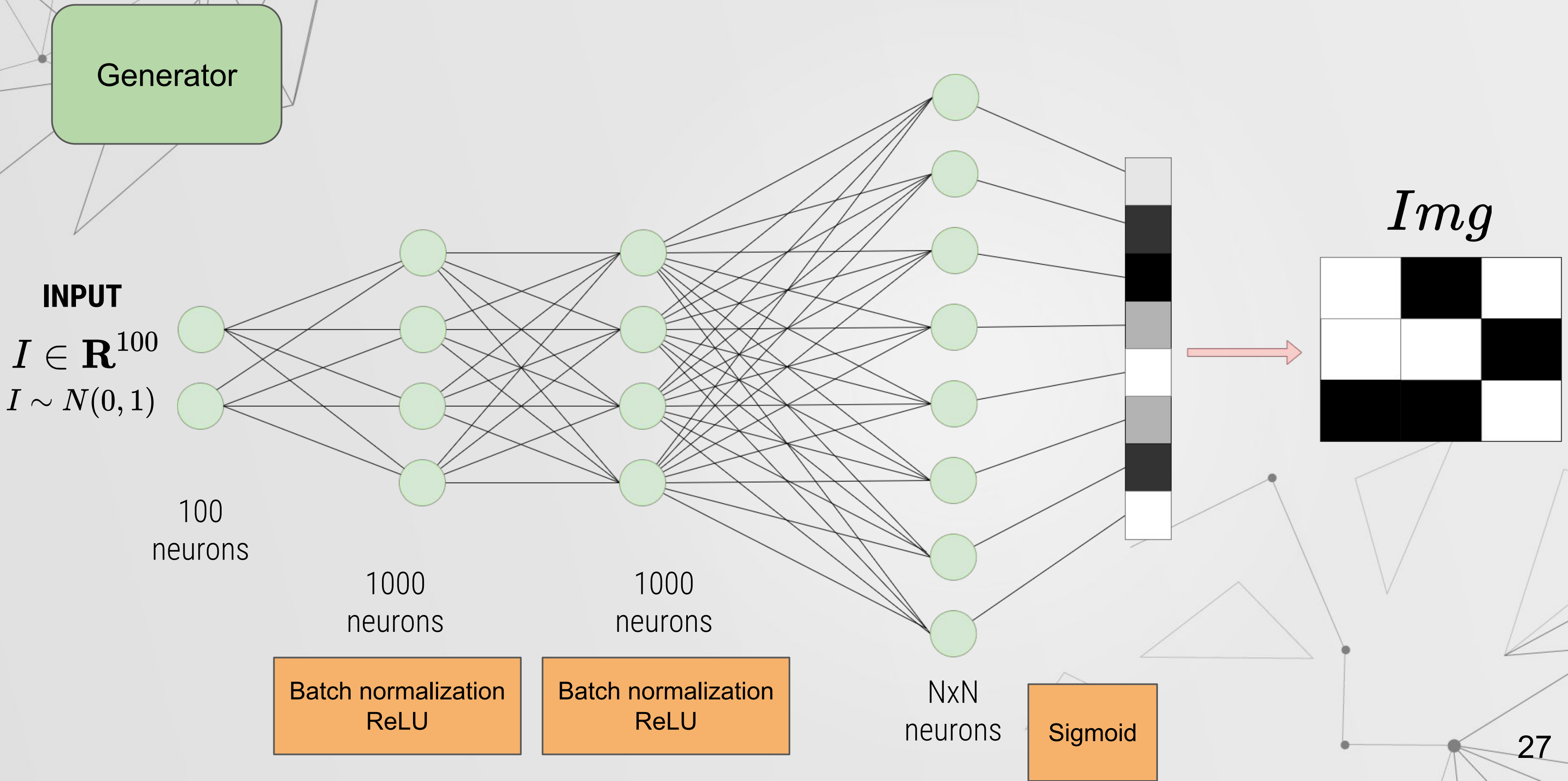
NxN neurons

Batch normalization ReLU

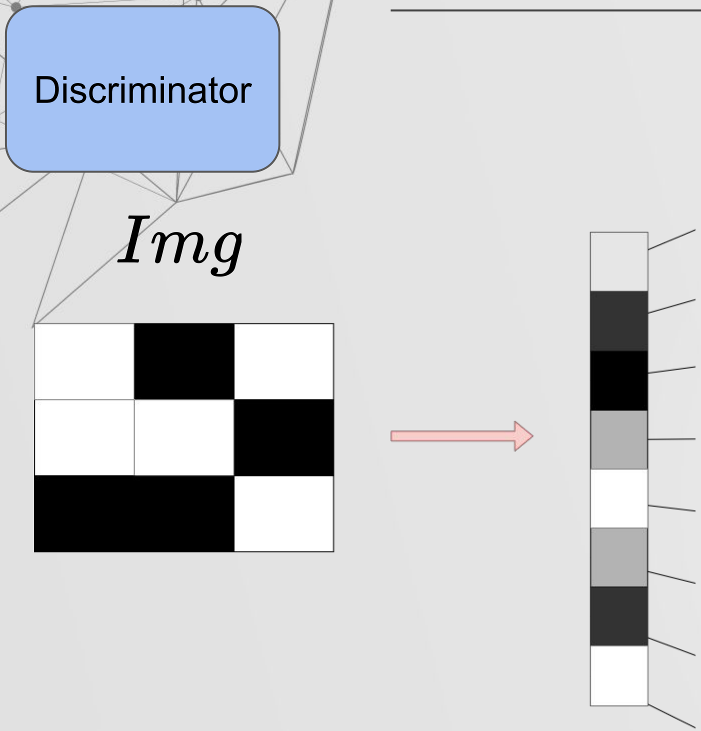
Batch normalization ReLU

Sigmoid

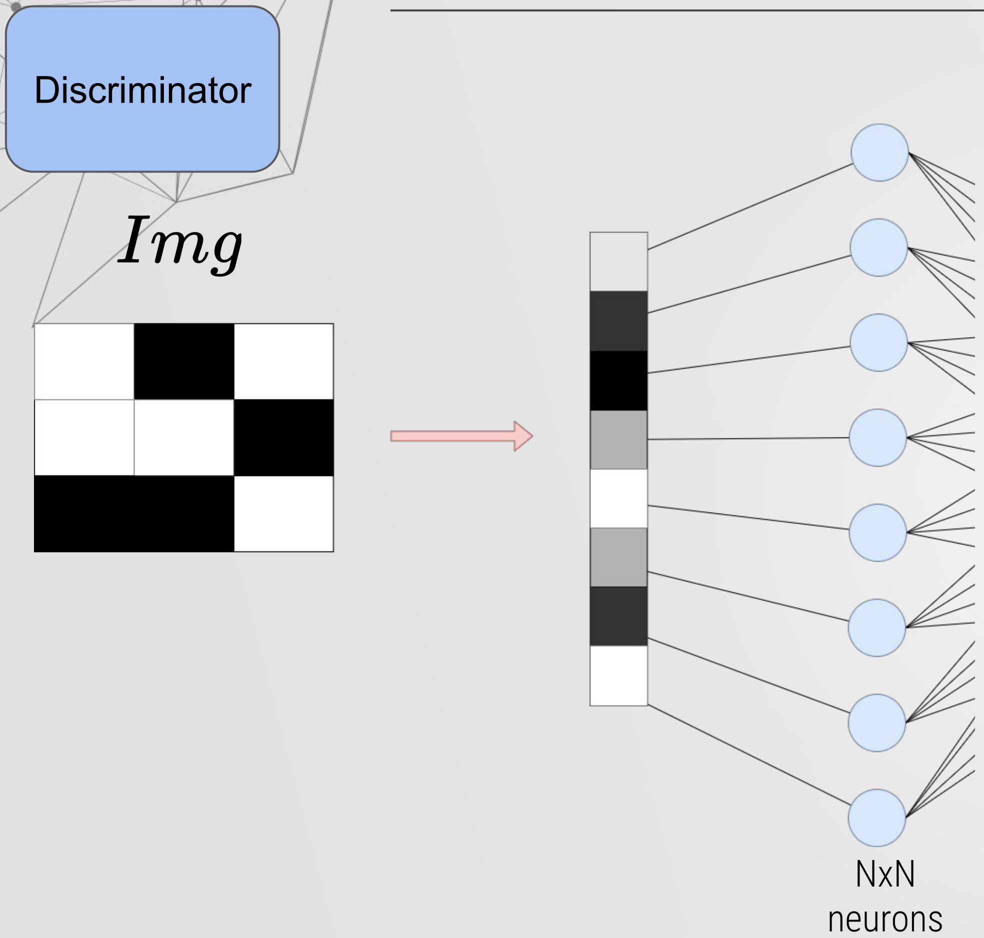
02 Learning social Graph using topologies using GANs



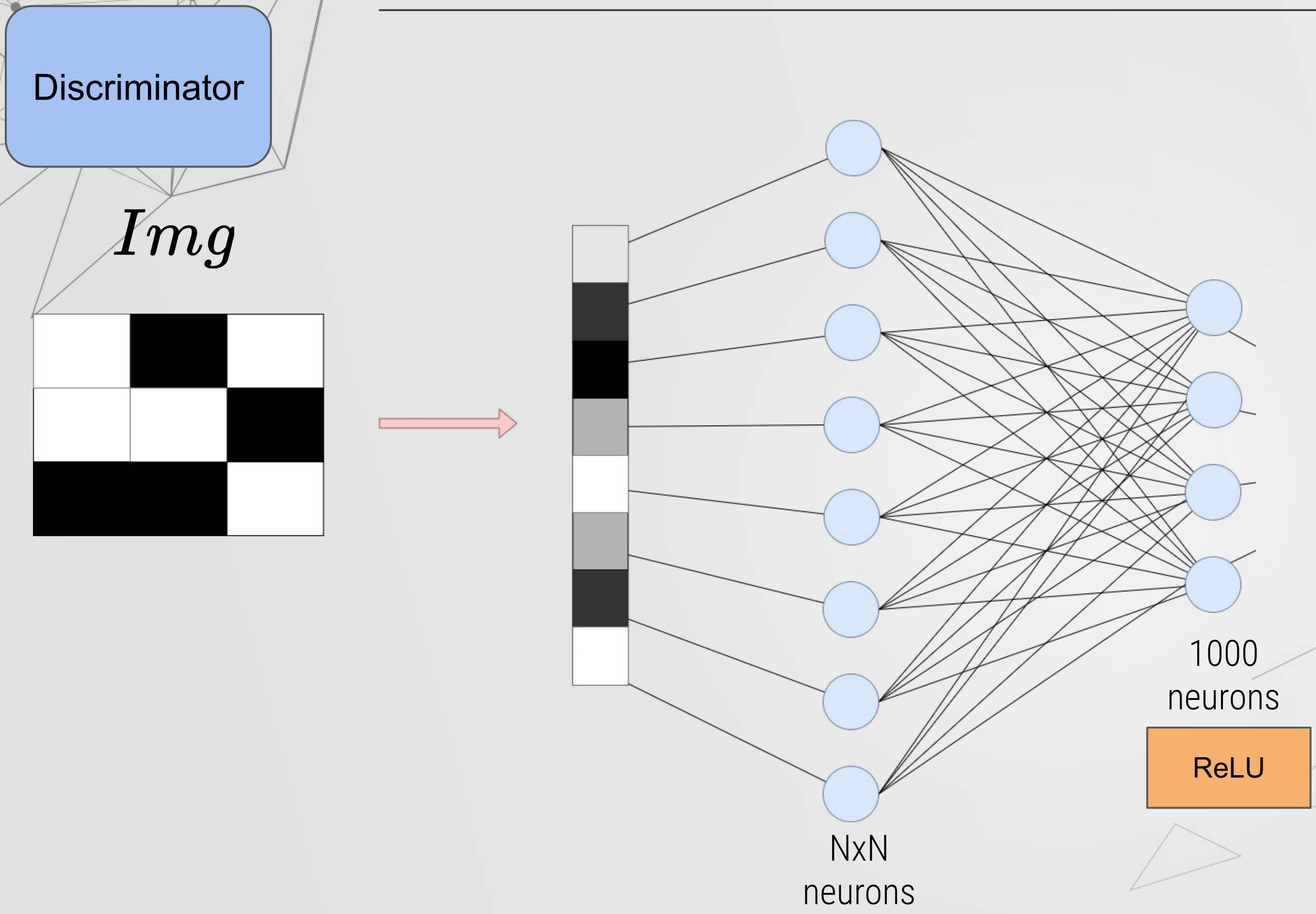
02 Learning social Graph using topologies using GANs



