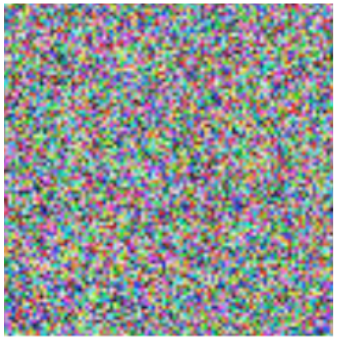
Directed Generative Nets 2

F. Noé[1]

Deep Learning Classes, FU Berlin 2018

Generative Neural Networks

Noise ~ N(0,1)



Generative Model

Generative Neural Networks

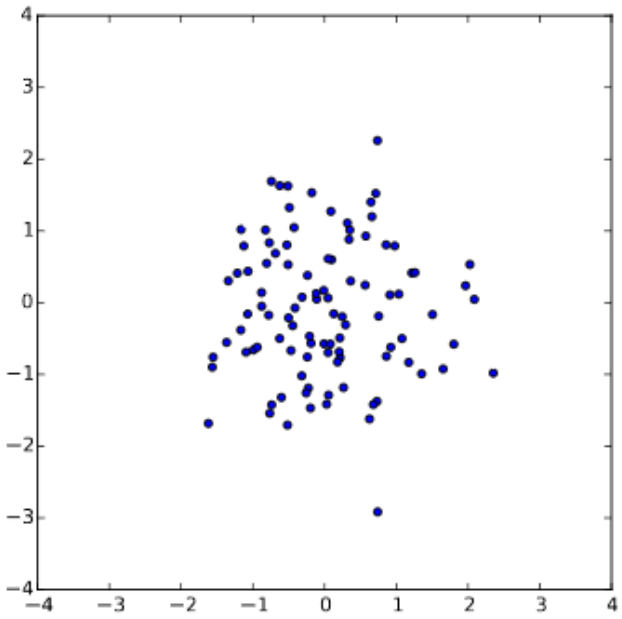
- **Idea**: Learn to sample intractable $p(\mathbf{x})$ by sampling tractable latent distribution

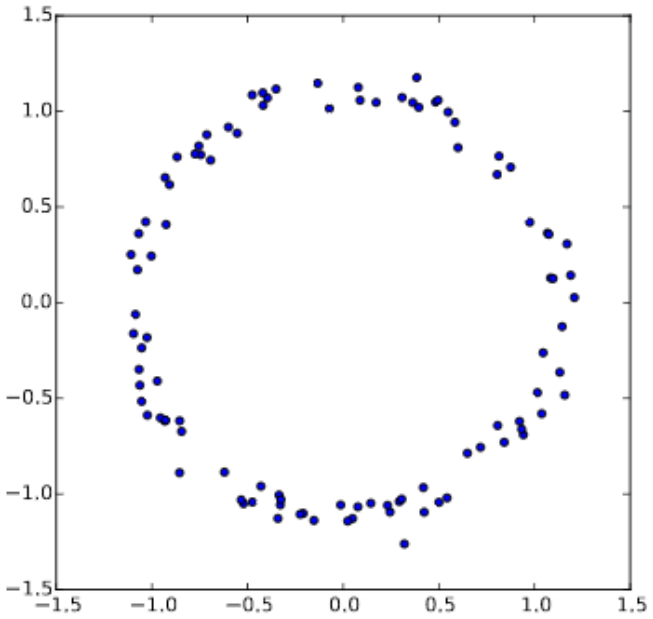
$$\mathbf{z} \sim p(\mathbf{z})$$

and perform a linear transformation to a desired distribution:

$$\mathbf{x} = G(\mathbf{z}, \theta) \sim p(\mathbf{x}).$$

- **Example**:
  - **Left**: Samples from normal distribution, $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
  - **Right**: Samples mapped through $G(\mathbf{z}) = \frac{\mathbf{z}}{10} + \frac{\mathbf{z}}{\|\mathbf{z}\|}$ to form a ring.





