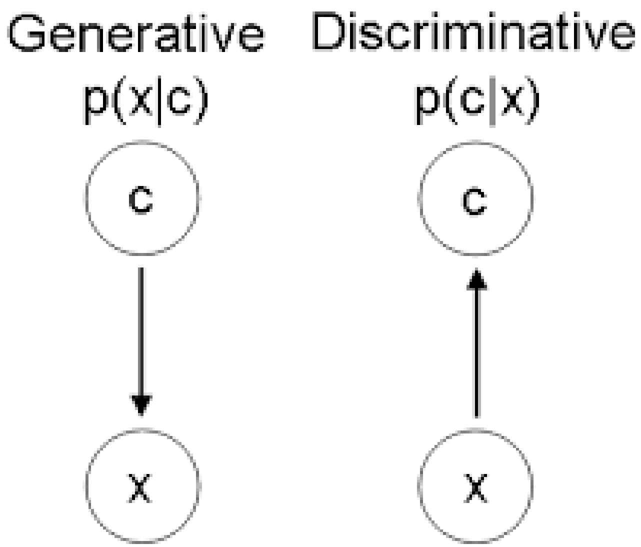
# Generative Adversarial Network (GAN)

# Two forms of Machine Learning

- **Discriminative**
- **Generative**



# Language Model

Given a language, **Language Model** estimate the probability of a word sequence occurring in the language.

$L$: Target language, say English.

$\mathrm{V}$: Set of vocabularies in the language L.

$S = \{w_1, w_2, \ldots, w_m \, | w_i \epsilon, V\}$: a word sequence

Goal: Estimate the $\Pr(S)$that S is a valid sentence in L

# Language Model

$$\Pr(S) = \Pr(w_1, w_2, \dots, w_m)$$

Apply chain rule.

$$\Pr(S) = \Pr(w_1, w_2, \dots, w_m)$$
$$= \Pr(w_m | w_1, w_2, \dots, w_{m-1}) \,.\, \Pr(w_1, w_2, \dots, w_{m-1})$$
$$=$$
$$\Pr(w_m | w_1, w_2, \dots, w_{m-1}) \,.\, \Pr(w_{m-1} | w_1, w_2, \dots, w_{m-2}) \,.\, \Pr(w_1, w_2, \dots, w_{m-2})$$
$$= \Pr(w_m | w_1, w_2, \dots, w_{m-1}) \,.\, \Pr(w_{m-1} | w_1, w_2, \dots, w_{m-2}) \dots \Pr(w_2 | w_1) \Pr(w_1)$$

$$\Pr(w_1, w_2, \dots, w_m) = \prod_i \Pr(w_i | w_1 w_2 .. w_{i-1})$$

# Language Model – Two Common Tasks

Sentence generation: Estimate $\Pr(S) = \Pr(w_1, w_2, \ldots, w_m)$

P("it is a nice movie") = P(it) × P(is|it) × P(a|it is) × P(nice|it is a) × P(movie|it is a nice)

# Language Model – Two Common Tasks

Sentence generation: Estimate $\Pr(S) = \Pr(w_1, w_2, \ldots, w_m)$

Next word prediction: Estimate $\Pr(w_m | w_1, w_2, \ldots, w_{m-1})$

P("it is a nice movie") = P(it) × P(is|it) × P(a|it is) × P(nice|it is a) × P(movie|it is a nice)

# Language Model – Challenge

Support of $\{w_1, w_2, \ldots, w_m\}$ may be small for large m i.e., there may not be enough data for such large sequence.

# Language Model – Challenge

Restrict the window size

$$P(movie|it\ is\ a\ nice) \approx P(movie|nice)$$

Or,

$$P(movie|it\ is\ a\ nice) \approx P(movie|a\ nice)$$

# Language Model – Challenge

Restrict the window size.

$$P(movie|it\ is\ a\ nice) \approx P(movie|nice)$$

Or,

$$P(movie|it\ is\ a\ nice) \approx P(movie|a\ nice)$$

So, restrict to a small window of size k.

$$P(w_1, w_2, \dots, w_m)$$
$$= \Pr(w_m|w_{m-k-1}, w_{m-k-2}, \dots, w_{m-1}) \,.\, \Pr(w_{m-1}|w_{m-k}, w_{m-k-2}, \dots, w_{m-2}) \dots \Pr(w_1|w_2) \Pr(w_1)$$

# Language Model – n gram

**Unigram (k=1):** $P(movie|it\ is\ a\ nice) \approx P(movie)$

$P(w_m|w_1 w_2 \cdots w_{m-1}) \approx P(w_m)$

Bigram (k=2): $P(movie|it\ is\ a\ nice) \approx P(movie|nice)$

$P(w_m|w_1 w_2 \cdots w_{m-1}) \sim P(w_m|w_{m-1})$

Trigram(k=3): $P(movie|it\ is\ a\ nice) \approx P(movie|a\ nice)$

$P(w_m|w_1 w_2 \cdots w_{m-1}) \sim P(w_m|w_{m-2} w_{m-1})$

n-gram (k=n)

$$P(w_1, w_2, \dots, w_m)$$
$$= \Pr(w_m|w_{m-k-1}, w_{m-k-2}, \dots, w_{m-1}) . \Pr(w_{m-1}|w_{m-k}, w_{m-k-2}, \dots, w_{m-2}) \dots \Pr(w_1|w_2) \Pr(w_1)$$
$$= \prod_i \Pr(w_i|w_{i-k-1} w_{i-k-2} .. w_{i-1})$$

# Language Generation

Consider a text corpus L in English, say, a collection of English sentences.

Let $V$ be the vocabulary set.

Let $v \subseteq V$

The sentence generated from $v$ is defined by $S = \underset{s'}{\operatorname{argmax}} P(S')$

Give $\boldsymbol{v} = \{\text{it}, \text{is}, \text{a}, \text{nice}, \text{movie}\}$ we may have various combinations such as

*It is a nice movie, nice a is it movie, movie it a is nice, …..*

$P(it\ is\ a\ nice\ movie) > P(nice\ a\ is\ it\ movie)$

# Language Model Could be built using

Statistical – we have just seen

*Neural Network – Word2Vec*

# Generative Adversarial Network

# Generative Adversarial Network

Counterfeiter

Fraud Detector

# Generative Adversarial Network

Counterfeiter

**Generator**

Fraud Detector

**Discriminator**

# Generative Adversarial Network

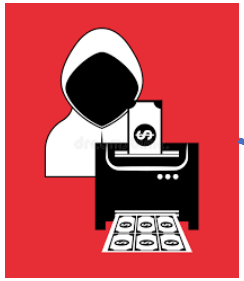
generates fake samples as real as possible and tries to fool the **Discriminator**

**Generator**

tries to detect the fake samples generated by the **Generator**

**Discriminator**

# Generative Adversarial Network



**Generator**

**Discriminator**

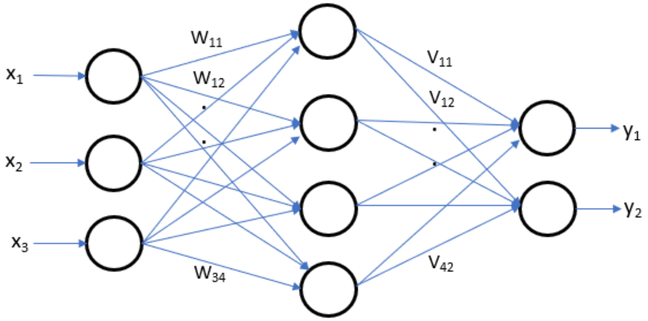
They are trained in an adversarial setting to master each other's task.

# Magic of GANs….

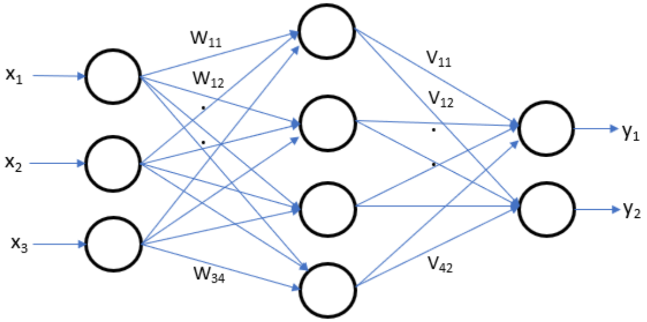
- Images generated using StyleGAN- a GAN variant
- These people don't exist in real!!!!!!
- Image from Paper '*A Style-Based Generator Architecture for Generative Adversarial Networks'*

Karras, Tero et al. "A Style-Based Generator Architecture for Generative Adversarial Networks." *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019): 4396-4405.

# GAN – both the generator and Discriminator are neural network

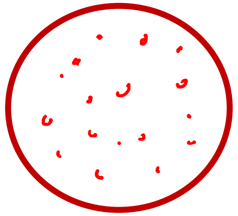



**Generator**




**Discriminator**

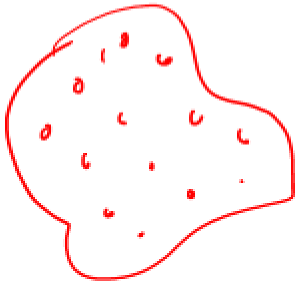
# GAN – both the generator and Discriminator are neural network



$D$

$x \sim D$

**Real Data Distribution**

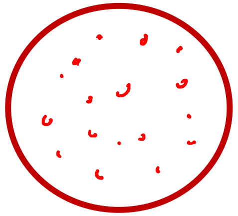


$Z$

$z \sim Z$

**Random Sample**

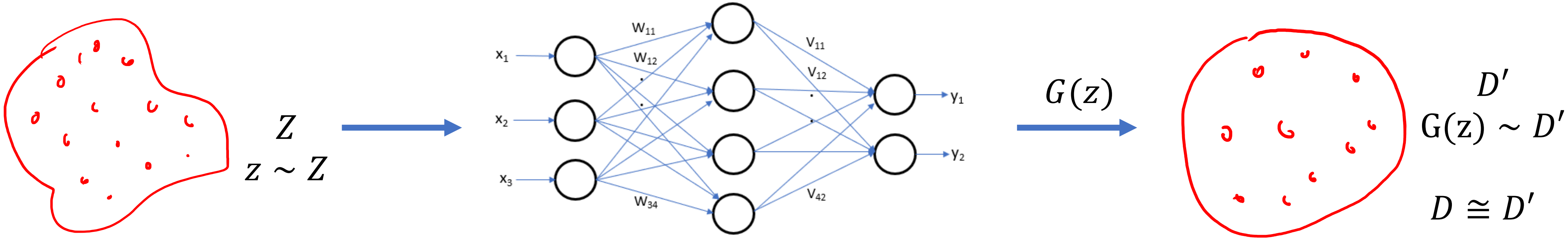
# GAN – both the generator and Discriminator are neural network