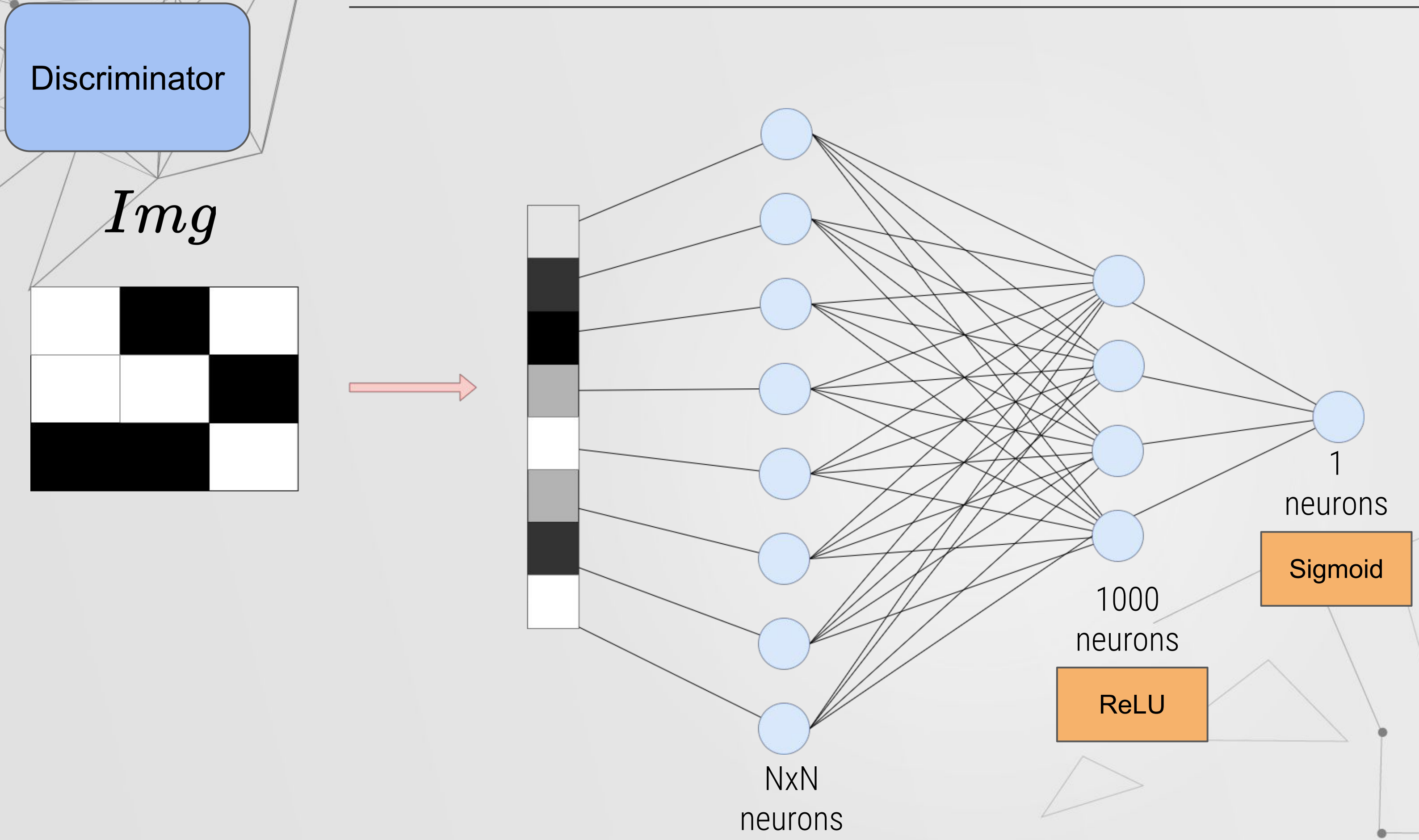
02 Learning social Graph using topologies using GANs



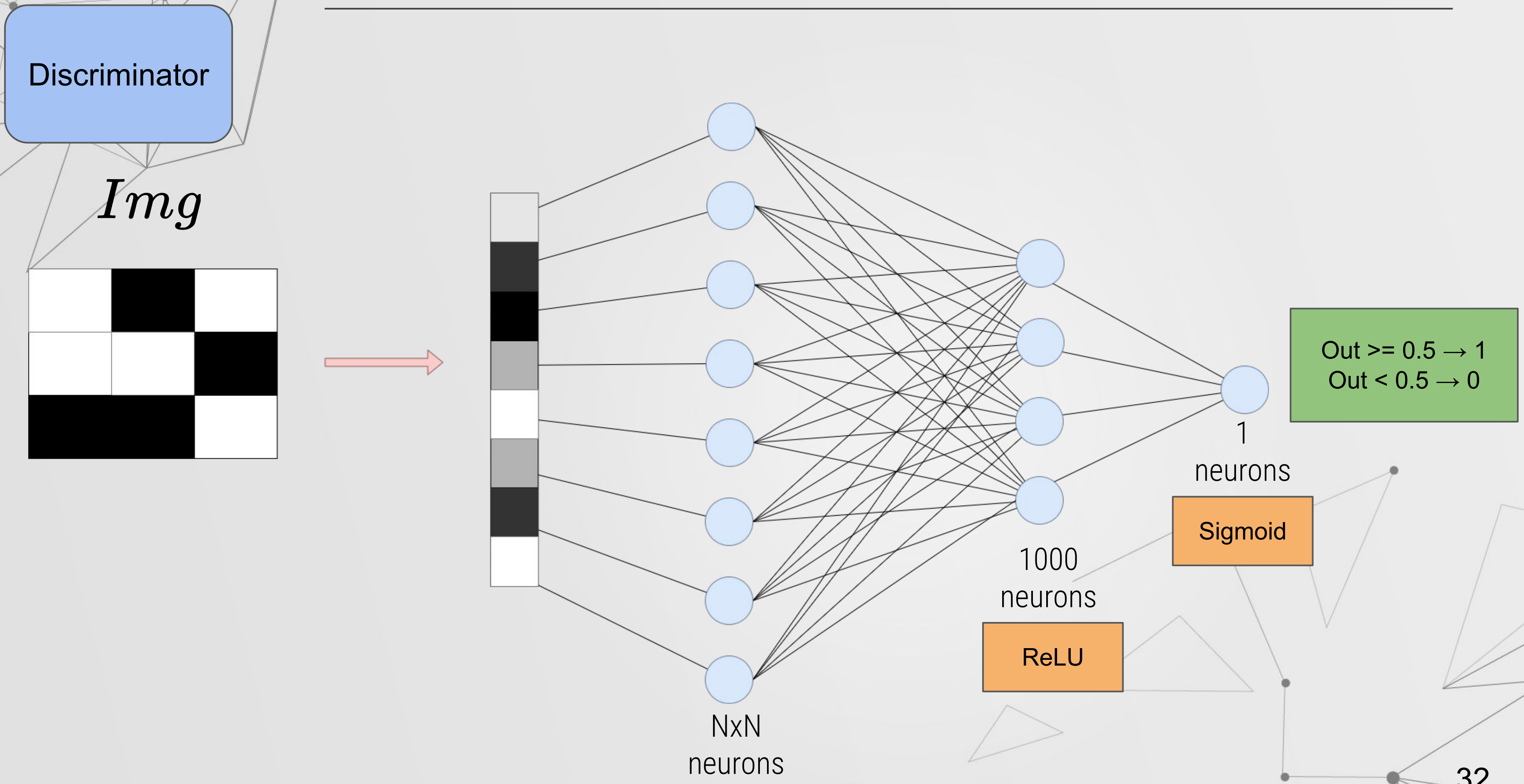
02 Learning social Graph using topologies using GANs



02 Learning social Graph using topologies using GANs



02 Learning social Graph using topologies using GANs



02 Learning social Graph using topologies using GANs

Results:

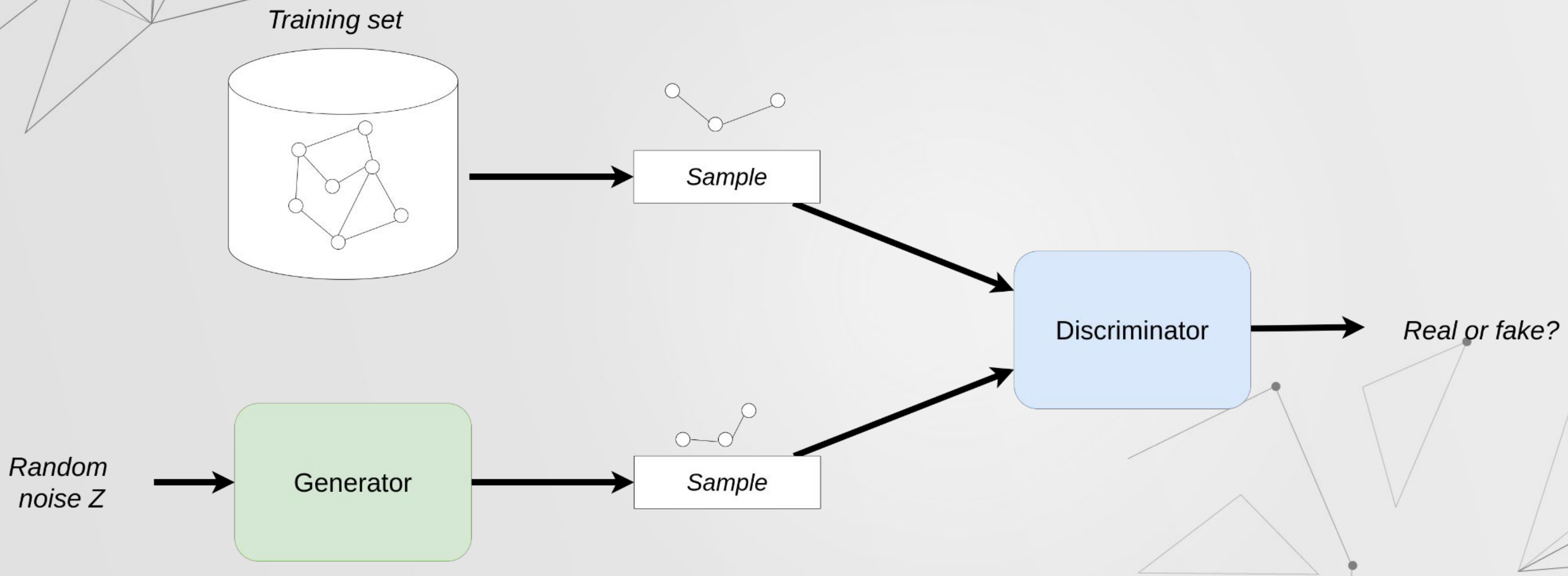
- **Comparison** of several **features** between original and generated graph. (Nb nodes, nb.edges, avg. degree, diameter, assortativity, etc ..)
- On several **social interaction networks** (Karate club, Football, Dolphins, Enron)

03 NetGAN

NetGAN: Generating Graphs via Random Walks

Aleksandar Bojchevski [*1] Oleksandr Shchur [*1] Daniel Zügner [*1] Stephan Günnemann [1]

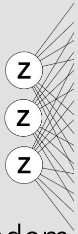
03 NetGAN



Training set
Sample
Discriminator
Real or fake?
Random
noise Z
Generator
Sample

03 NetGAN

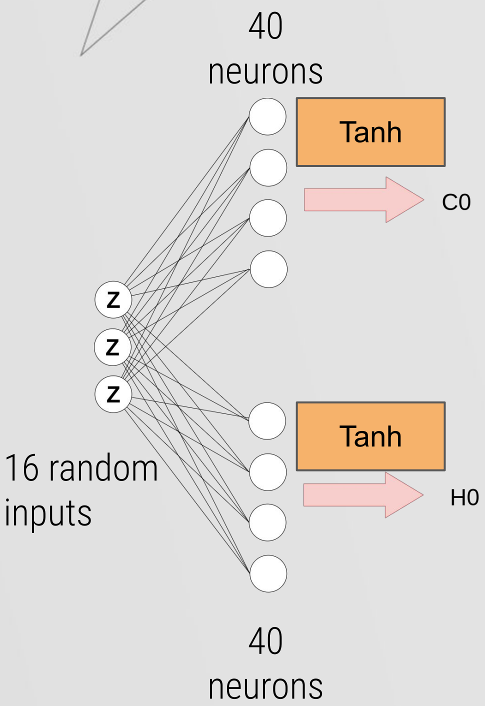
Generator



16 random inputs

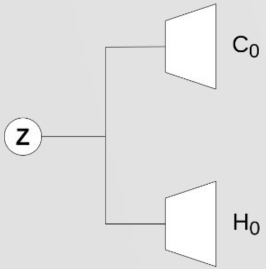
03 NetGAN

Generator

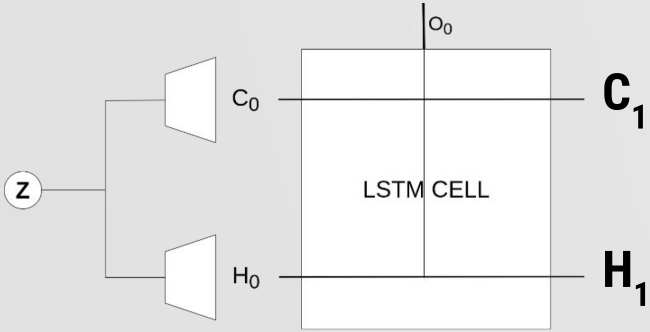


03 NetGAN

Generator



03 NetGAN

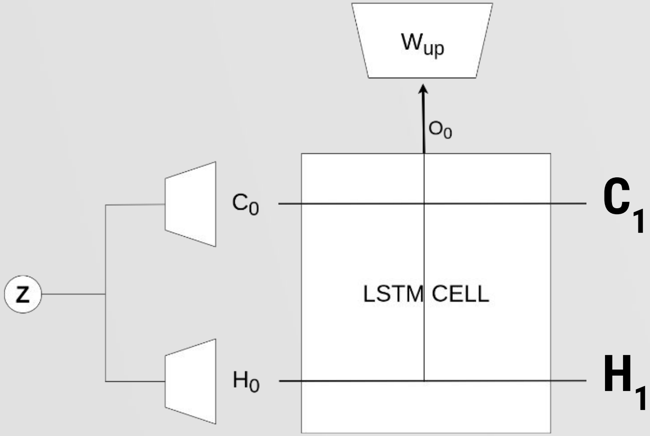


LSTM cell

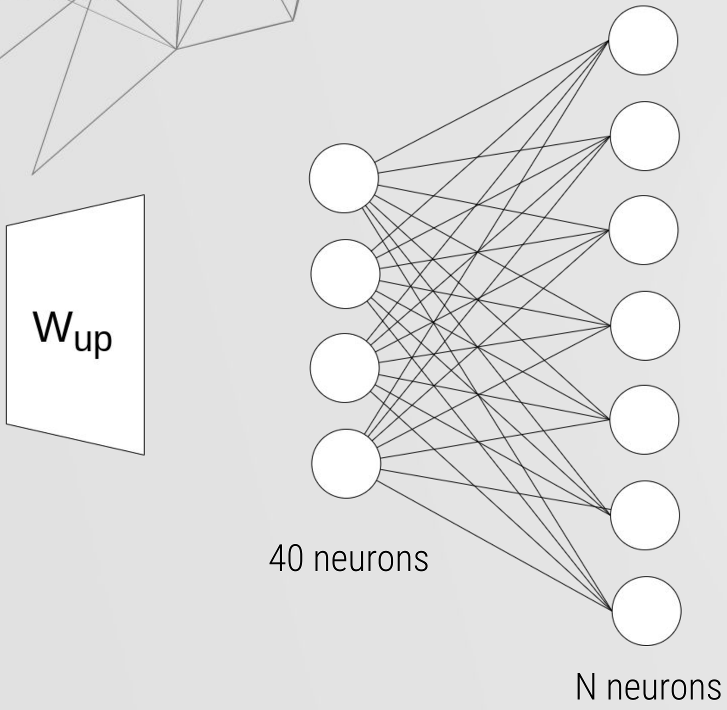
Generator

03 NetGAN

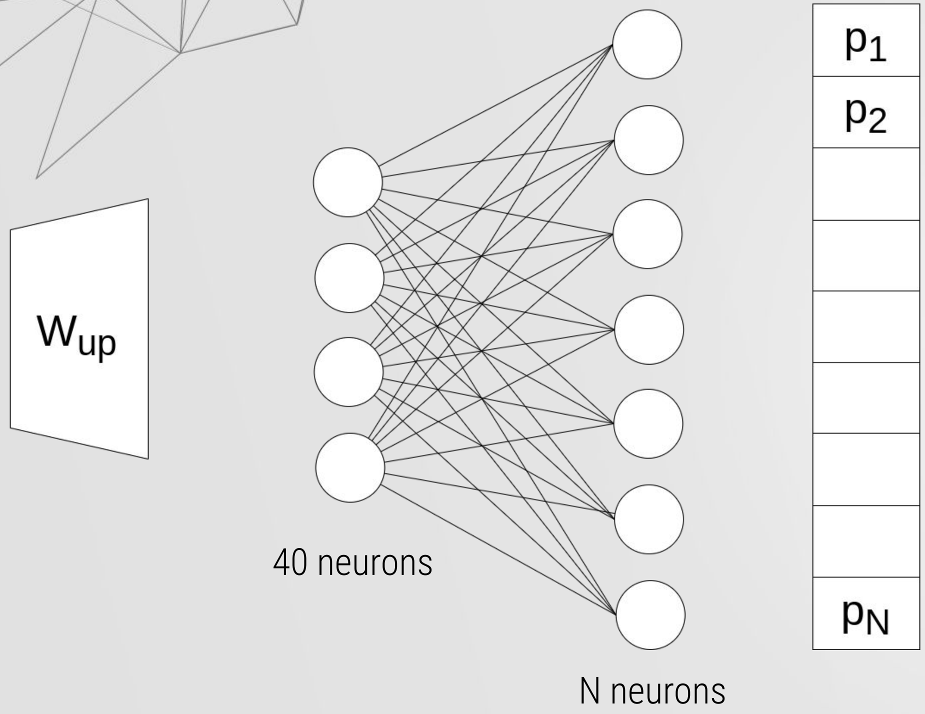
W_{up}



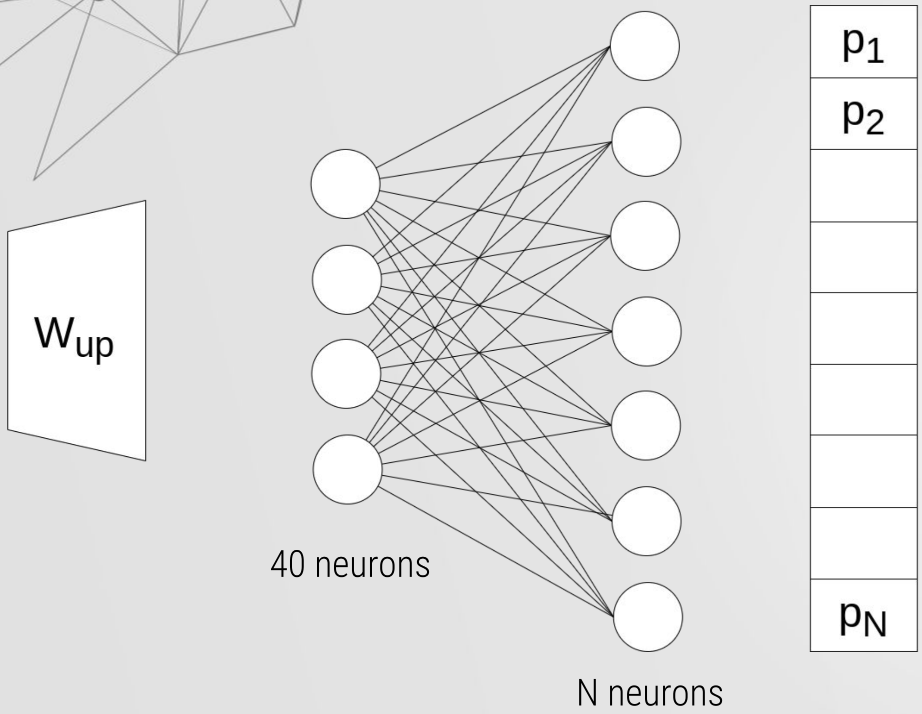
03 NetGAN



03 NetGAN



03 NetGAN

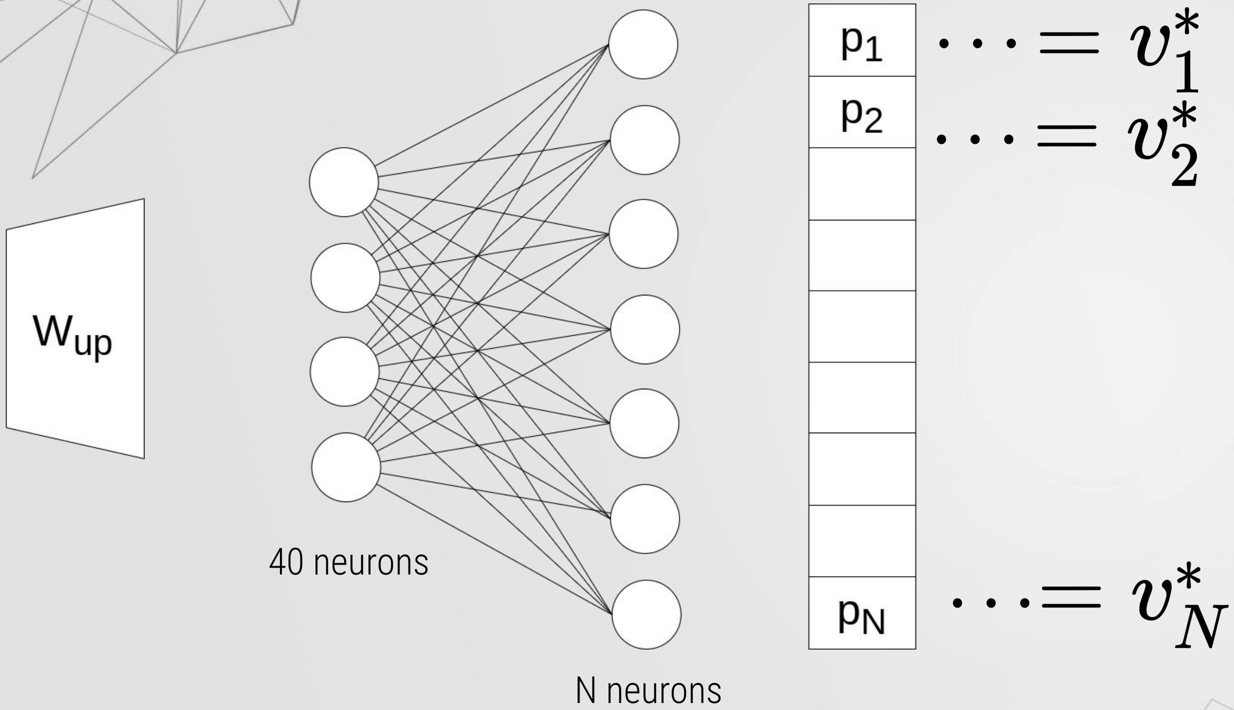


$$\sigma((p_1 + g)/\tau) = v_1^*$$

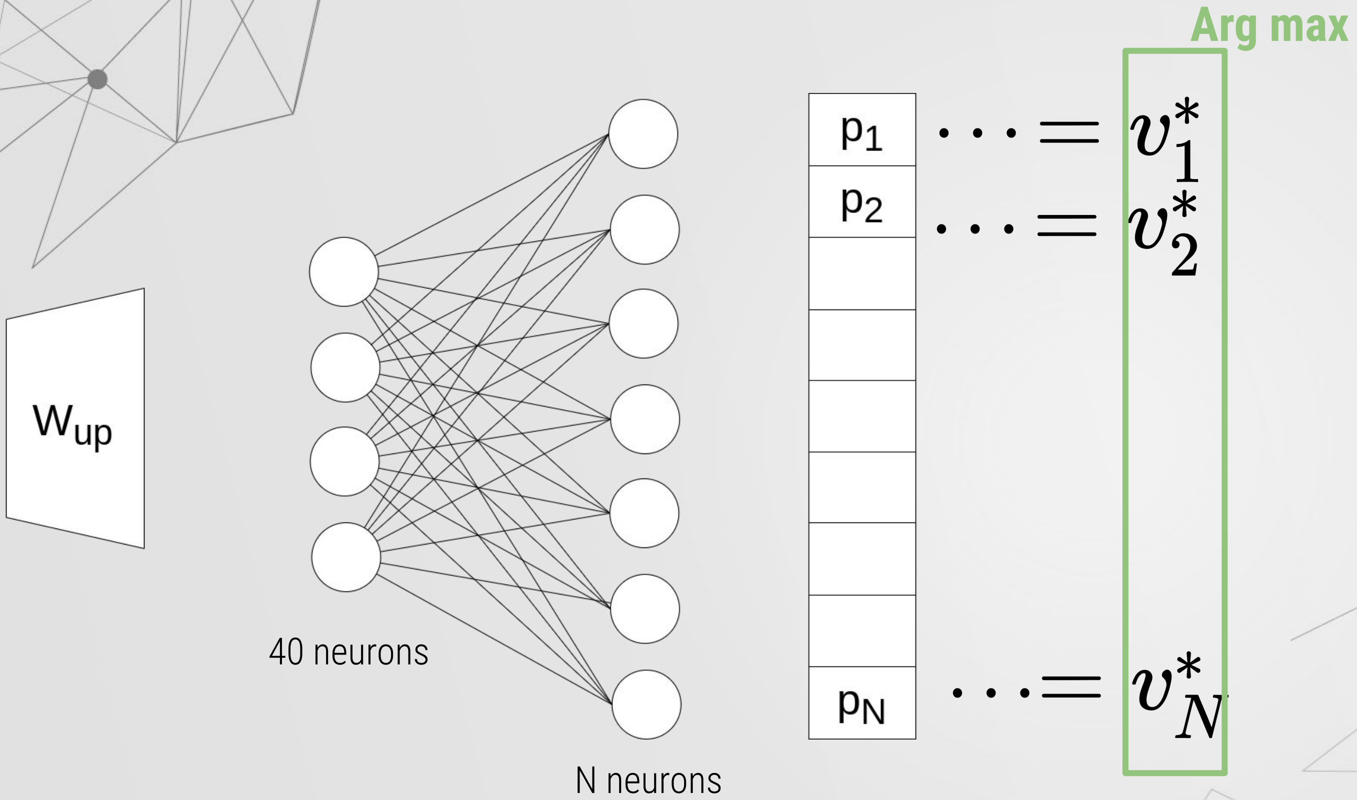
Temperature param

i.i.d. Sample from Gumbel dist.