$D$

$x \sim D$

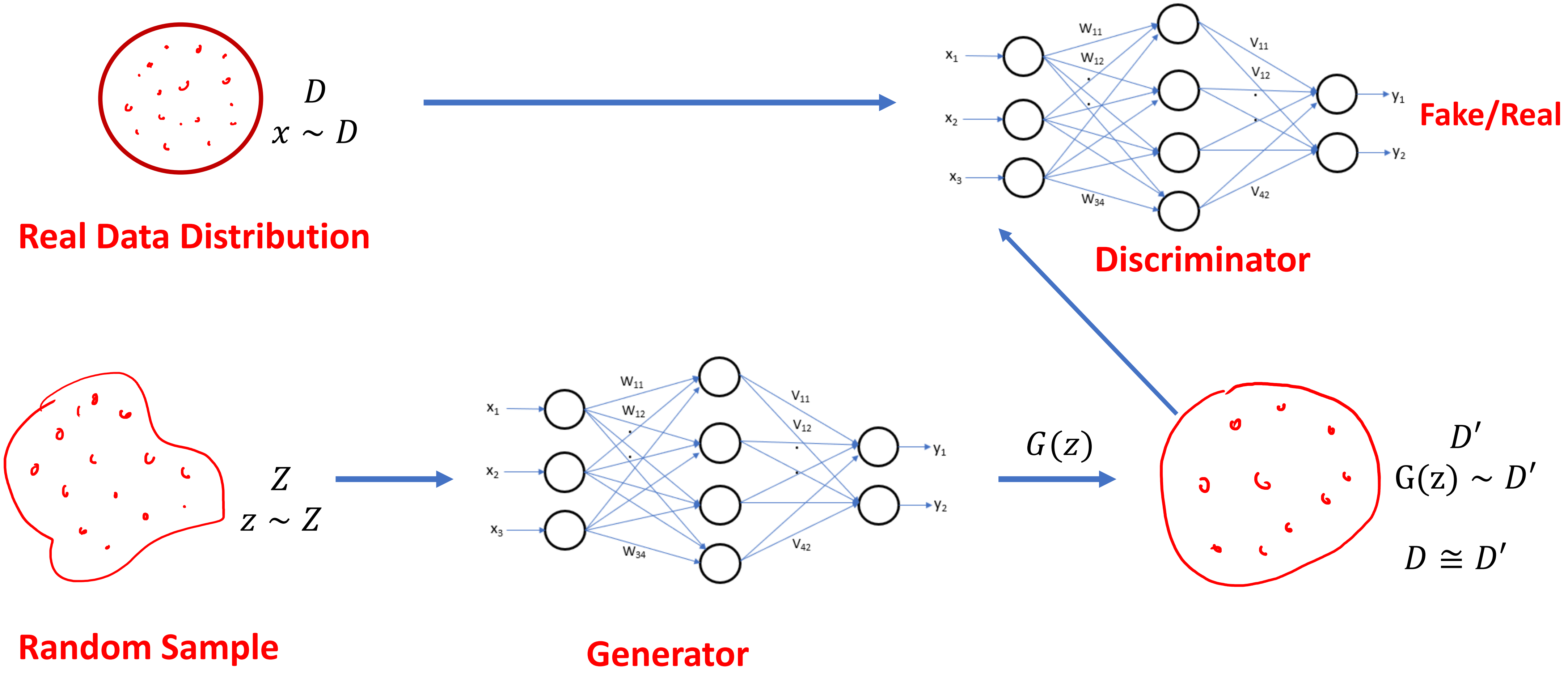
**Real Data Distribution**




**Random Sample**

**Generator**

# GAN – both the generator and Discriminator are neural network

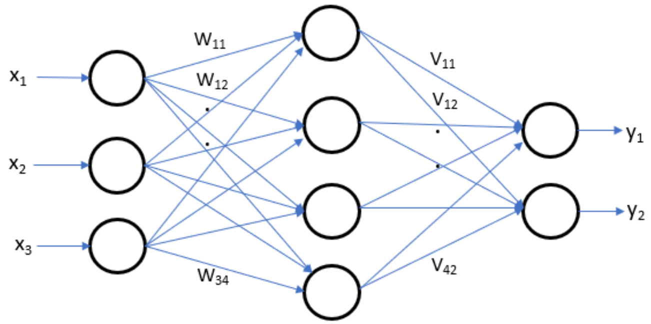


# GAN – both the generator and Discriminator are neural network

$D$

$x \sim D$

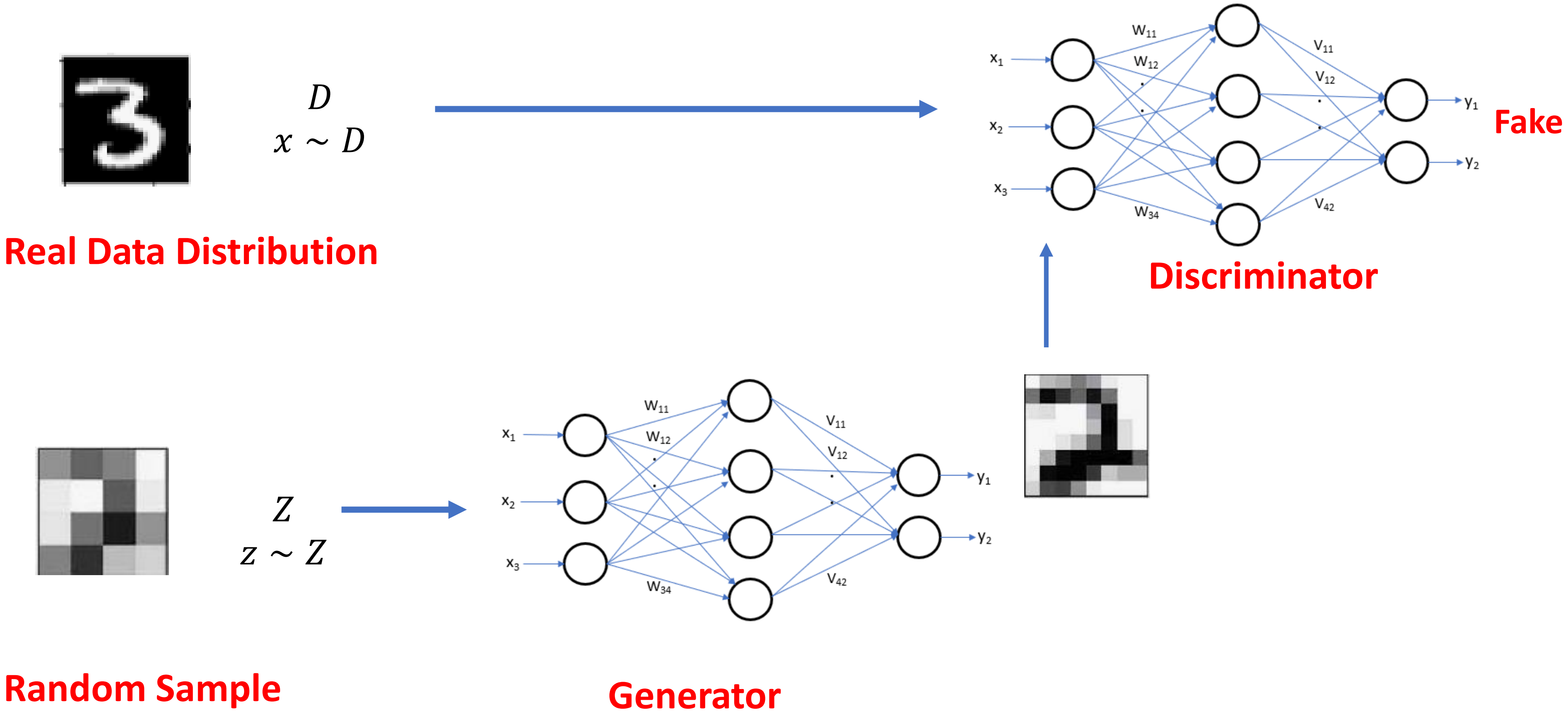
**Real Data Distribution**



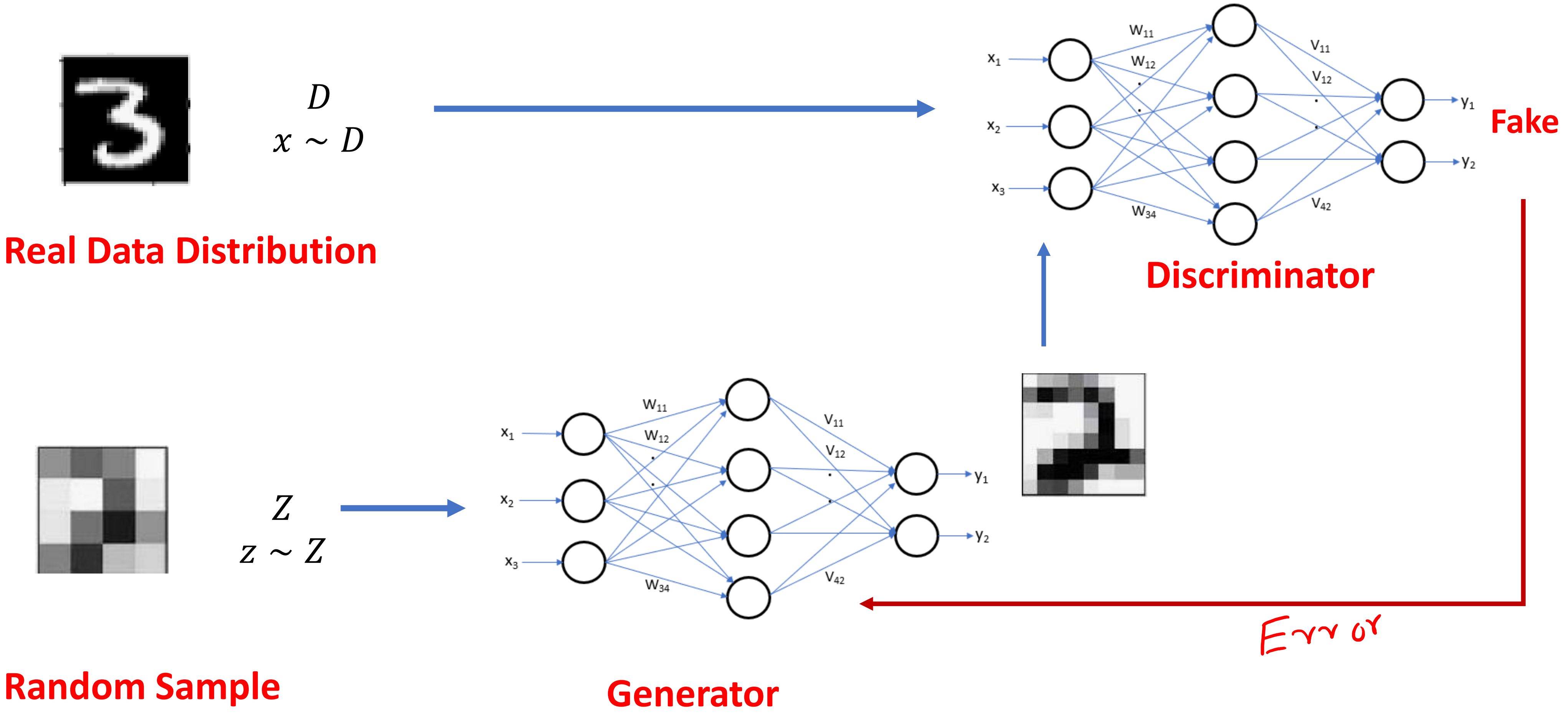

**Real**

**Discriminator**

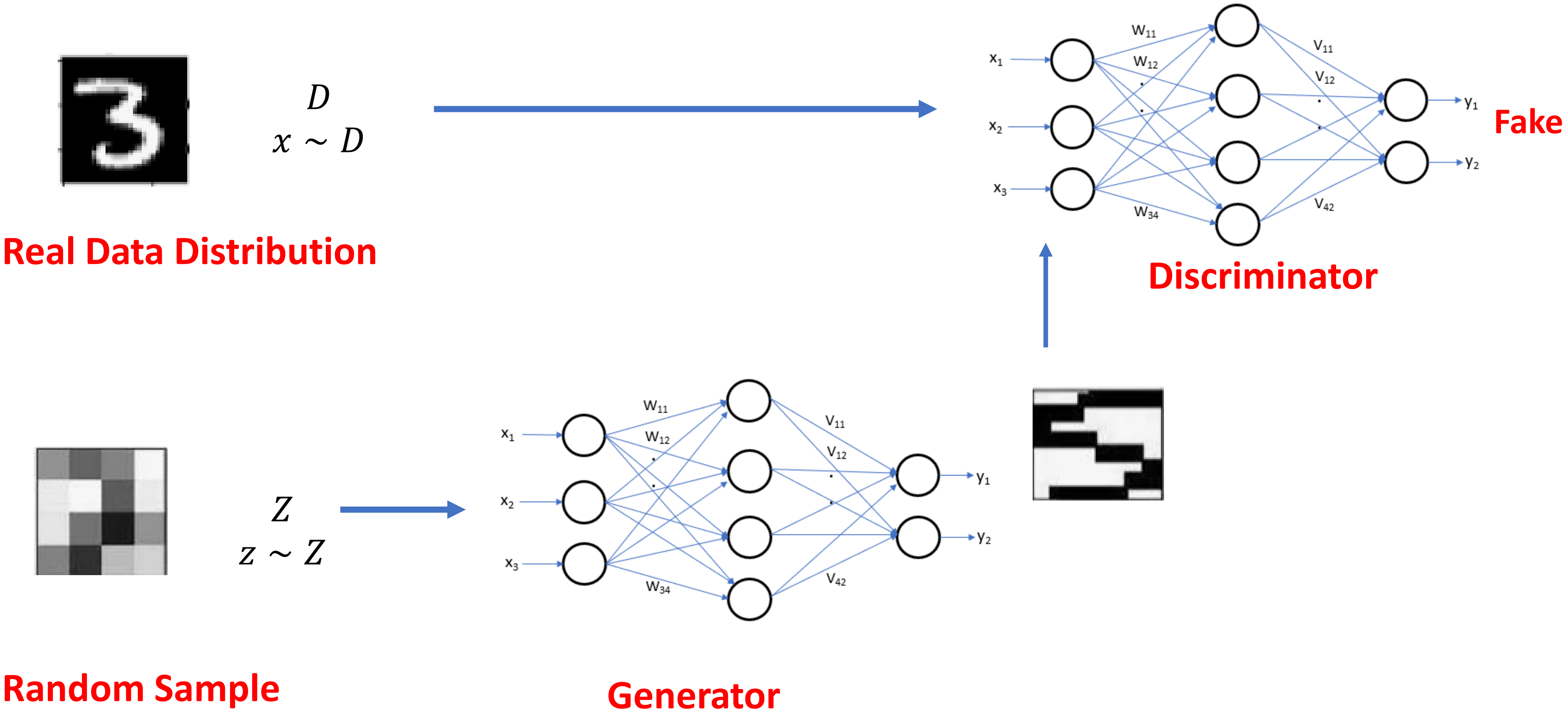
# GAN – both the generator and Discriminator are neural network



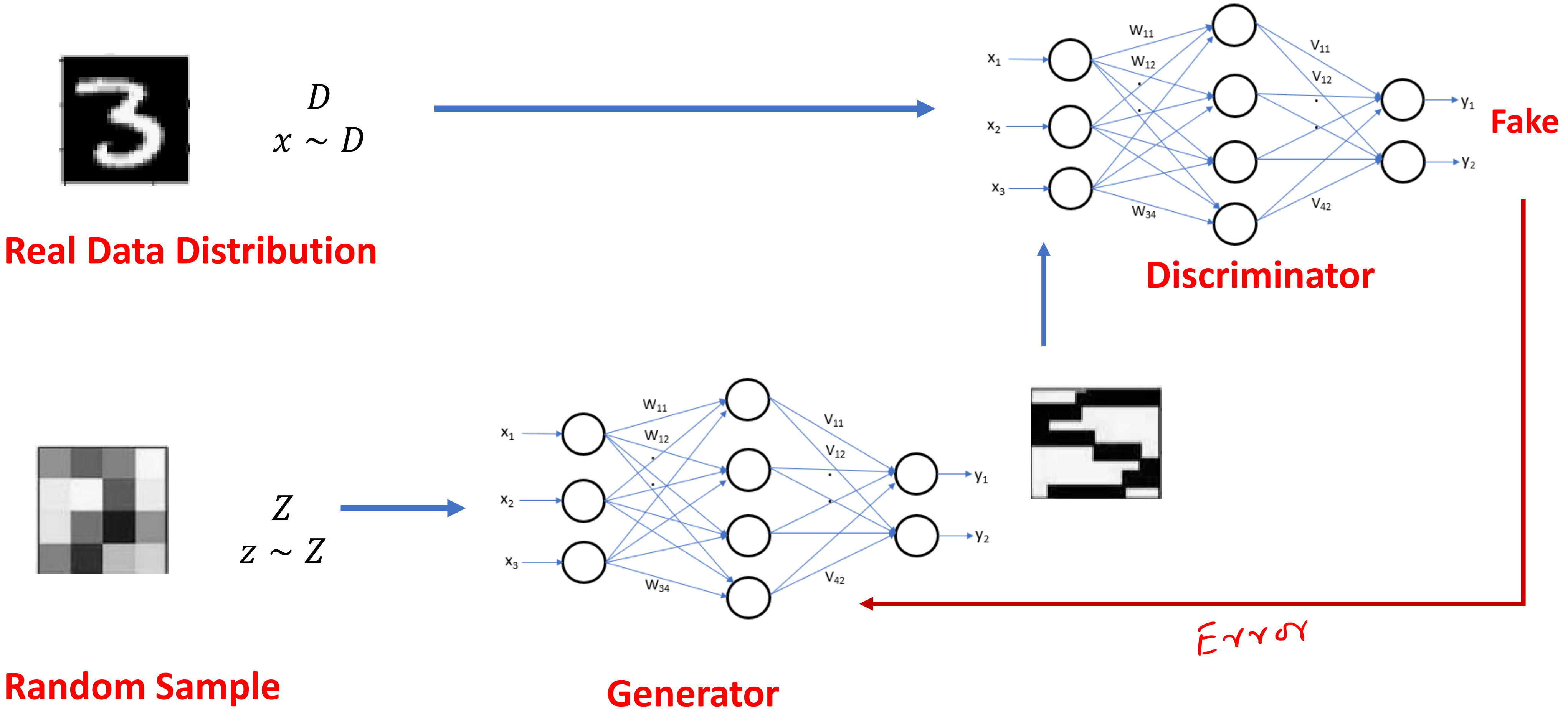
# GAN – both the generator and Discriminator are neural network