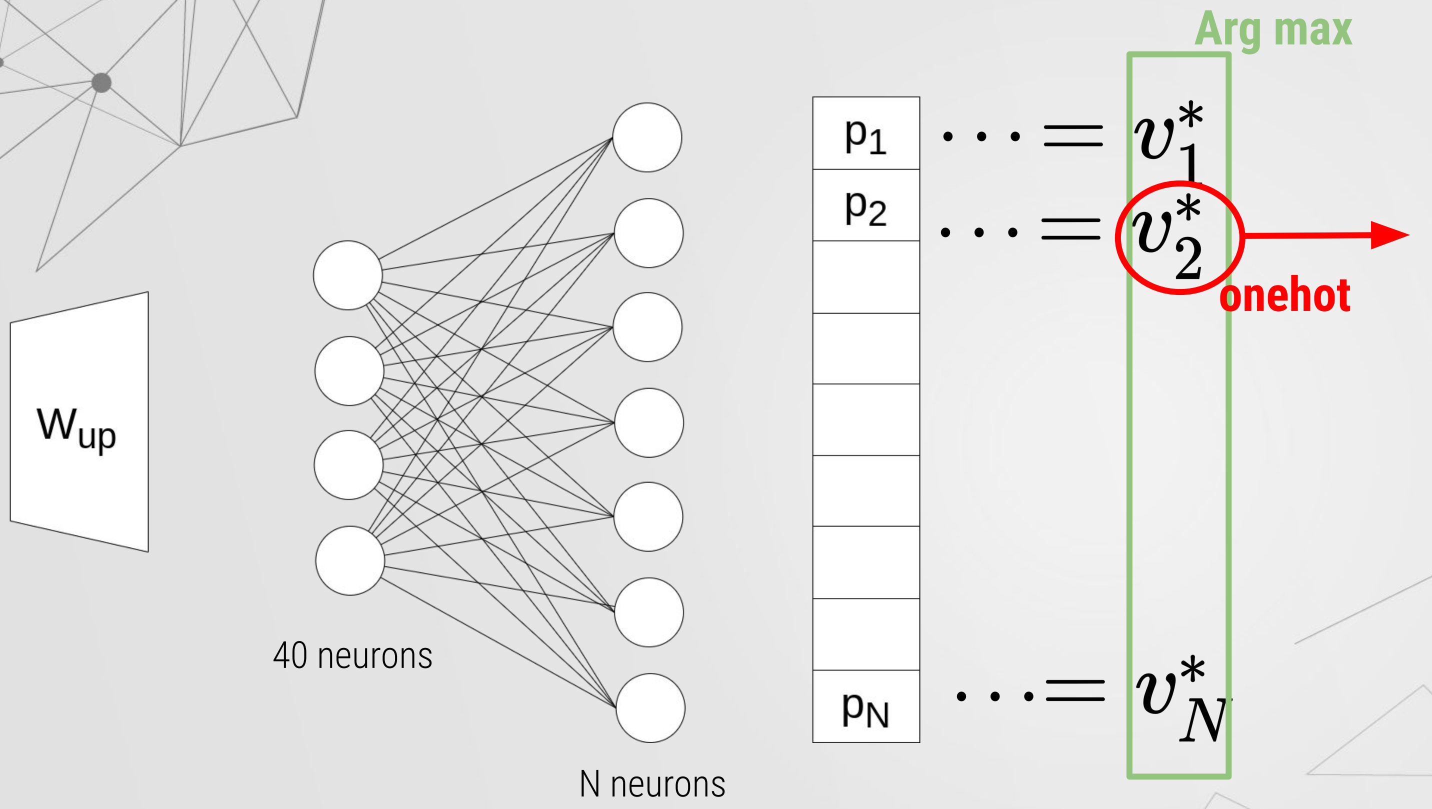
03 NetGAN



03 NetGAN

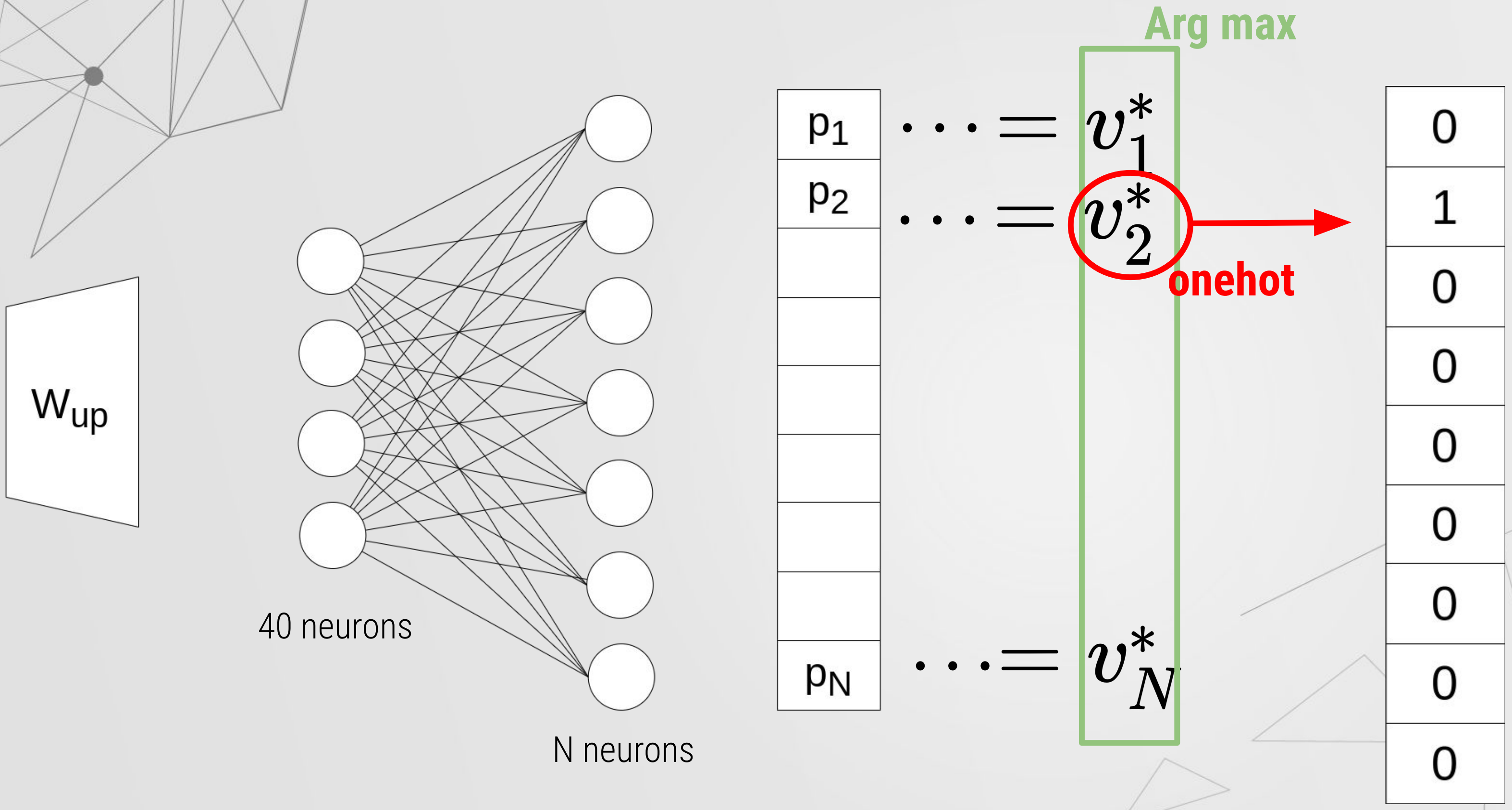

Arg max
W_{up}
40 neurons
N neurons
p_1 $\cdots = v_1^*$
p_2 $\cdots = v_2^*$
p_N $\cdots = v_N^*$

03 NetGAN



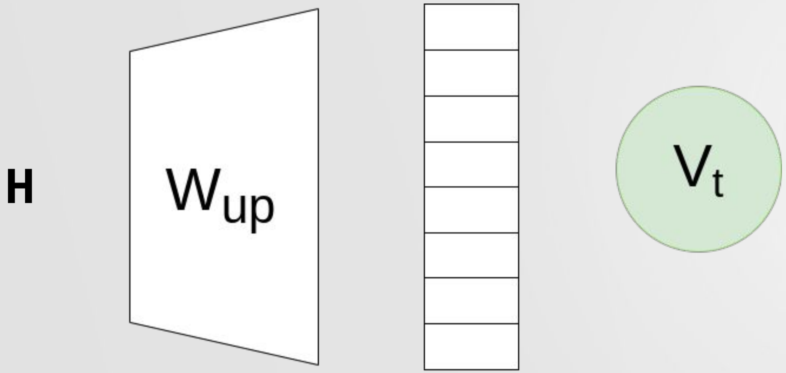
Arg max
W_{up}
40 neurons
N neurons
p_1
p_2
p_N
$\cdots = v_1^*$
$\cdots = v_2^*$
$\cdots = v_N^*$
onehot

03 NetGAN



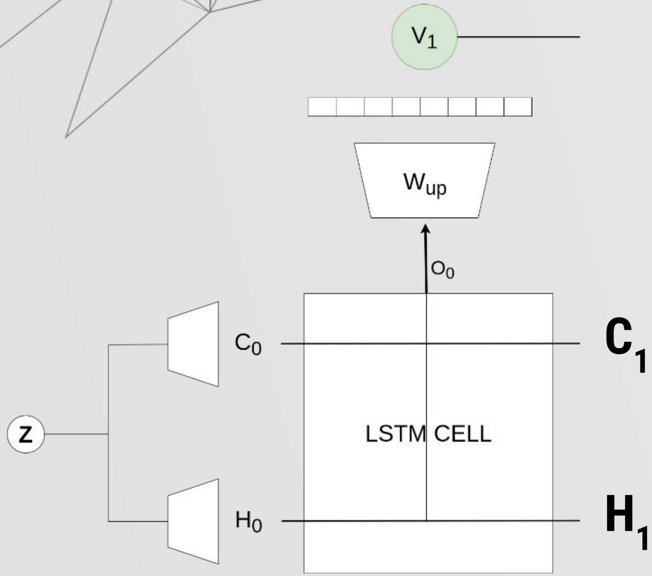
Arg max
W_{up}
40 neurons
N neurons
p_1
p_2
p_N
$\cdots = v_1^*$
$\cdots = v_2^*$
$\cdots = v_N^*$
onehot
0
1
0
0
0
0
0
0
0