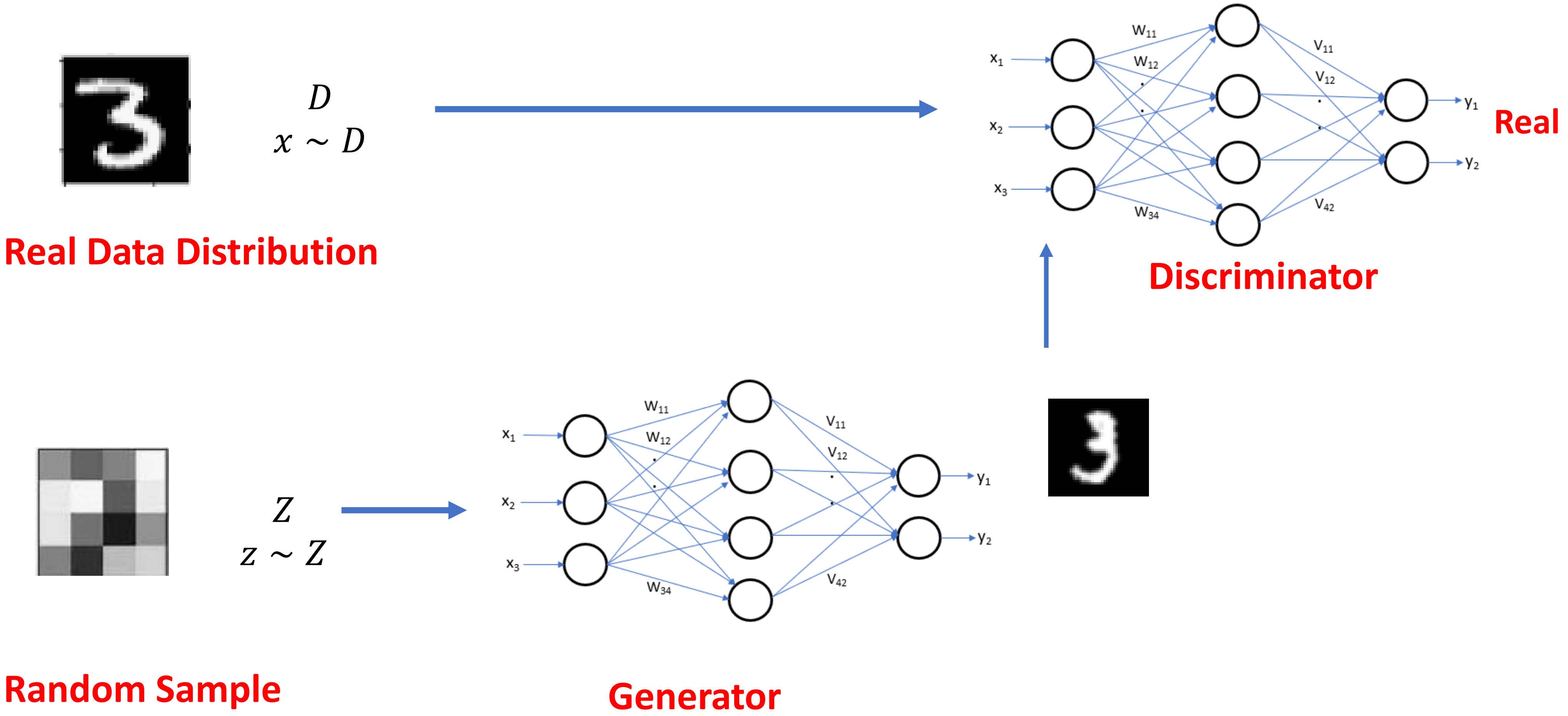
# GAN – both the generator and Discriminator are neural network



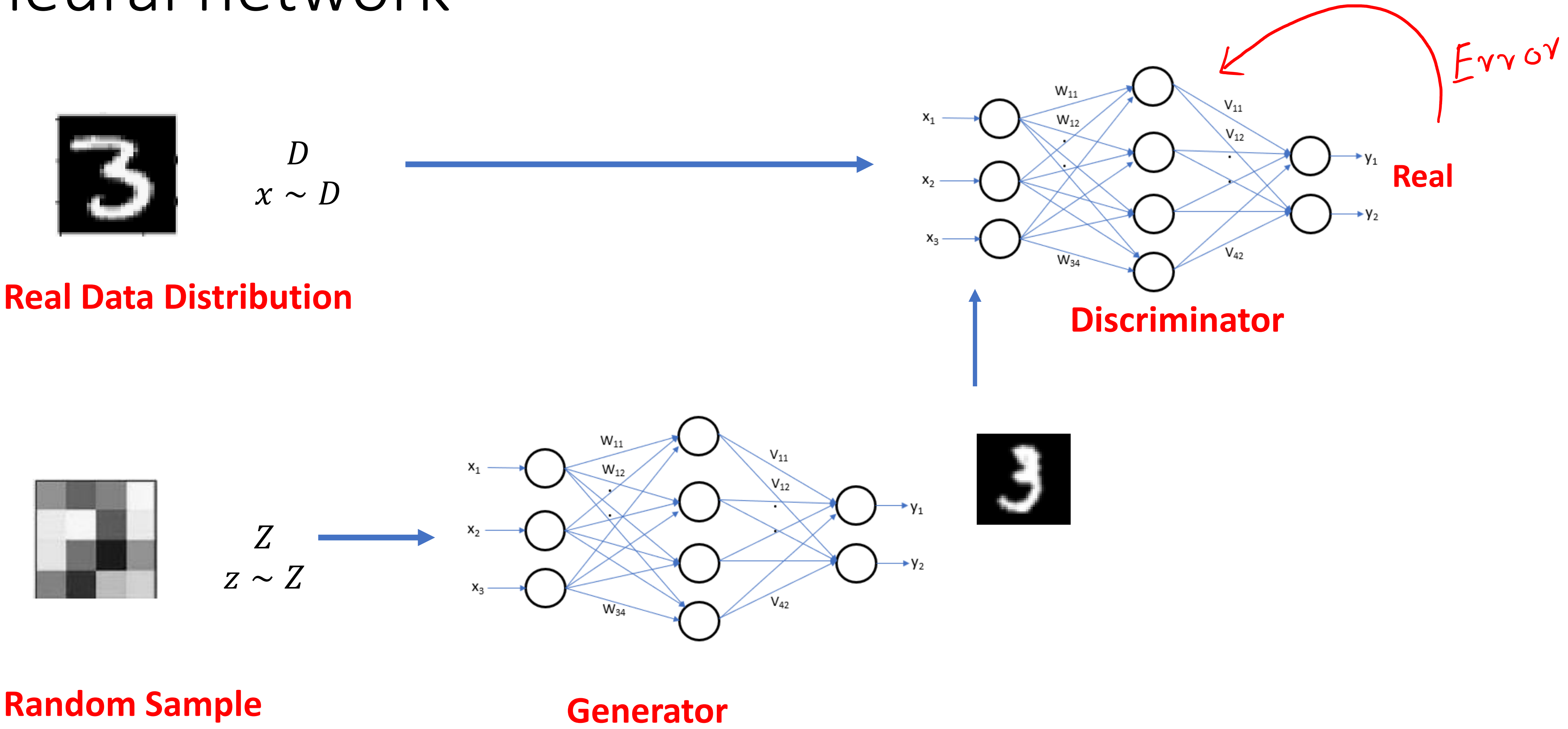
# GAN – both the generator and Discriminator are neural network



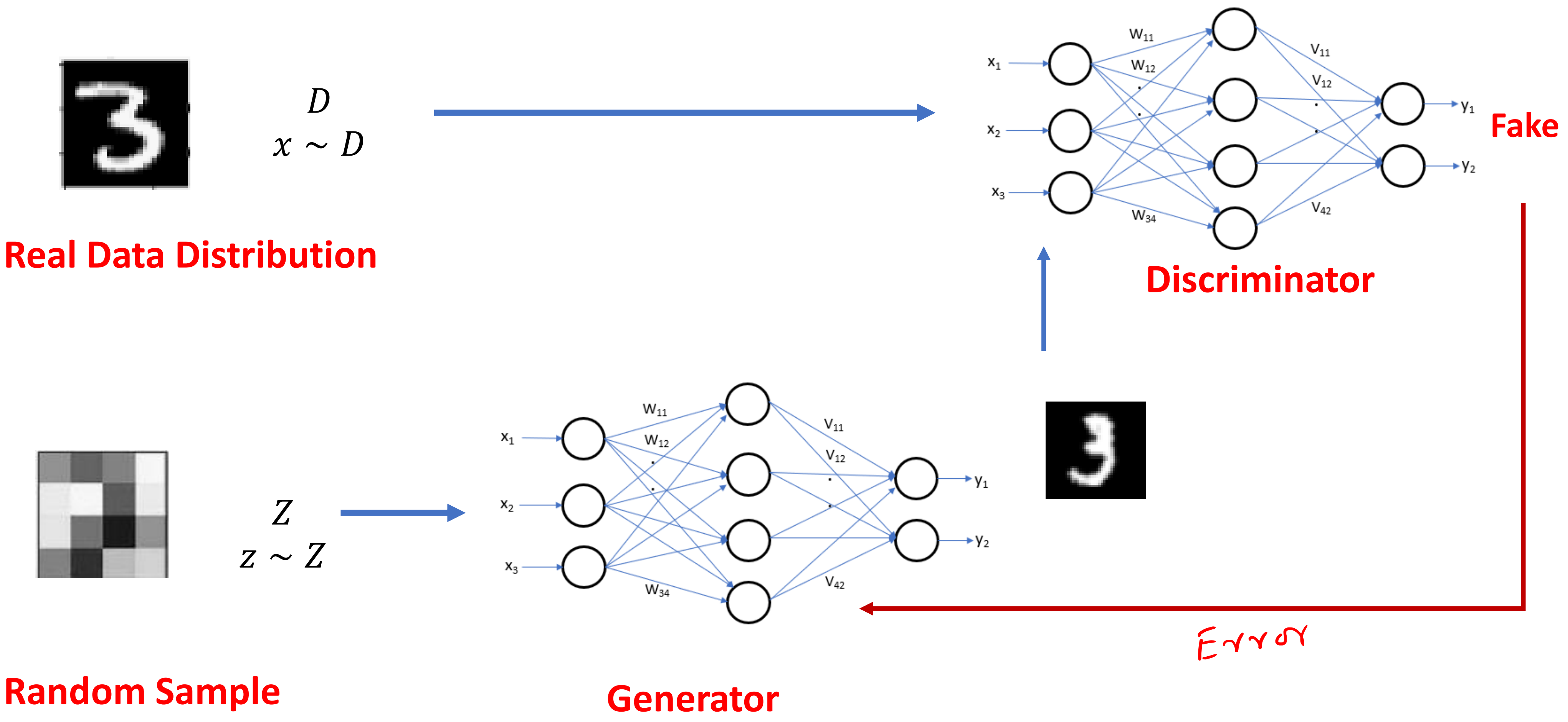
# GAN – both the generator and Discriminator are neural network



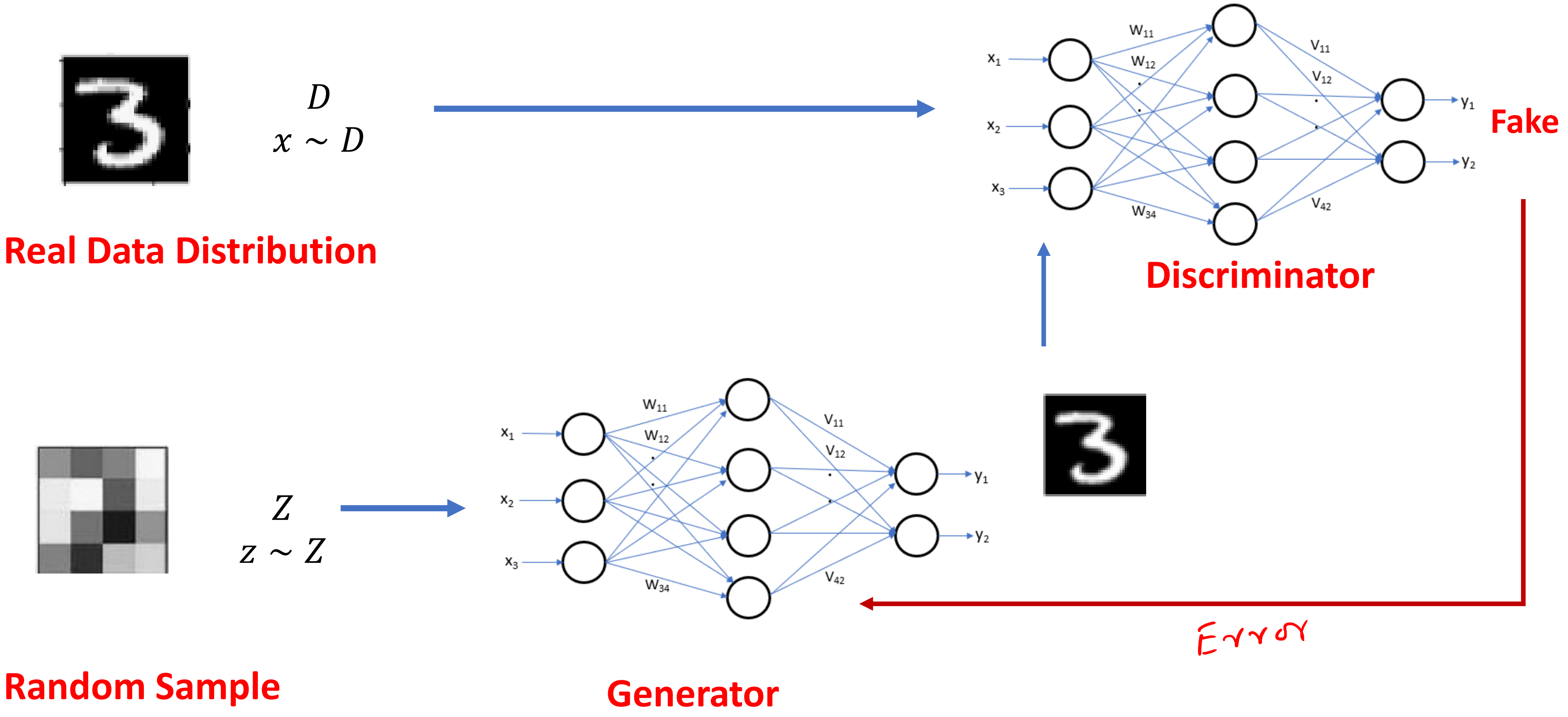
# GAN – both the generator and Discriminator are neural network