03 NetGAN



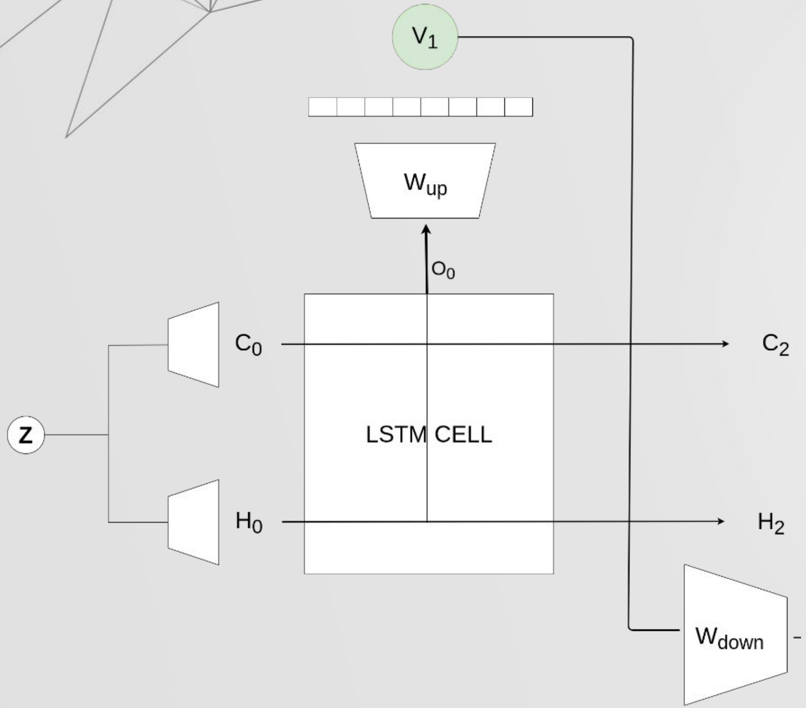
03 NetGAN

Generator



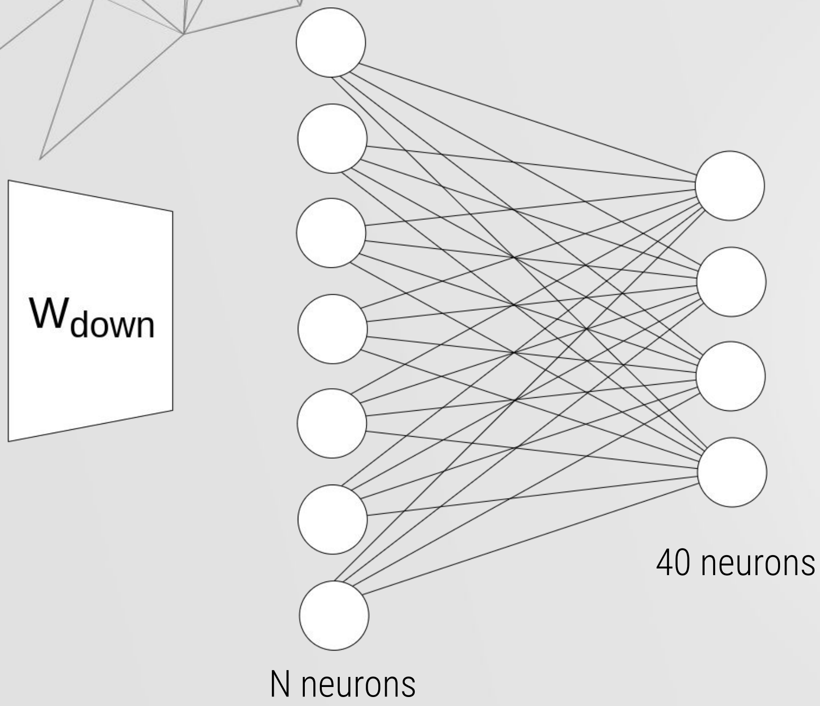
Generator

03 NetGAN



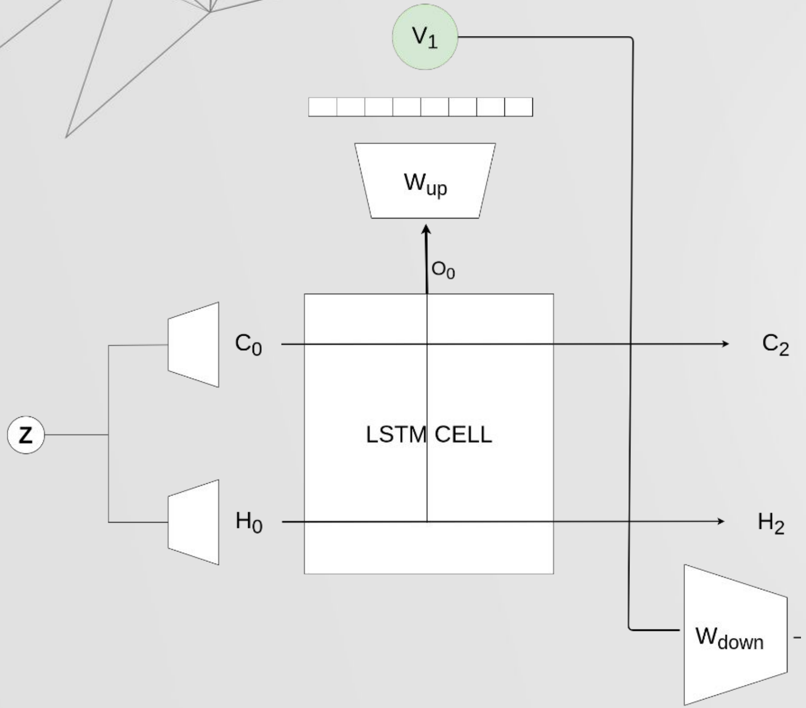
W_{down}

03 NetGAN



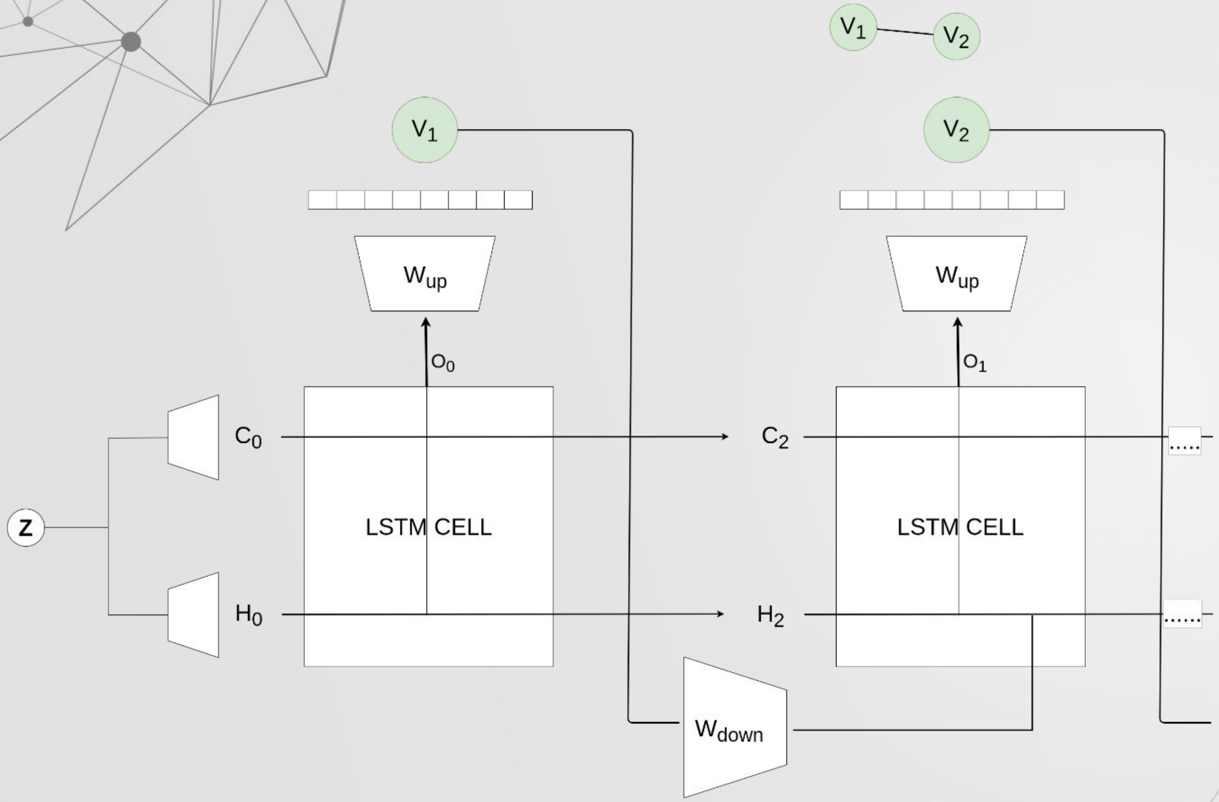
03 NetGAN

Generator

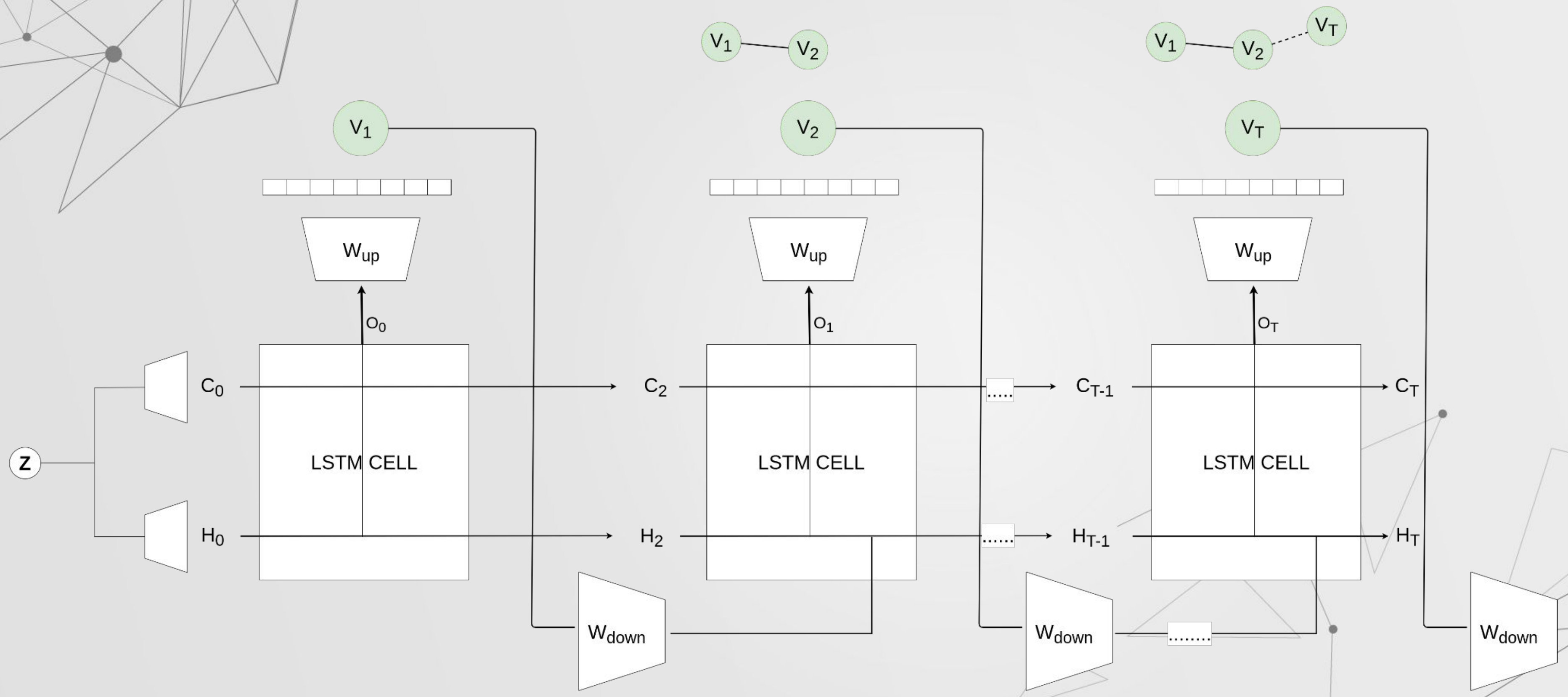


Generator

03 NetGAN

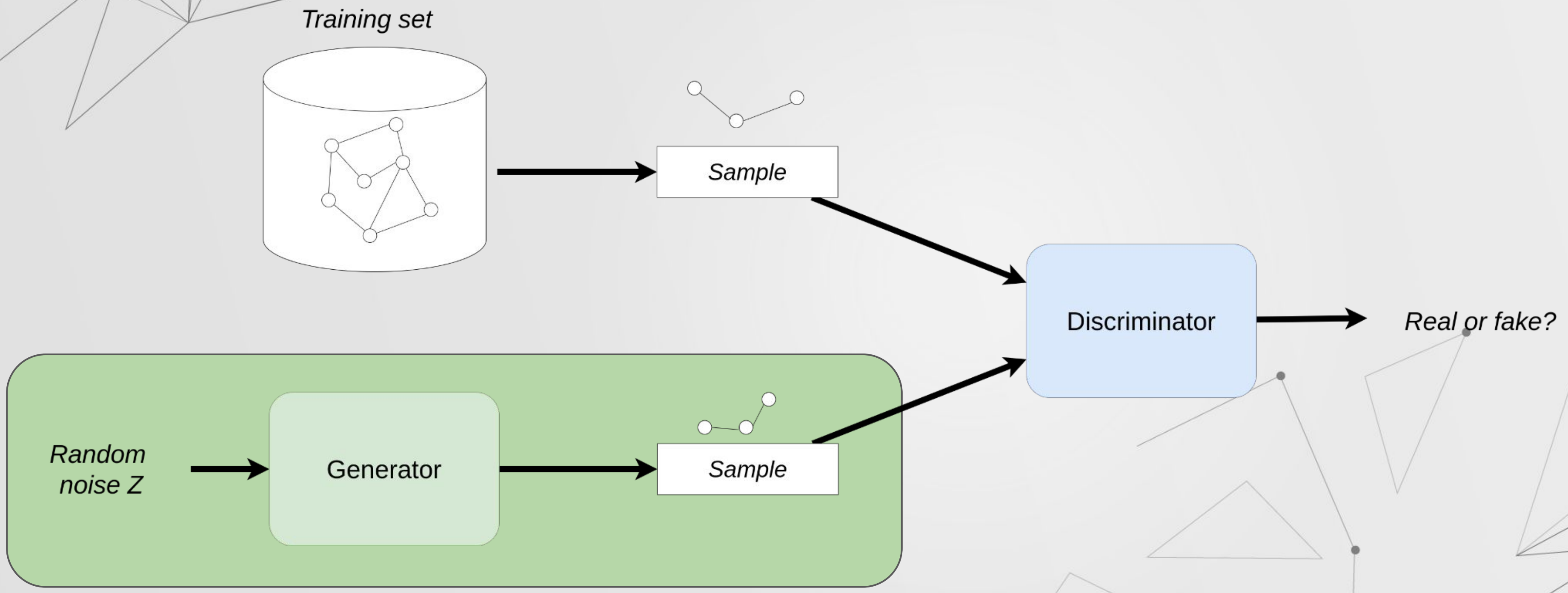


03 NetGAN

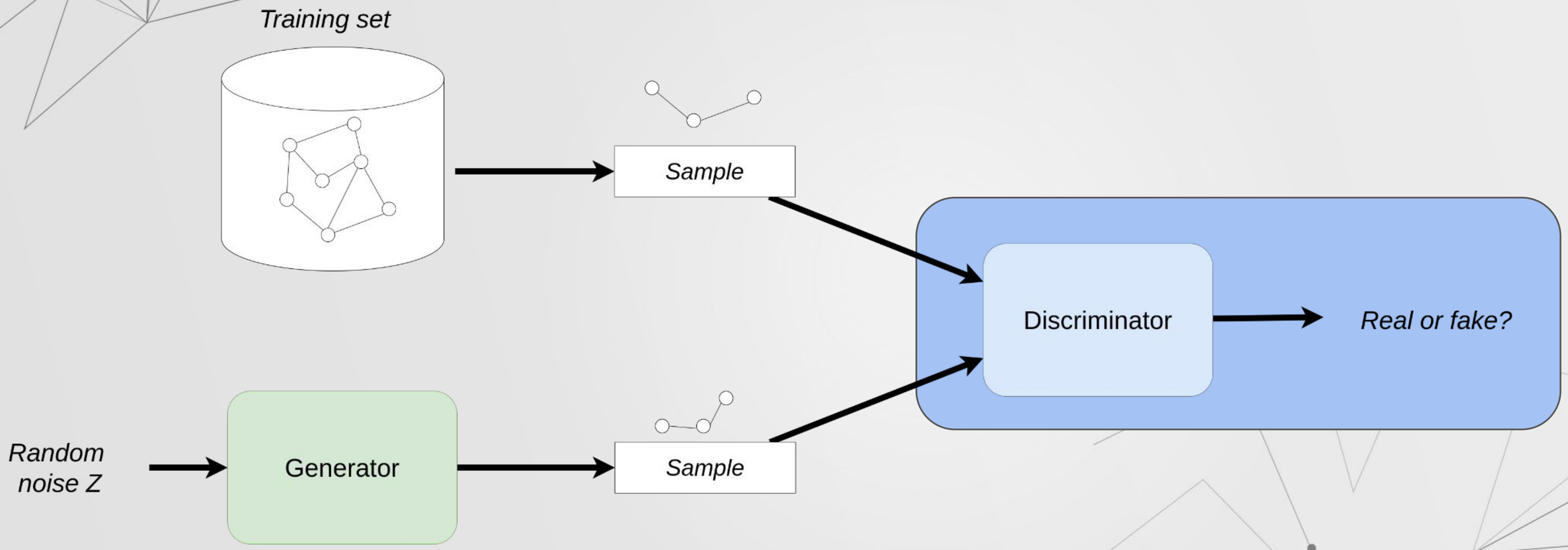


V_1
V_2
V_1
V_2
V_T
V_1
V_2
V_T
W_{up}
W_{up}
W_{up}
O_0
O_1
O_T
C_0
C_2
C_{T-1}
C_T
Z
LSTM CELL
LSTM CELL
LSTM CELL
H_0
H_2
H_{T-1}
H_T
W_{down}
W_{down}
W_{down}

03 NetGAN


Training set
Sample
Discriminator
Real or fake?
Random noise Z
Generator
Sample

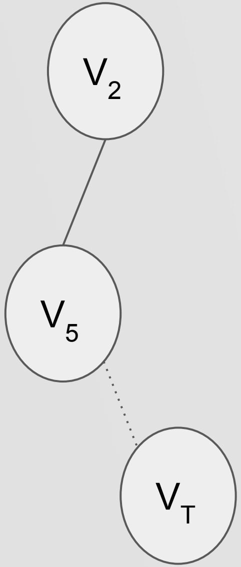
03 NetGAN



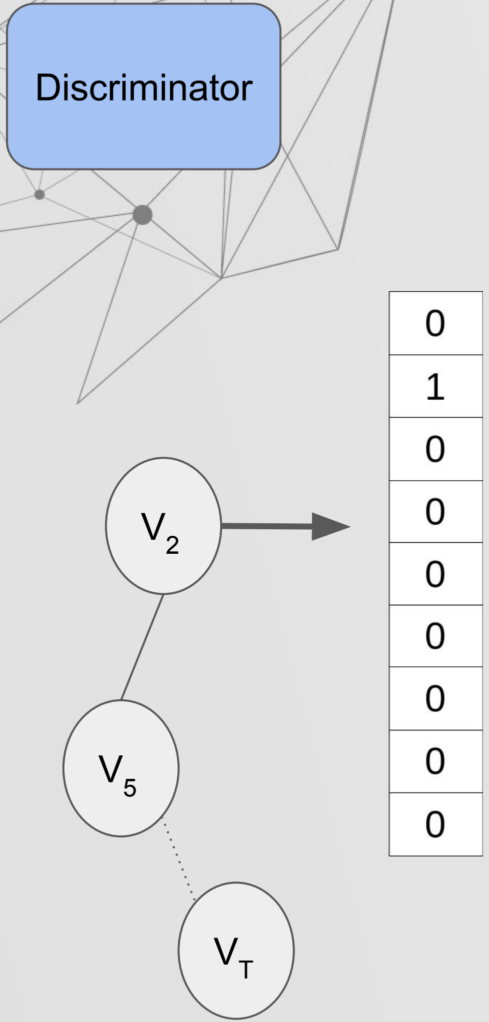
Training set
Sample
Discriminator
Real or fake?
Random noise Z
Generator
Sample