# GAN – both the generator and Discriminator are neural network

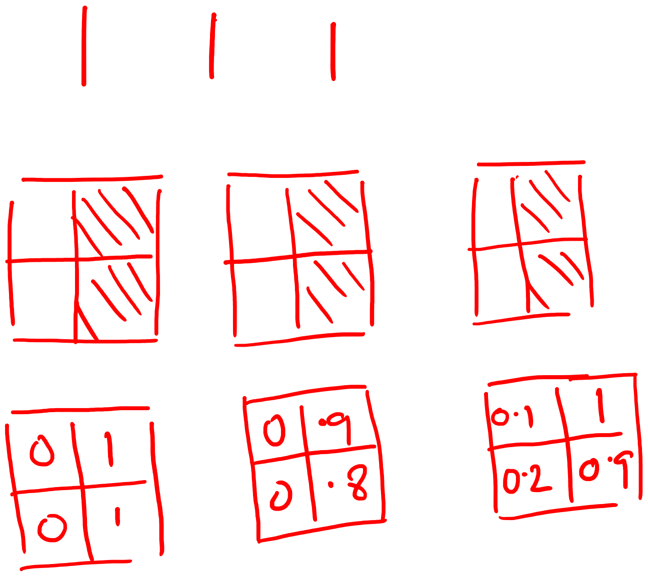


# GAN – both the generator and Discriminator are neural network

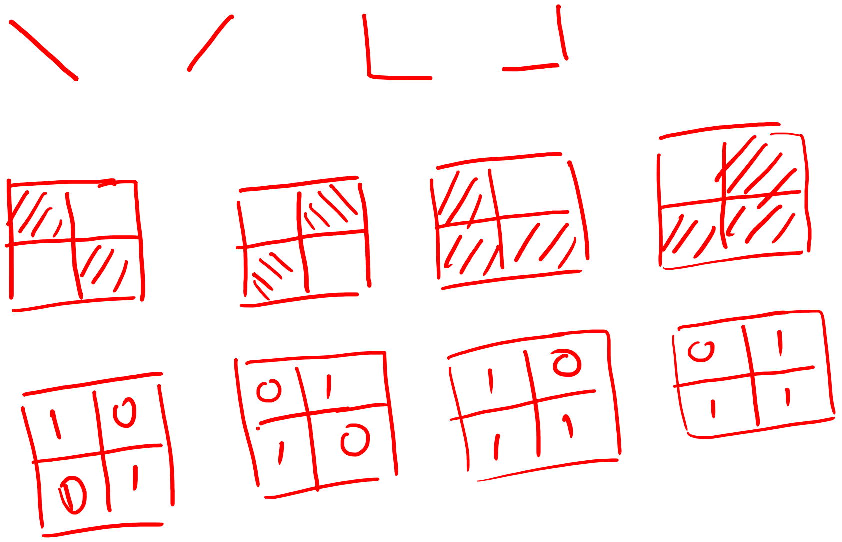


# GAN – A toy Example

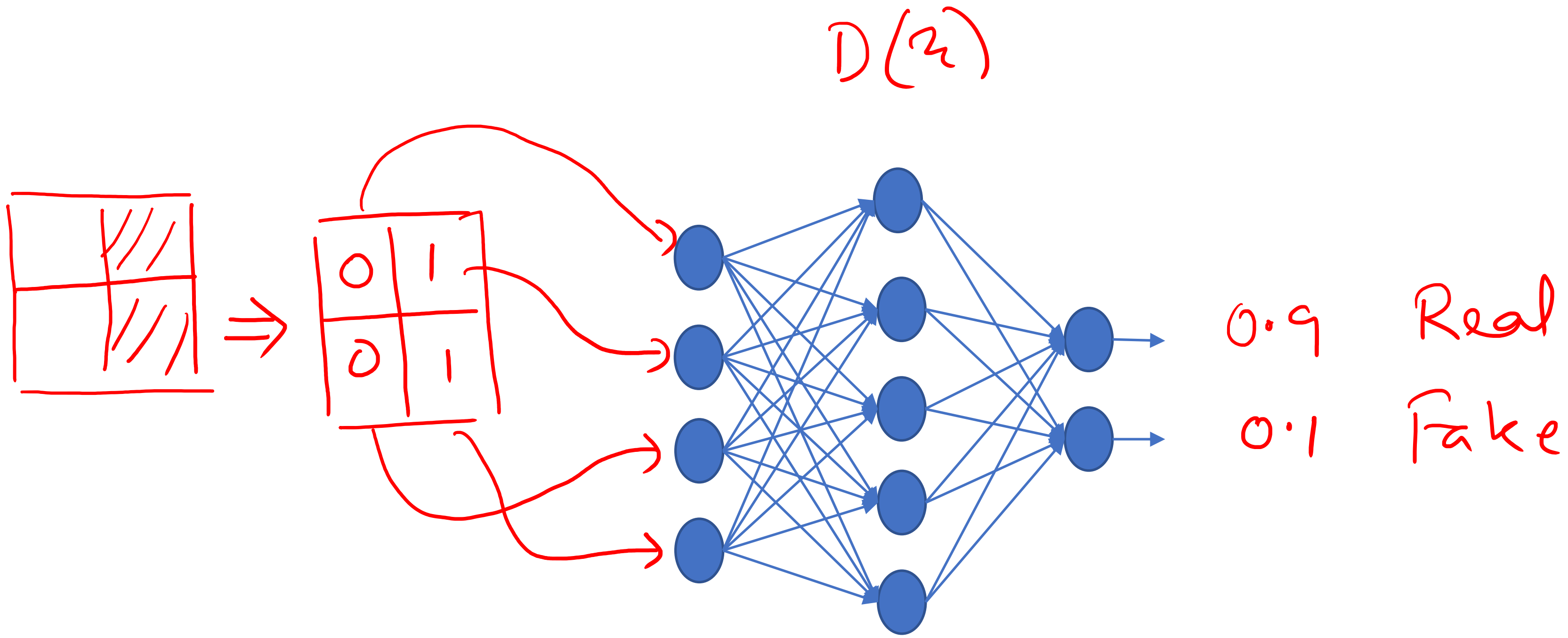
Real:



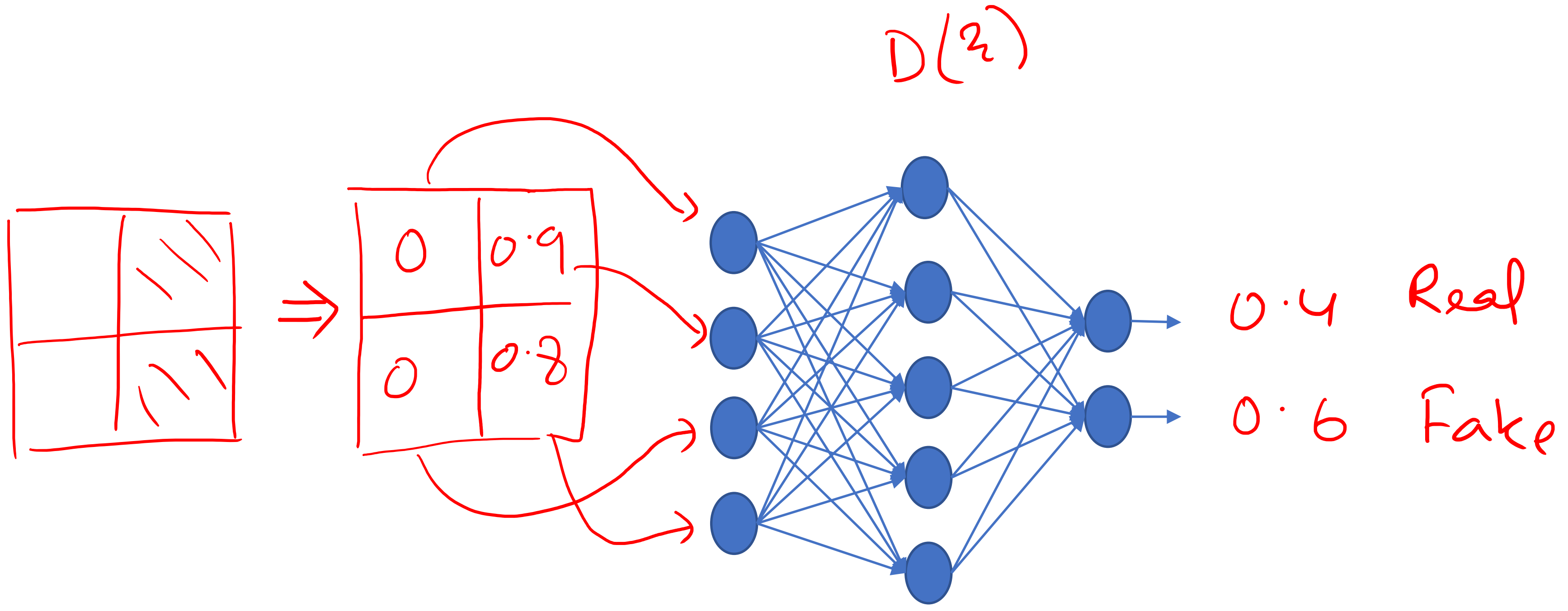

Fake:



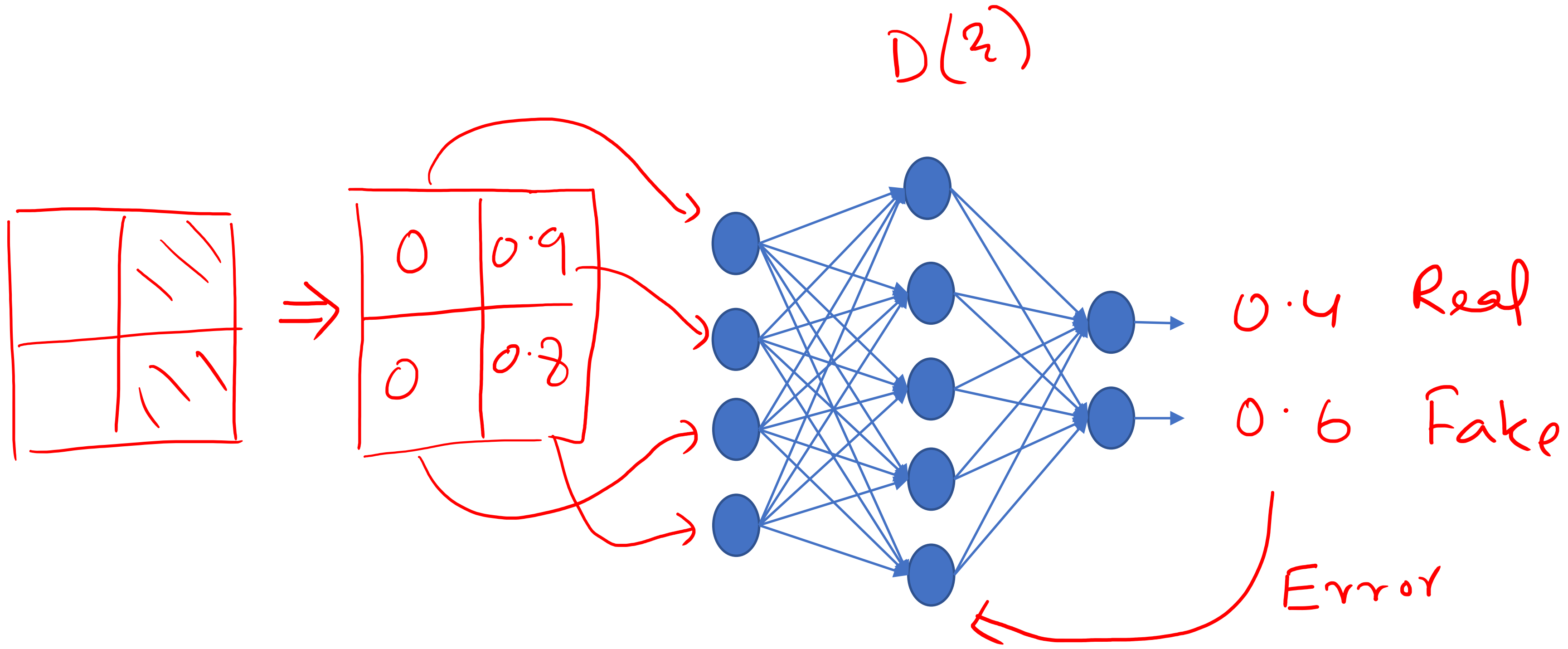
# GAN – A toy Example



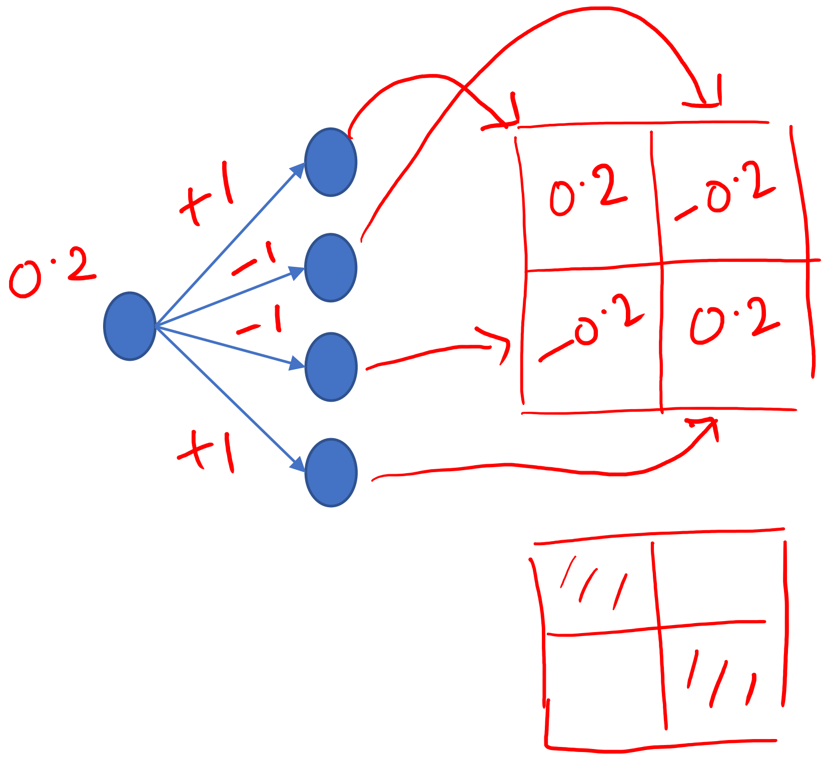
# GAN – A toy Example

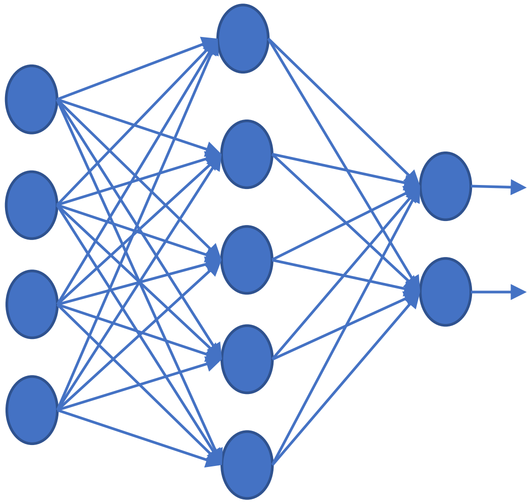


# GAN – A toy Example

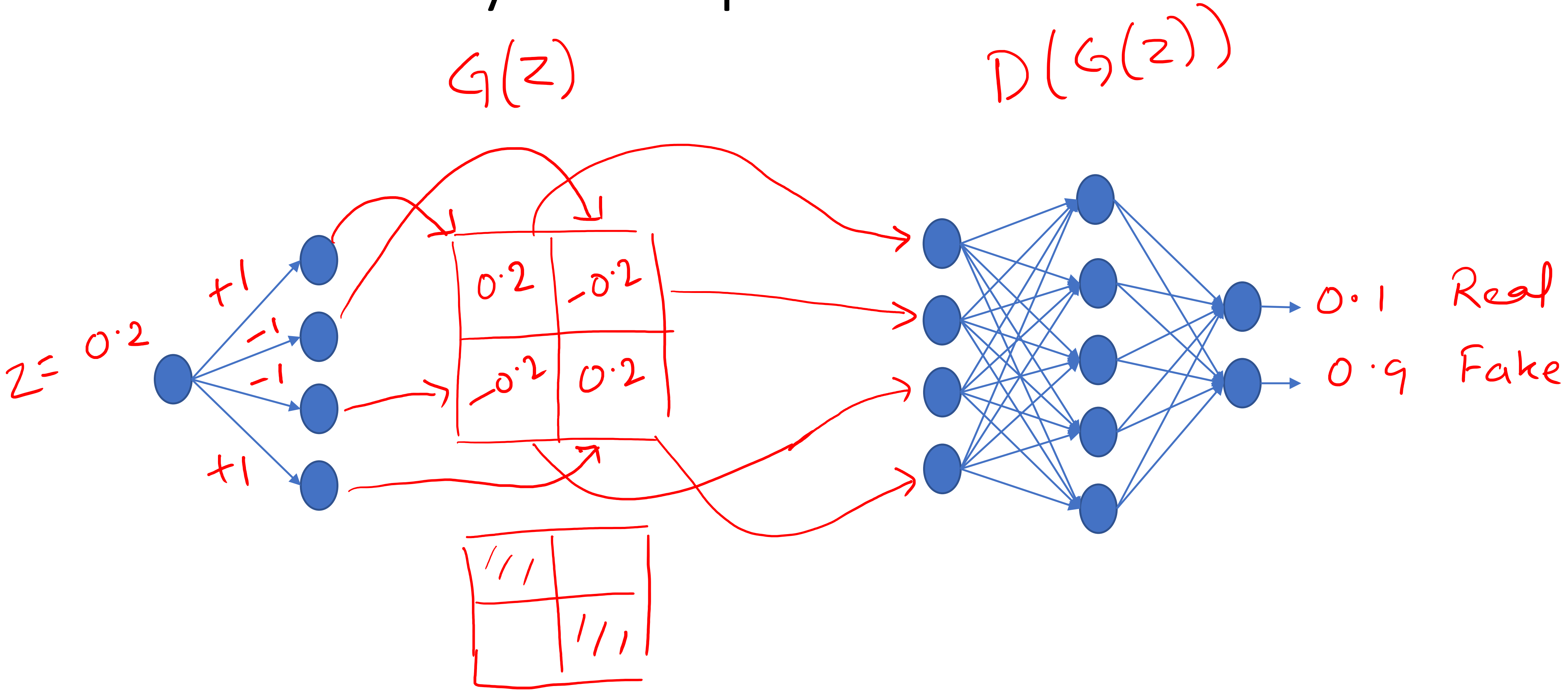


# GAN – A toy Example

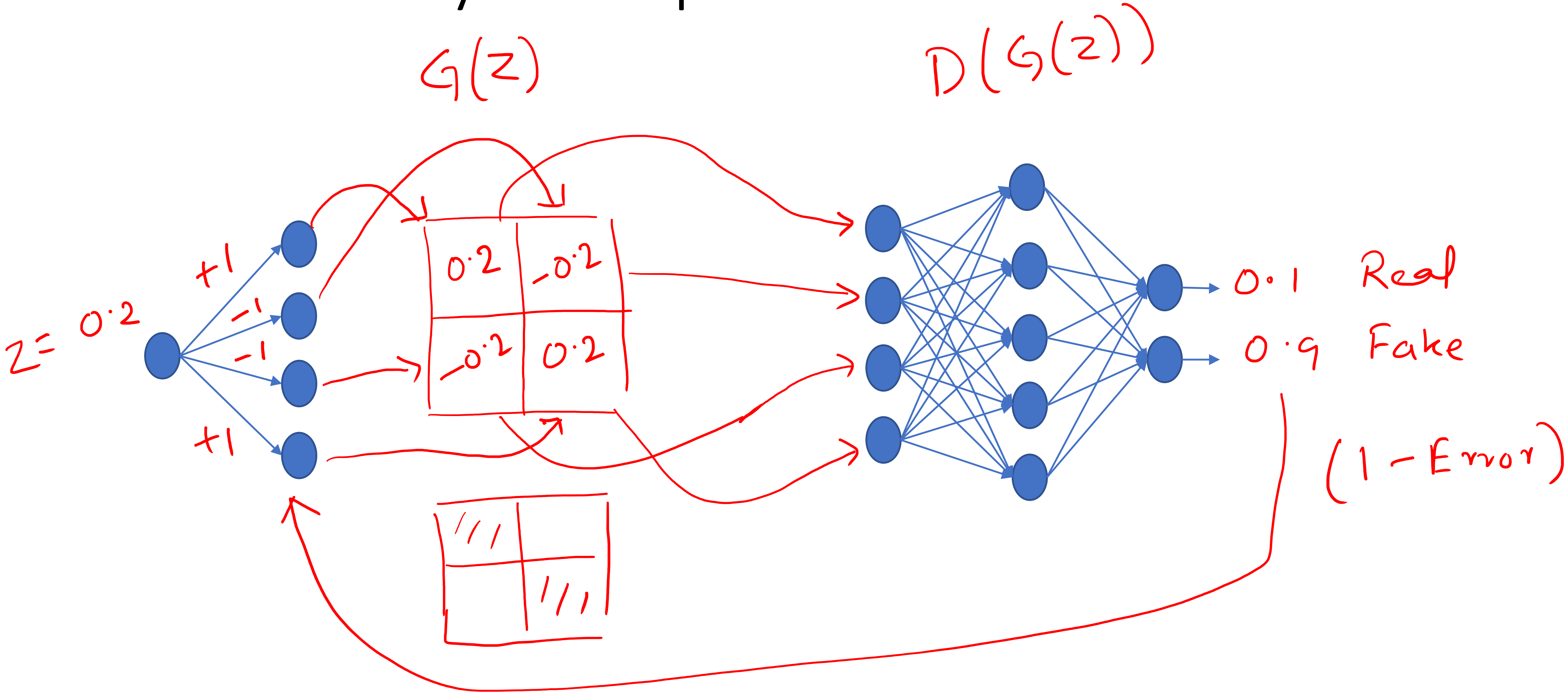


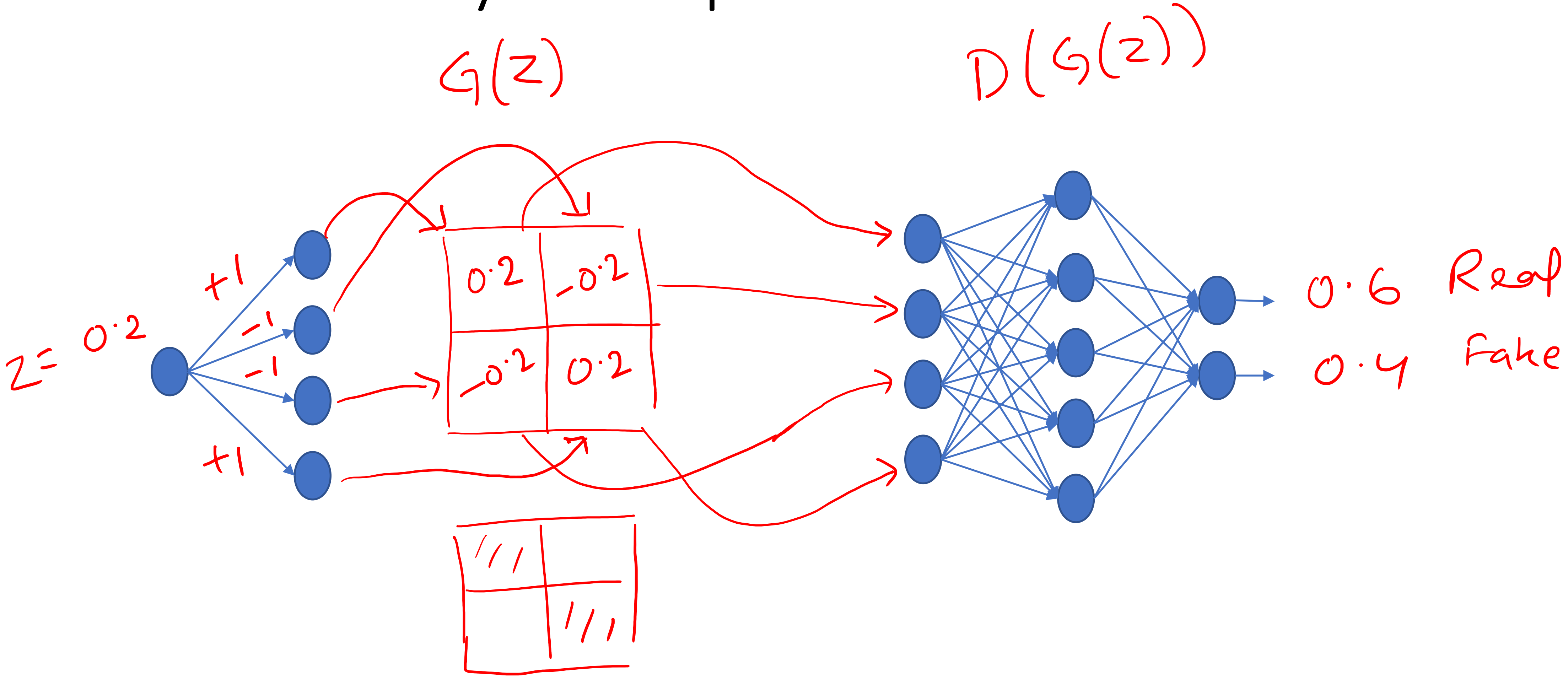
# GAN – A toy Example



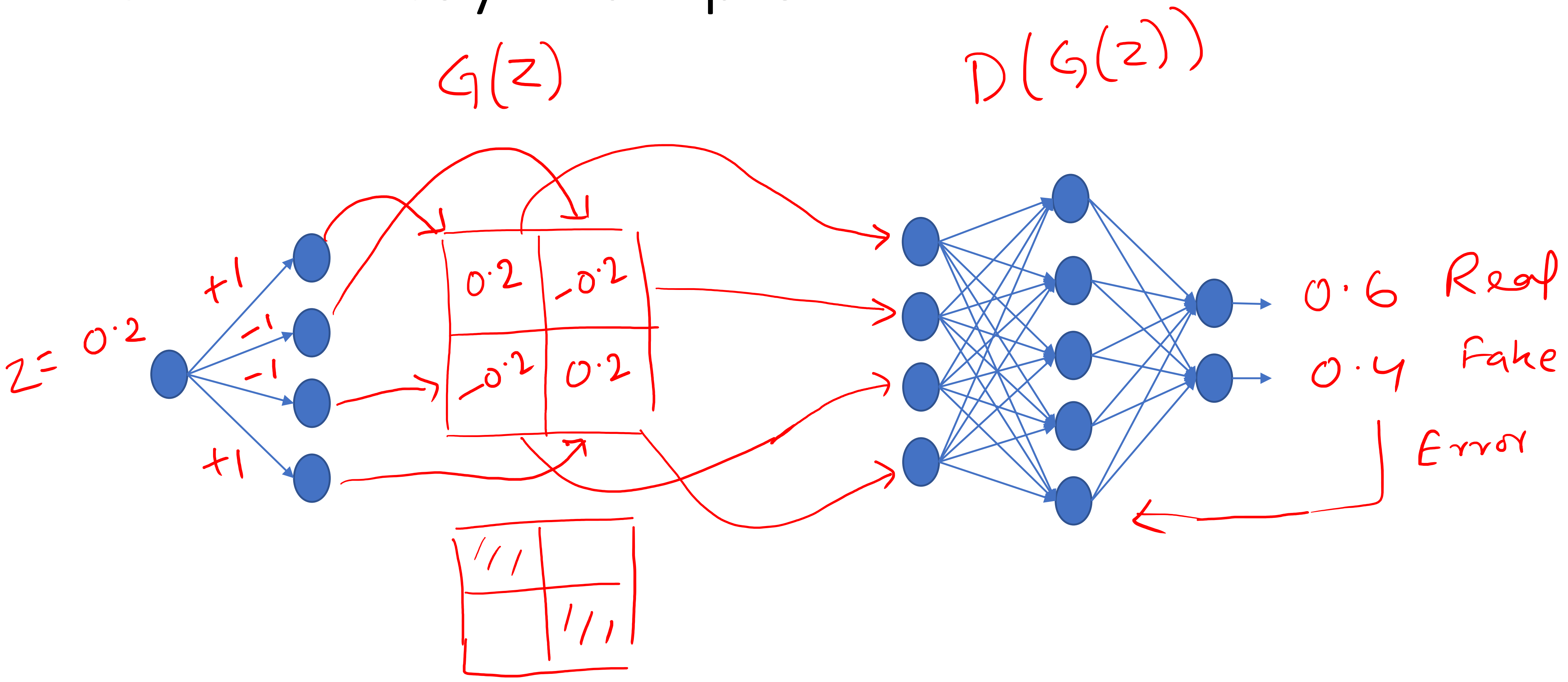
# GAN – A toy Example



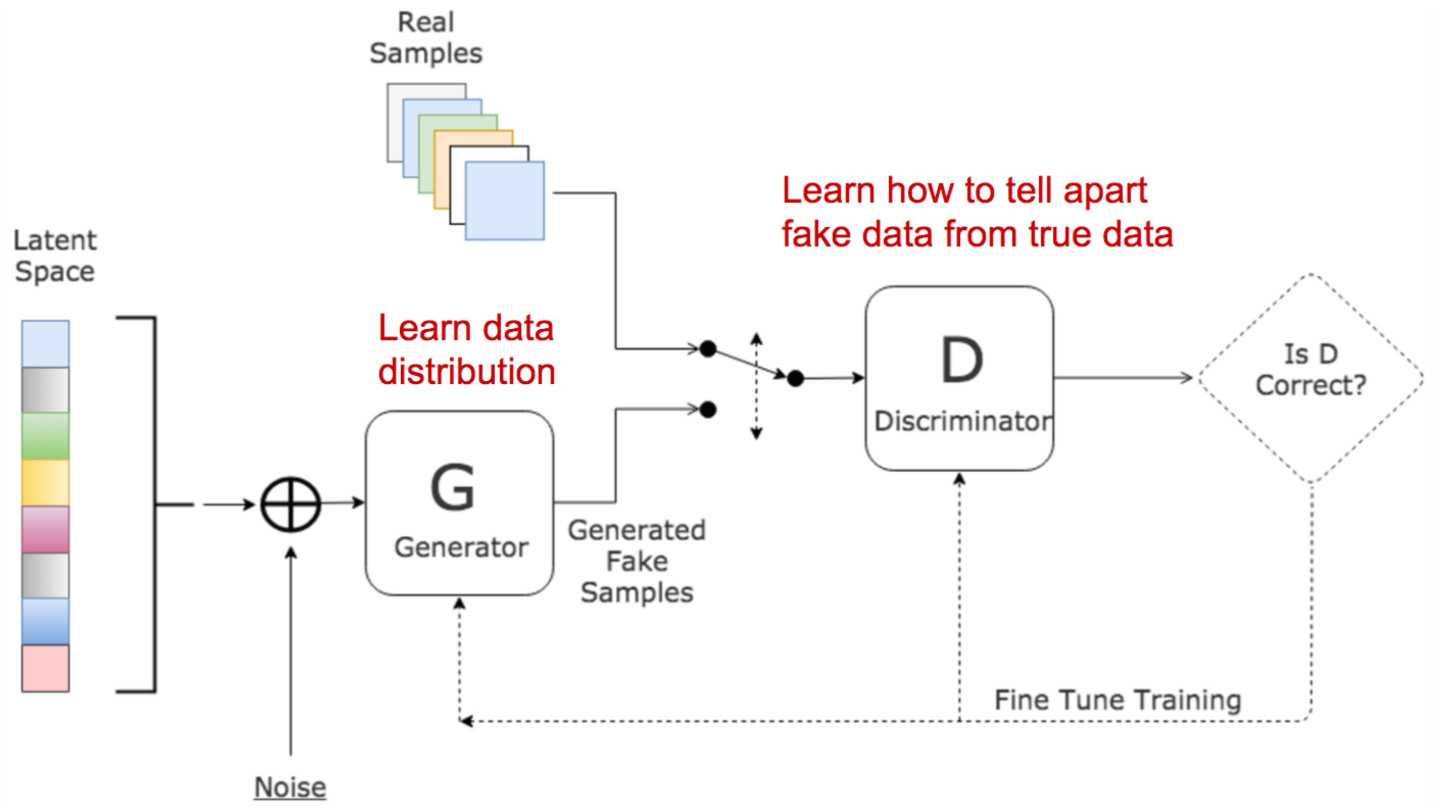
# GAN – A toy Example



# GAN – A toy Example



# GAN Framework




- Generator + Discriminator = **GAN**
- The latent vector belongs to some random distribution (Uniform/Gaussian)