03 NetGAN

Discriminator

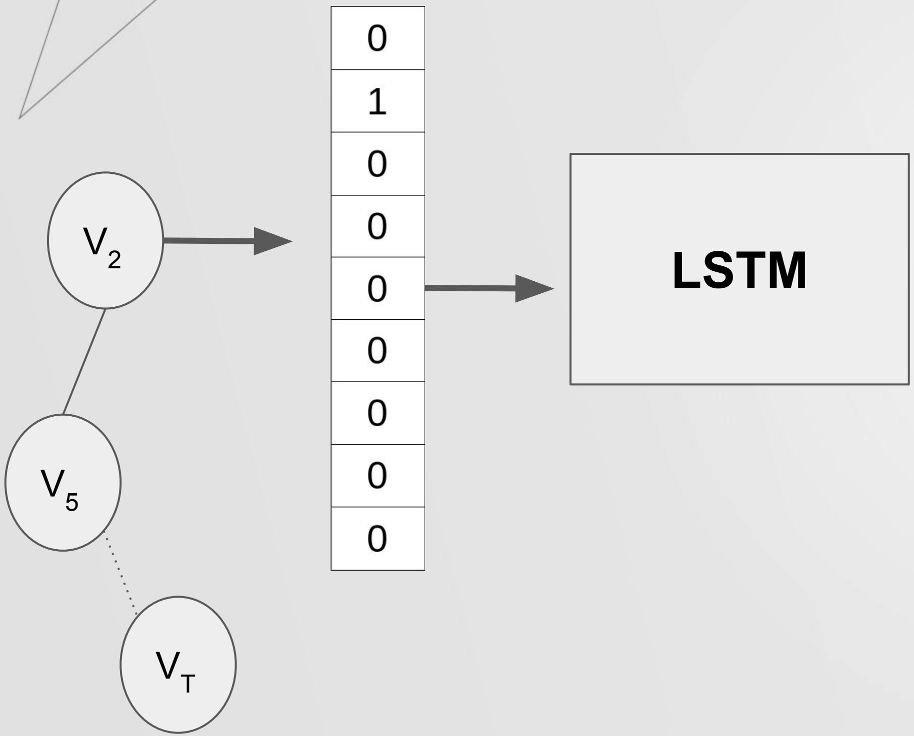


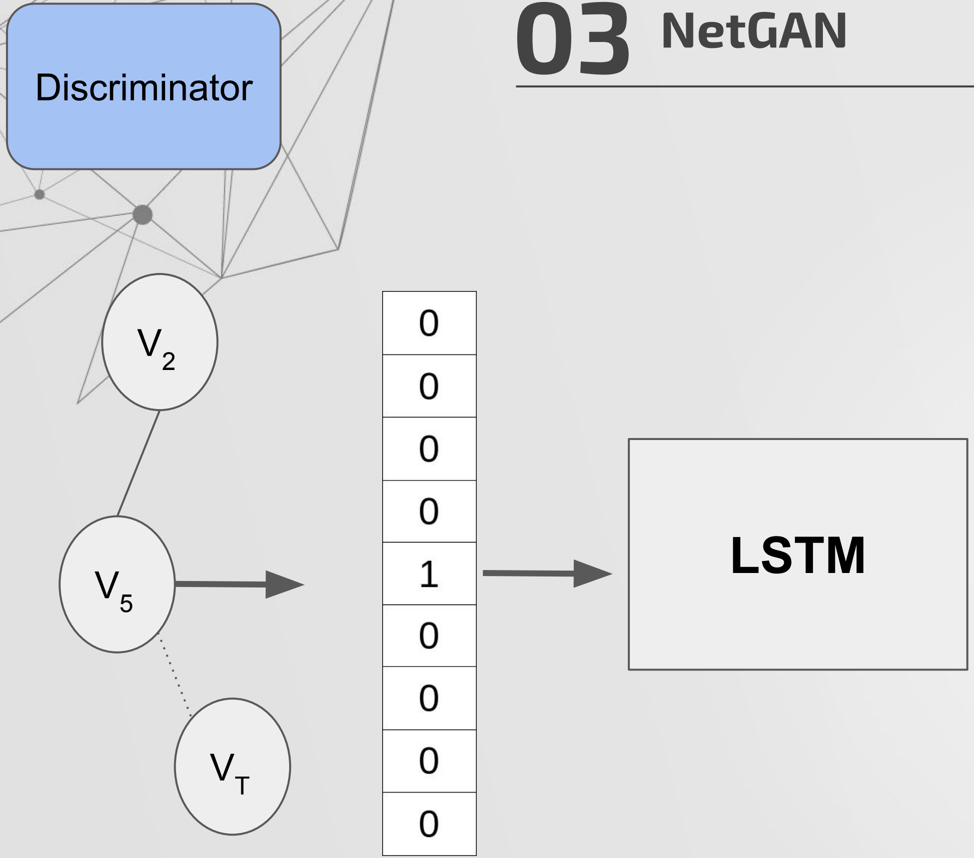
03 NetGAN



Discriminator

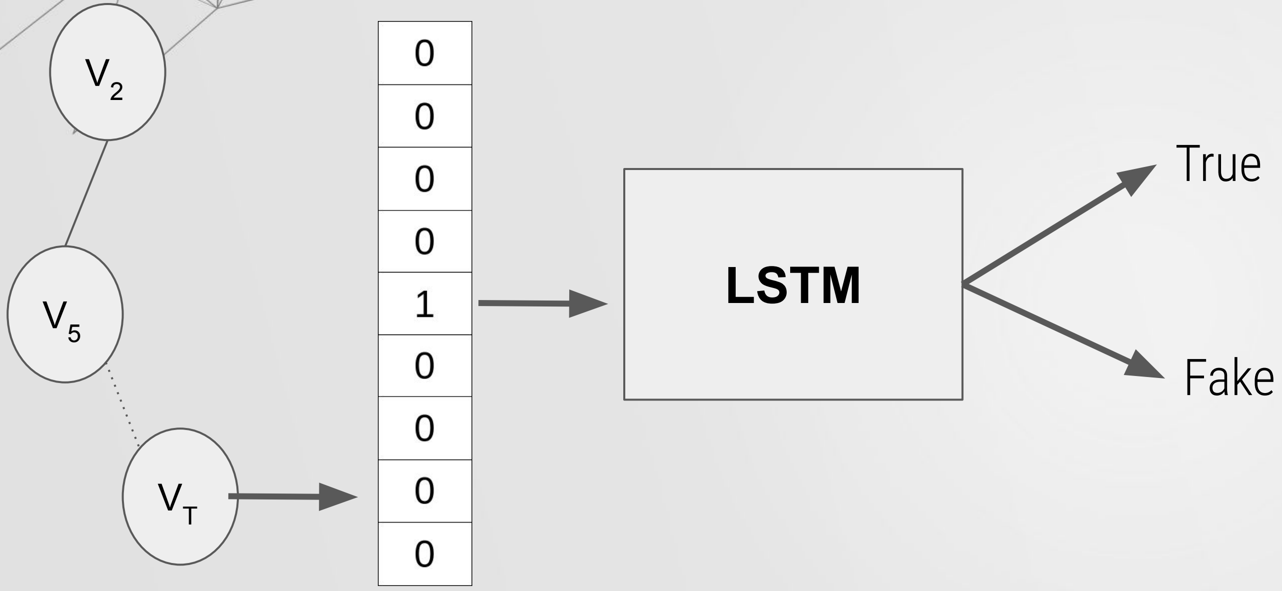
03 NetGAN



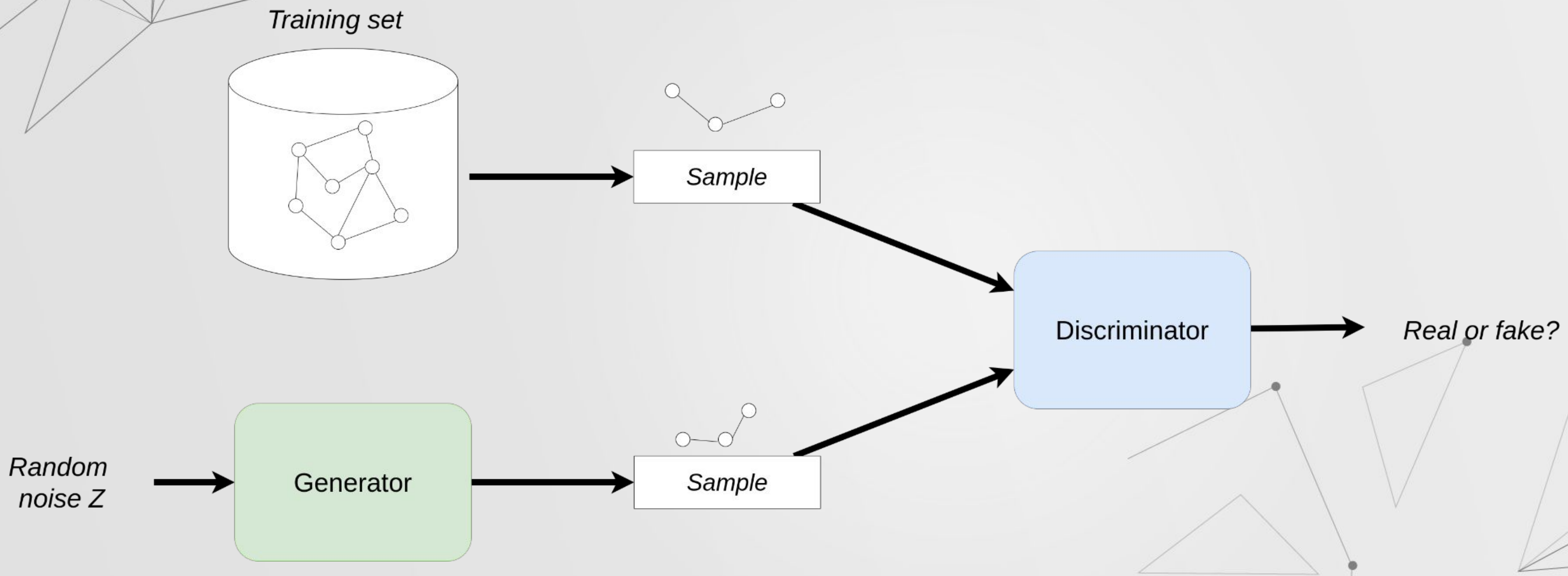

Discriminator
V2
V5
VT
0
0
0
0
1
0
0
0
0
LSTM

03 NetGAN

Discriminator

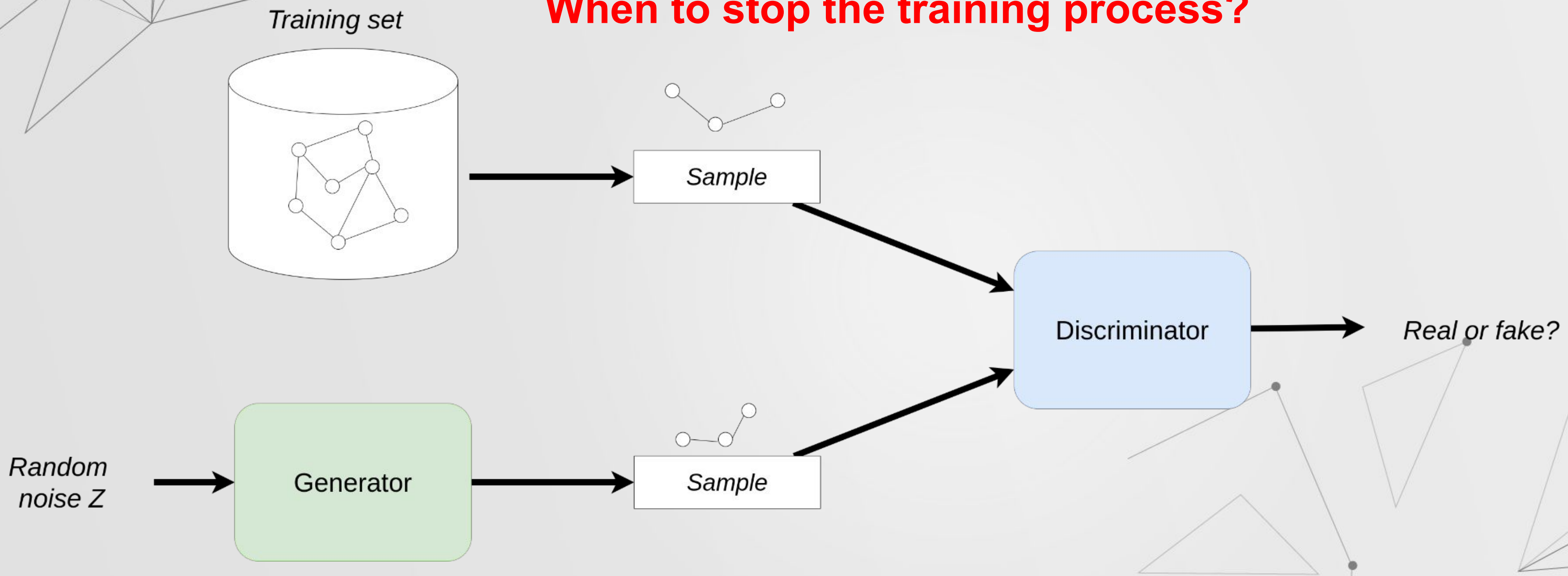


03 NetGAN


Training set
Sample
Discriminator
Real or fake?
Random
noise Z
Generator
Sample

03 NetGAN

When to stop the training process?



When to stop the training process?

EO-Criterion & VAL-Criterion

When to stop the training process?

<u>EO-Criterion</u> & VAL-Criterion

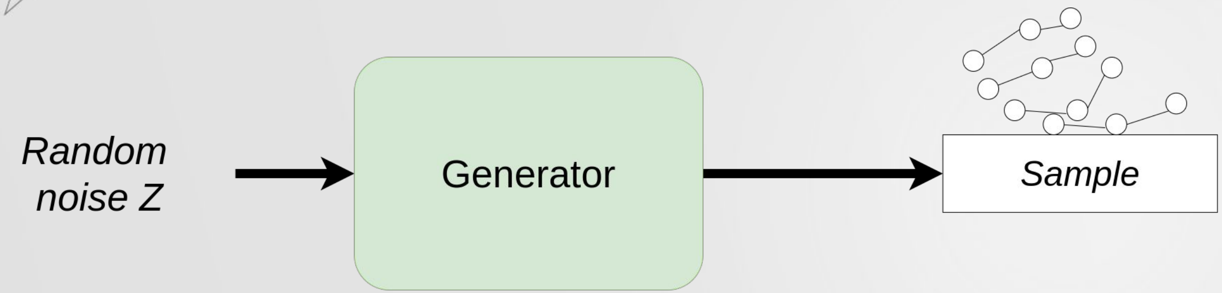
When to stop the training process?

EO-Criterion:

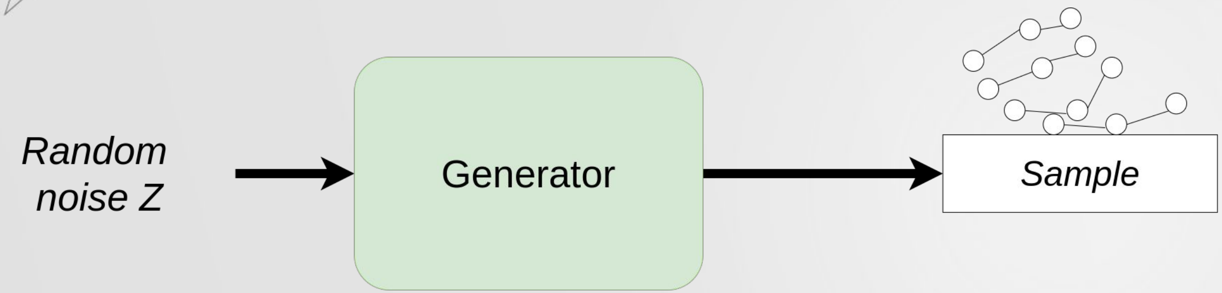
Stop the training process, when the input graph and the generated graph has an **edge overlapping** specified by the user.

How to build the graph?

How to build the graph?