- Both the generator and discriminator network parameters are updated during training

# GAN – Loss Function

- Discriminator's decision over real data should be accurate
  - **Maximize** $\mathbb{E}_{x\sim p_r(x)}[\log D(x)]$

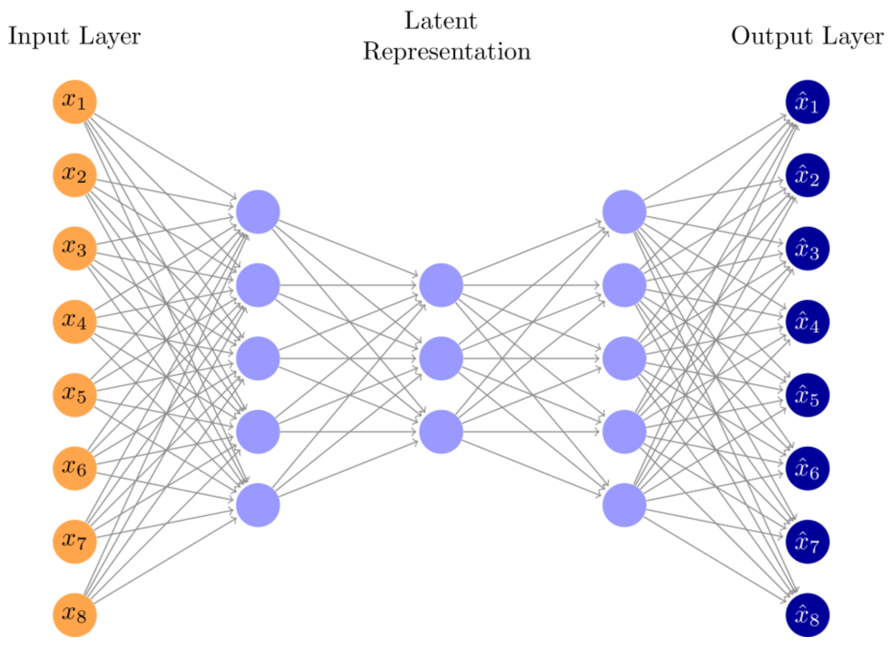
- Discriminator's decision over generated data should be considered fake
  - **Maximize** $\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$.

- Generator is trained to increase the chances of D producing a high probability for a fake sample

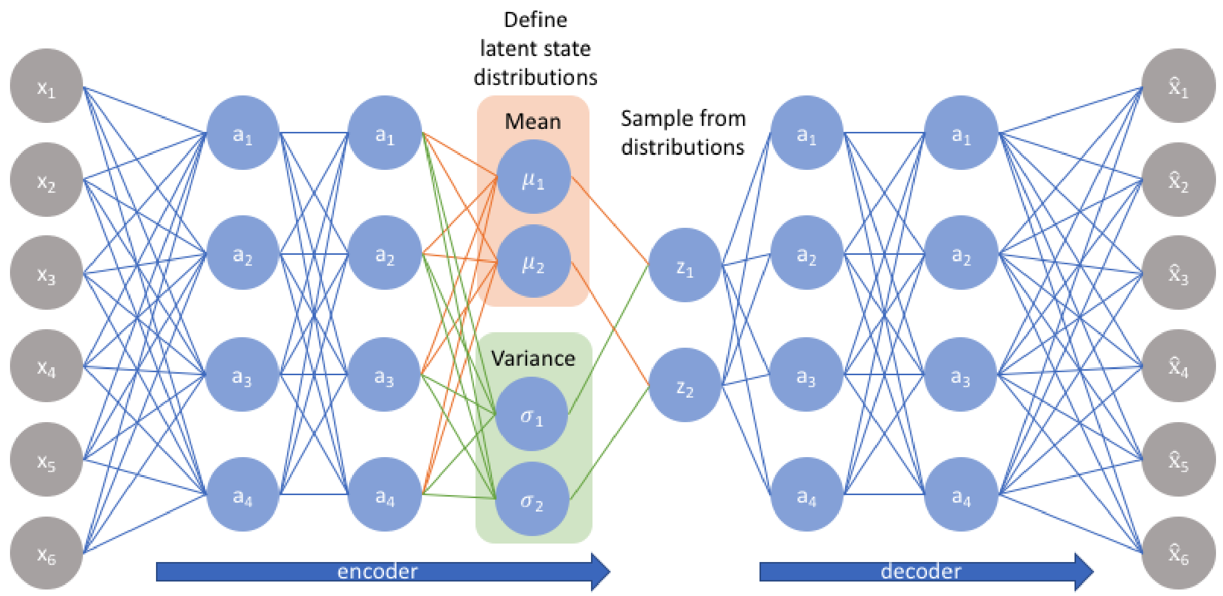
$$\min_G \max_D L(D, G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