How to build the graph?



0	10	2	0	1
..	..	..	..	2
..	..	..	..	..
..	..	..	..	..
..	..	..	..	..

0	10	2	0	1
..	..	..	..	2
..	..	..	..	..
..	..	..	..	..
..	..	..	..	..

How to build the graph?

1. Symmetrize $$s_{i,j} = s_{j,i} = max(s_{i,j}, s_{j,i})$$

03 NetGAN

0	10	2	0	1
..	..	..	..	2
..	..	..	..	..
..	..	..	..	..
..	..	..	..	..

How to build the graph?

1. Symmetrize $$s_{i,j} = s_{j,i} = max(s_{i,j}, s_{j,i})$$
2. Ensure every node i has at least one edge $$p_{i,j} = \frac{s_{i,j}}{\sum_v s_{i,v}}$$

0	10	2	0	1
..	..	..	..	2
..	..	..	..	..
..	..	..	..	..
..	..	..	..	..

How to build the graph?

1. Symmetrize $s_{i,j} = s_{j,i} = max(s_{i,j}, s_{j,i})$
2. Ensure every node i has at least one edge $p_{i,j} = \frac{s_{i,j}}{\sum_v s_{i,v}}$
3. Continue sampling edges with probability $p_{i,j} = \frac{s_{i,j}}{\sum_{u,v} s_{u,v}}$

04 NetGAN practice

Jupyter Notebook

Code:
https://github.com/mmiller96/netgan_pytorch