# Variational AutoEncoder

# Variational AutoEncoder

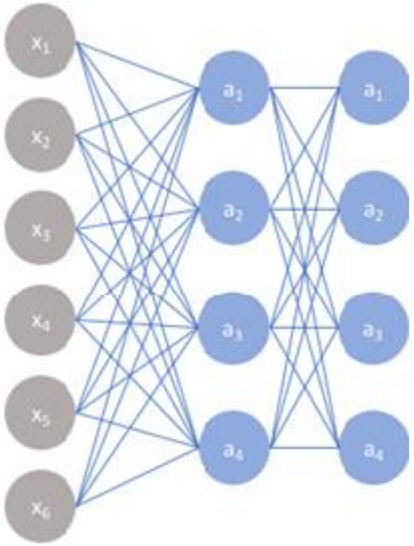



AutoEncoder



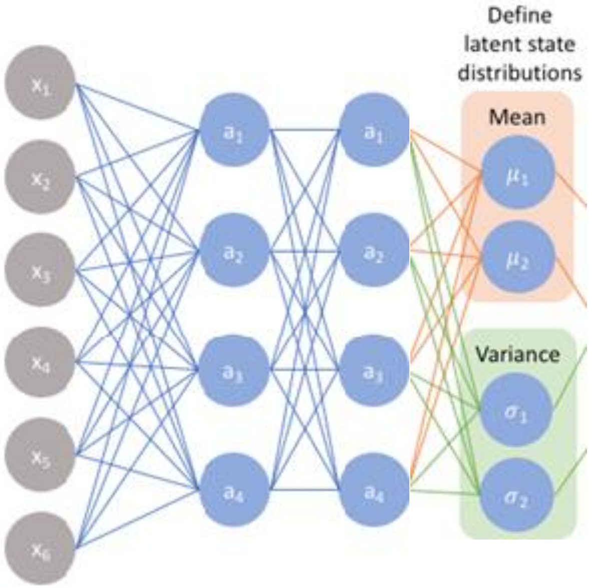

Variational AutoEncoder

# Variational AutoEncoder



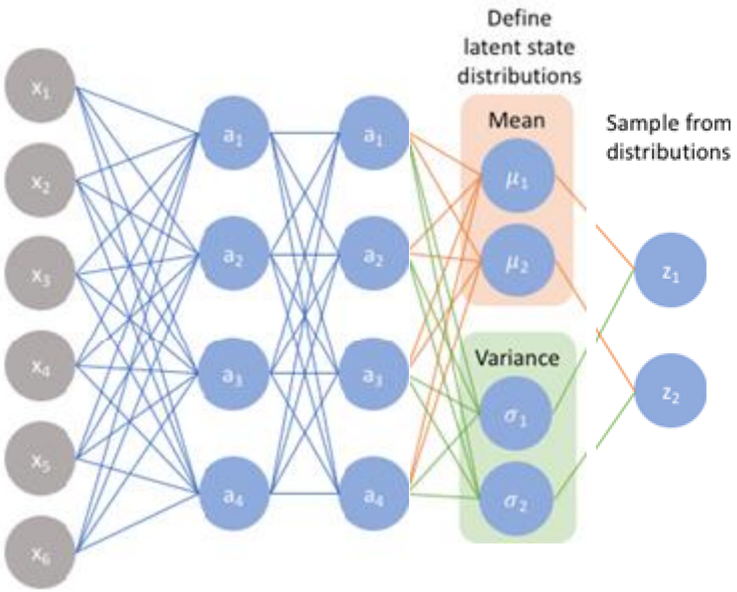

**x** **p** $P(z|x)$

# Variational AutoEncoder




**x** **p** $\mathrm{p}(z|x) \sim q_\theta(z|x), where\ \theta =< \mu, \sigma >$

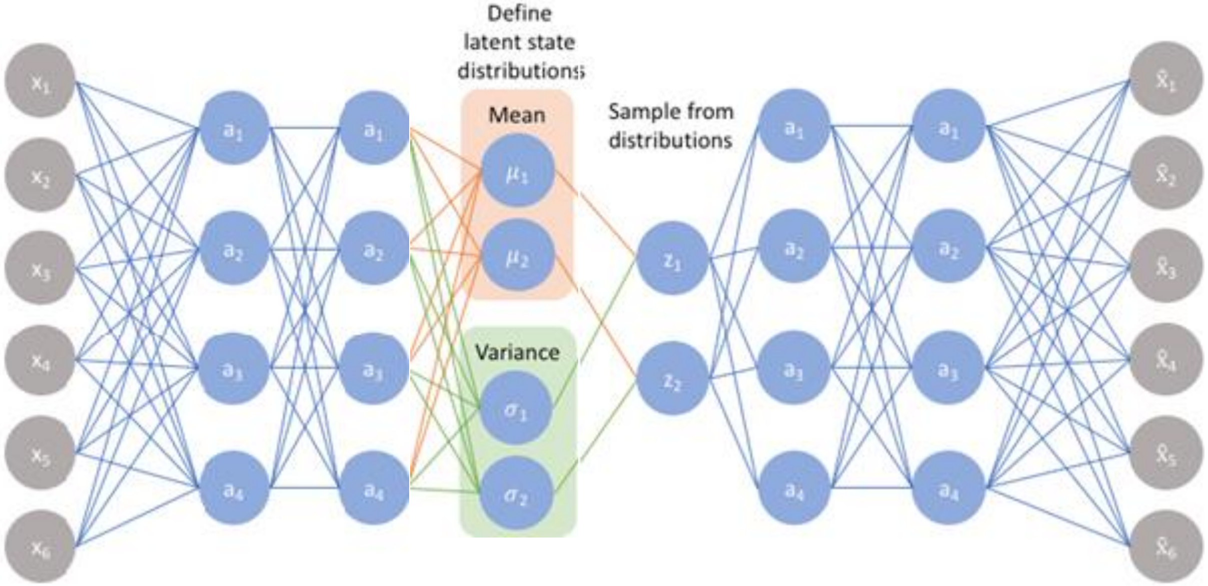
# Variational AutoEncoder



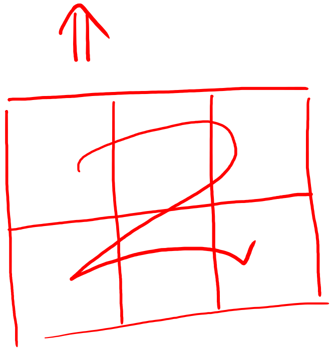

**x** **p** $\mathrm{p}(z|x) \sim q_\theta(z|x), where\ \theta =< \mu, \sigma >$

**Using $q_\theta(z|x)$, generate a random sample z'**

# Variational AutoEncoder



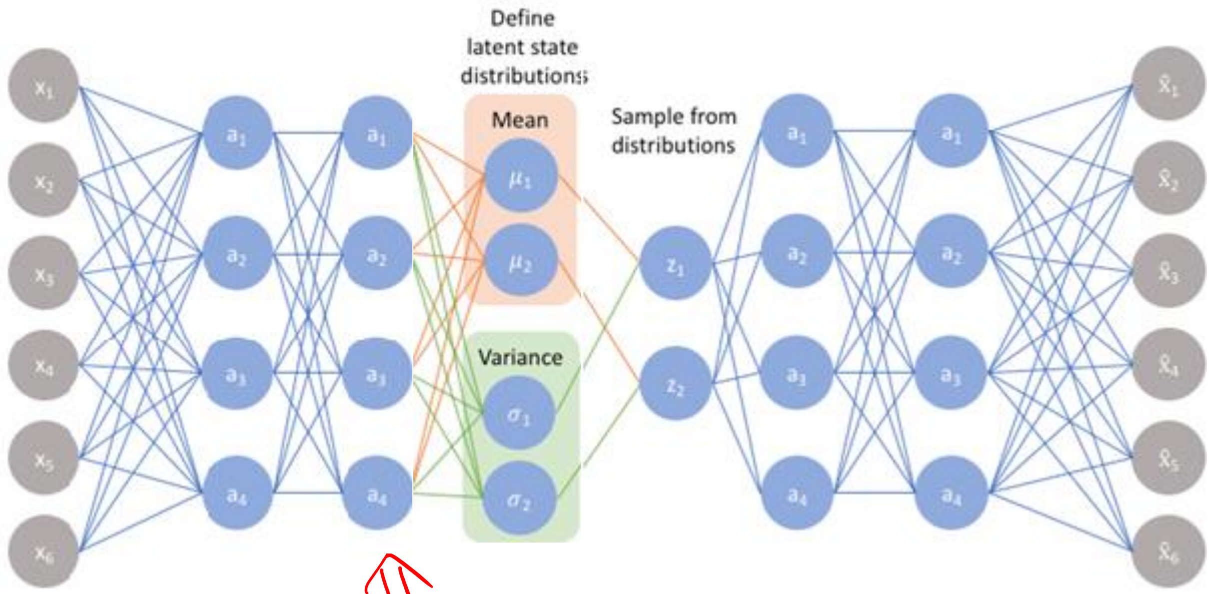

$p(x|z)$



$\sim$

| | | |
|---|---|---|
| .2 | .6 | .2 |
| 0.3 | 0.7 | .2 |

# Variational AutoEncoder



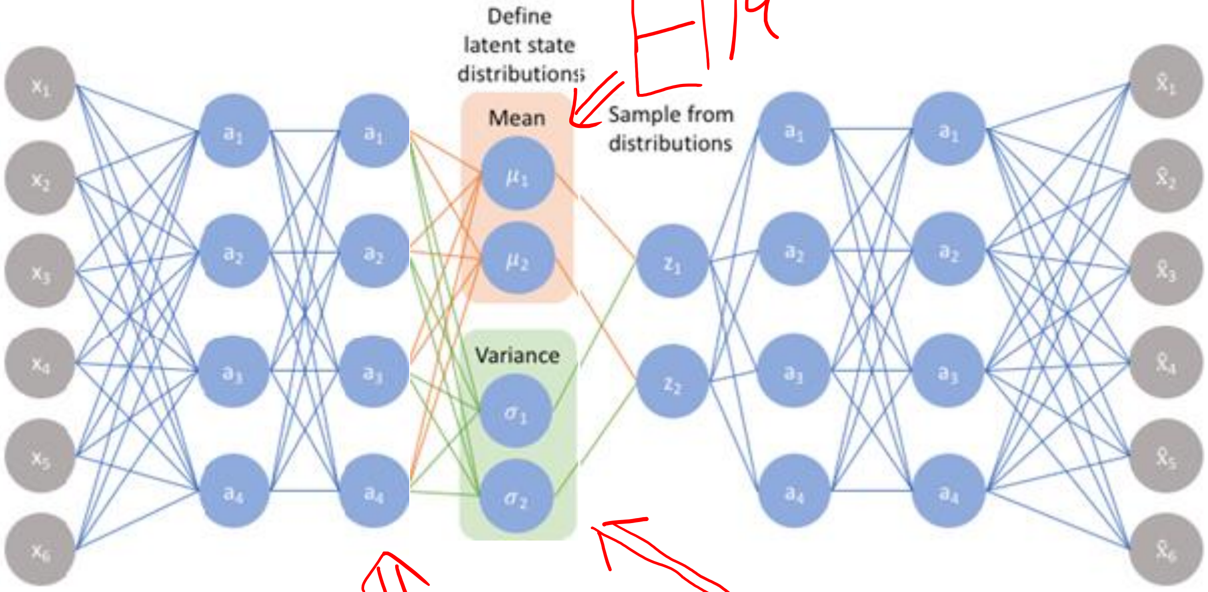

$p(x|z)$

| | | |
|---|---|---|
| •2 | •6 | •2 |
| 0.3 | 0.7 | •2 |

| | |
|---|---|
| 0.1 | 0.8 |
| 0.4 | 0.9 |

# Variational AutoEncoder

$\bar{\mu}$



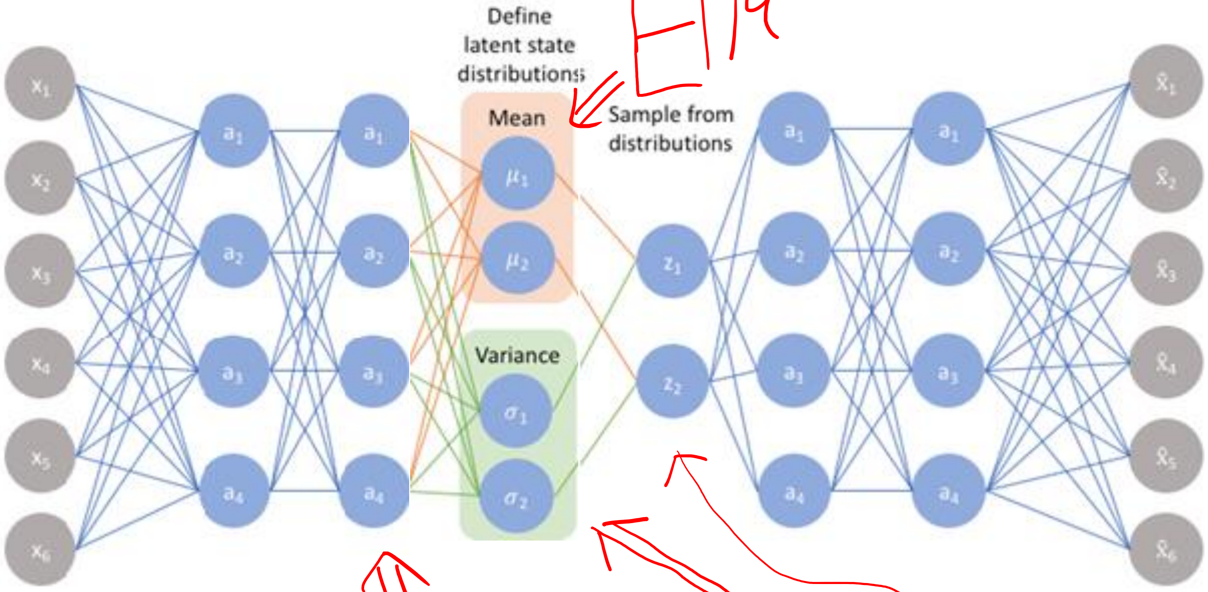

$p(x|z)$

| | | |
|---|---|---|
| .2 | .6 | .2 |
| 0.3 | 0.7 | .2 |

| | |
|---|---|
| 0.1 | 0.8 |
| 0.4 | 0.9 |

$\bar{\sigma}$

# Variational AutoEncoder




$\mu$

$p(x|z)$

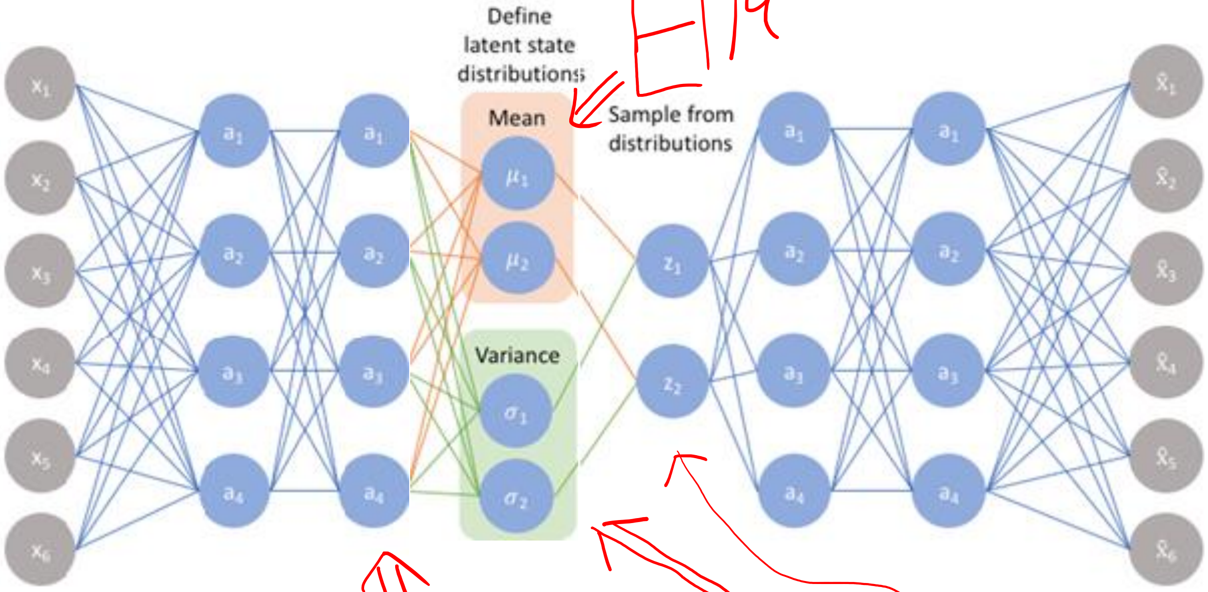
| | | |
|---|---|---|
| .2 | .6 | .2 |
| 0.3 | 0.7 | .2 |

| | |
|---|---|
| 0.1 | 0.8 |
| 0.4 | 0.9 |

$\sigma$

$z \sim q_\theta(z|x), \theta = \langle \mu, \sigma \rangle$

# Variational AutoEncoder




$\mu$

p$(x|z)$

| | | |
|---|---|---|
| 0.2 | 0.7 | 0.2 |
| 0.3 | 0.6 | 0.1 |

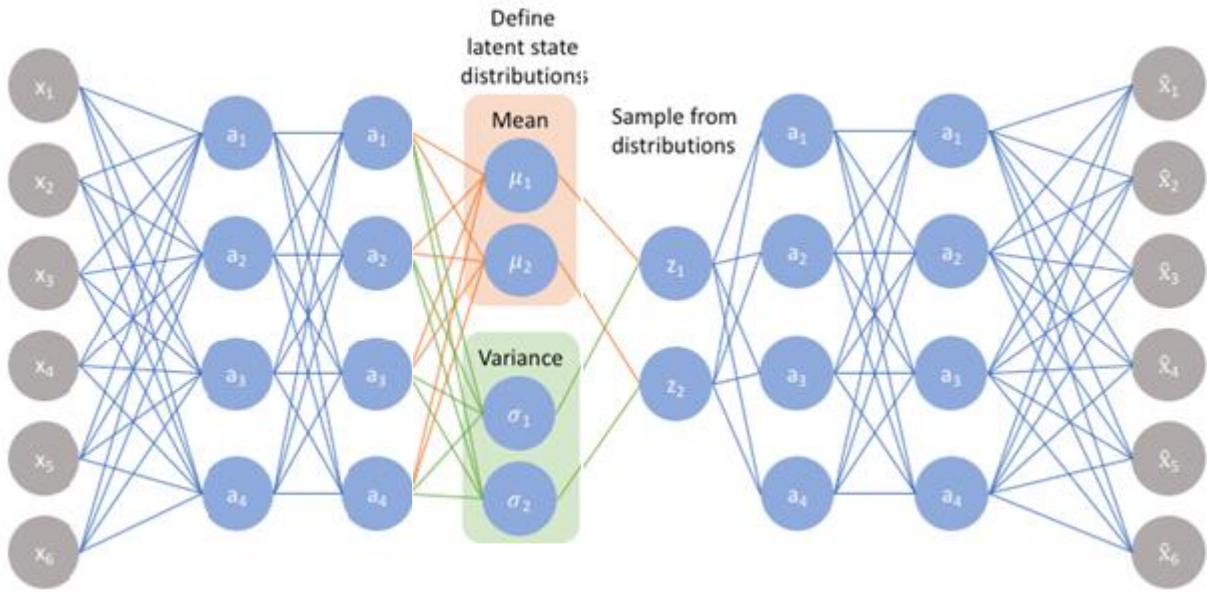
| | | |
|---|---|---|
| .2 | .6 | .2 |
| 0.3 | 0.7 | .2 |

| | |
|---|---|
| 0.1 | 0.8 |
| 0.4 | 0.9 |

$\sigma$

$z \sim q_\theta(z|x), \theta = \langle \mu, \sigma \rangle$

# Variational AutoEncoder – Loss Function



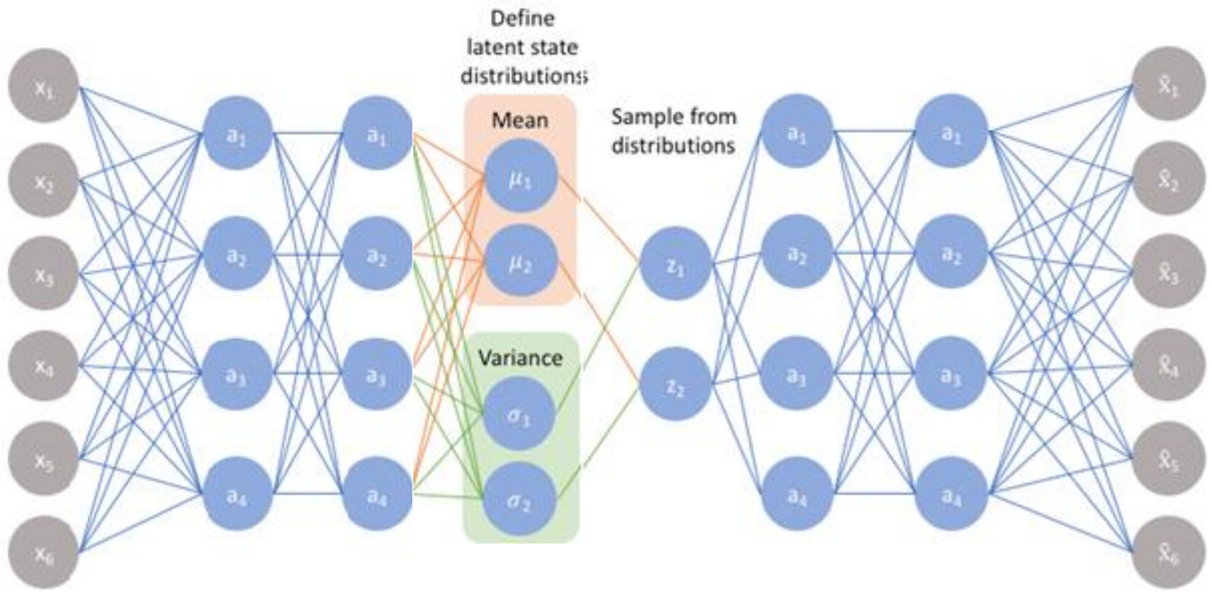

p$(x|z)$

$$E_{q(z|x)} \log p\left(x|z\right) - KL\left(q\left(z|x\right)||p\left(z\right)\right)$$

Reconstruction likelihood

Distance between learned distribution q and true prior distribution p.

# Variational AutoEncoder – Loss Function



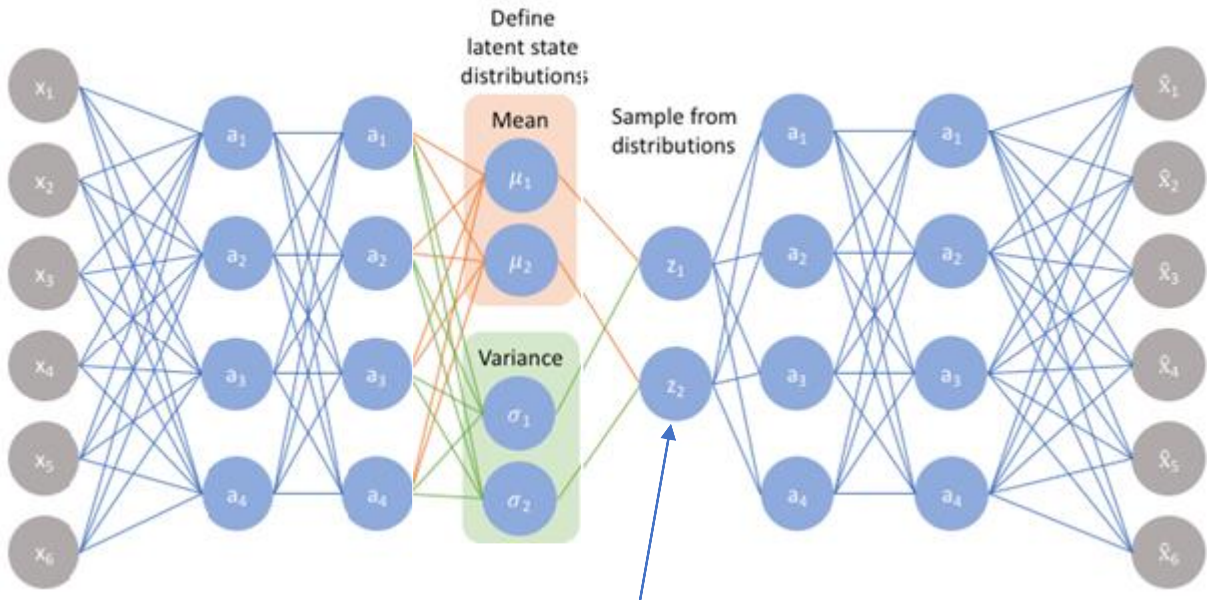

p$(x|z)$

$$E_{q(z|x)} \log p\left(x|z\right) - KL\left(q\left(z|x\right)||p\left(z\right)\right)$$

Reconstruction likelihood

Distance between learned distribution q and true prior distribution p.

# Variational AutoEncoder – Problem

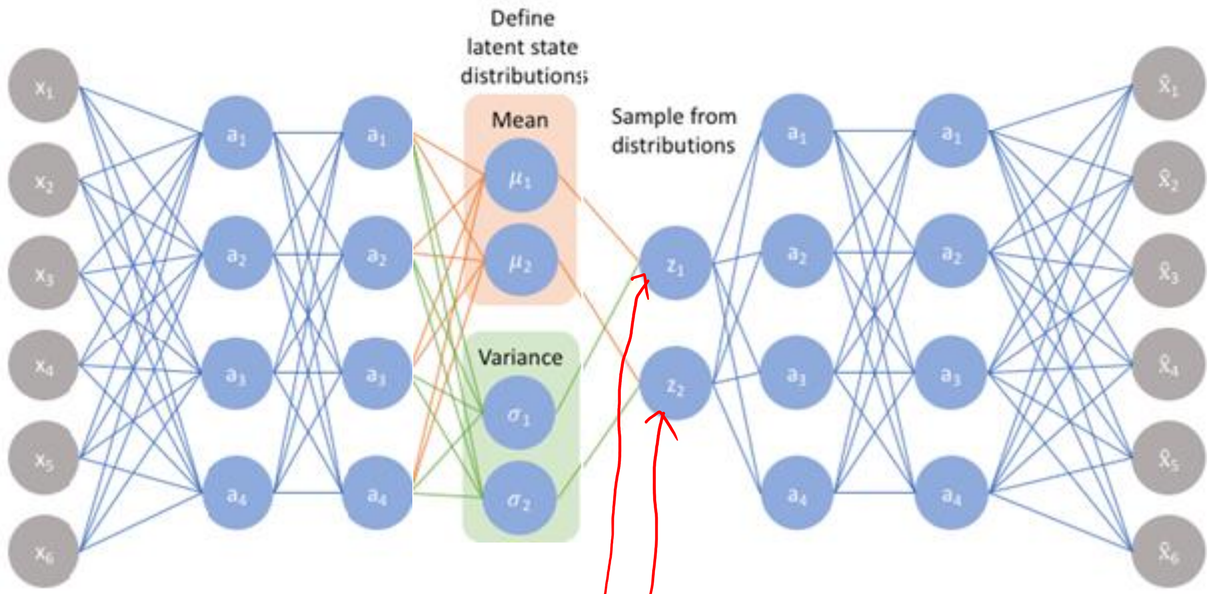



Z is randomly sample using $< \mu, \sigma >$. For a random node, Backpropagation can not flow through a random node

$p(x|z)$

$$E_{q(z|x)} \log p(x|z) - KL(q(z|x) || p(z))$$

# Variational AutoEncoder – Reparameterization



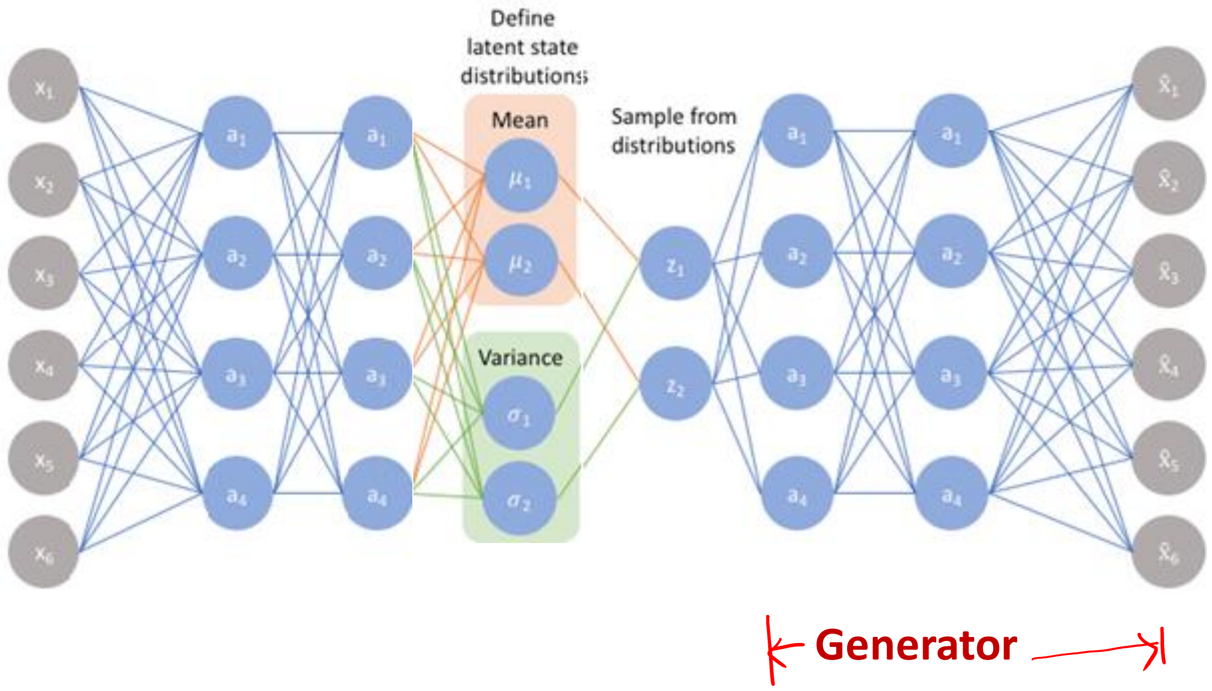

p$(x|z)$

$$E_{q(z|x)} \log p(x|z) - KL(q(z|x) || p(z))$$

Add Noise

$N(0,1) \Leftarrow$ Gaussian Noise

# Variational AutoEncoder

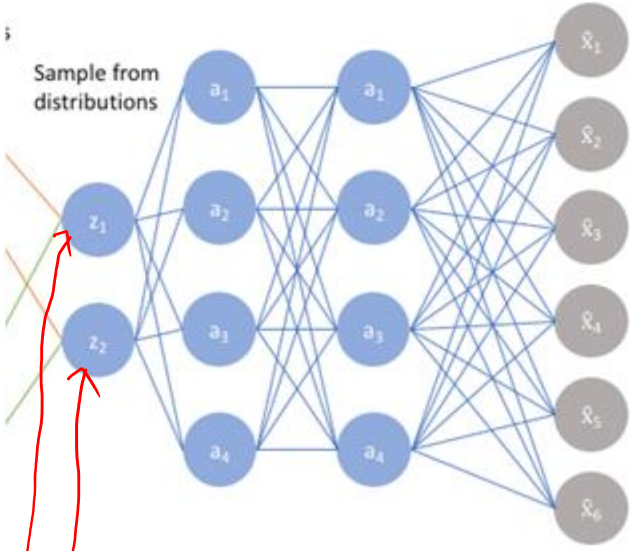



$p(x|z)$

# Variational AutoEncoder – Reparameterization

$\bar{\mu}$

$\bar{\sigma}$




$p(x|z)$

$$E_{q(z|x)} \log p\left(x|z\right) - KL\left(q\left(z|x\right)||p\left(z\right)\right)$$

Add Random Noise

$N(0,1) \Leftarrow$ Gaussian Noise