Generative Neural Networks

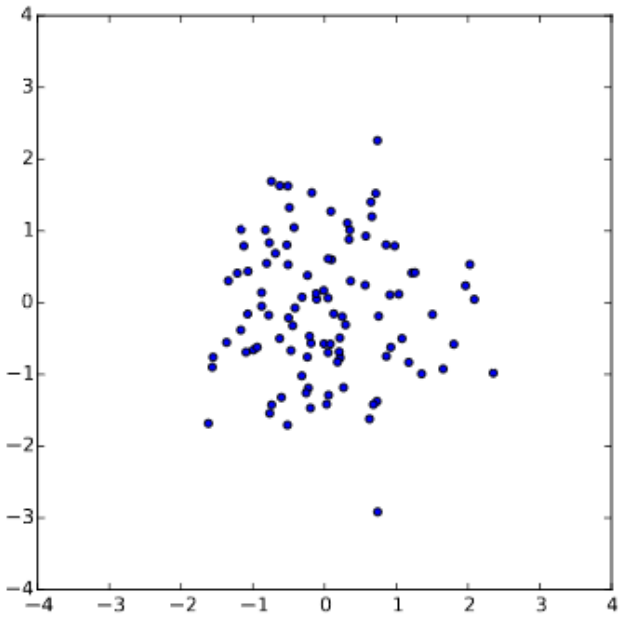
- **Idea**: Learn to sample intractable $p(\mathbf{x})$ by sampling tractable latent distribution

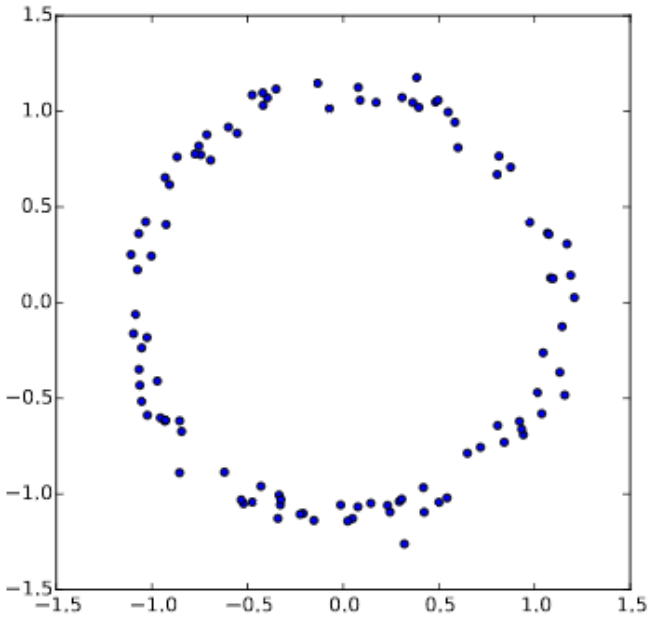
$$\mathbf{z} \sim p(\mathbf{z})$$

and perform a linear transformation to a desired distribution:

$$\mathbf{x} = G(\mathbf{z}, \theta) \sim p(\mathbf{x}).$$

- **Example**:
  - **Left**: Samples from normal distribution, $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
  - **Right**: Samples mapped through $G(\mathbf{z}) = \frac{\mathbf{z}}{10} + \frac{\mathbf{z}}{\|\mathbf{z}\|}$ to form a ring.





Differentiable Generator Nets

- **Idea**: Learn to sample intractable $p(\mathbf{x})$ by sampling tractable latent distribution

$$\mathbf{z} \sim p(\mathbf{z})$$

and perform a linear transformation to a desired distribution:

$$\mathbf{x} = G(\mathbf{z}, \theta) \sim p(\mathbf{x}).$$

- **Complex Distribution**:
  - G feedforward neural network
  - *train* parameters θ to sample from correct distribution.
- Well-known neural network architectures:
  - **Variational Autoencoders** (inference net + generator net)
  - **Generative Adversarial Networks** (generator network + discriminator network)

Differentiable Generator Nets

- **Idea**: Learn to sample intractable $p(\mathbf{x})$ by sampling tractable latent distribution

$$\mathbf{z} \sim p(\mathbf{z})$$

and perform a linear transformation to a desired distribution:

$$\mathbf{x} = G(\mathbf{z}, \theta) \sim p(\mathbf{x}).$$

- **Complex Distribution**:
  - G feedforward neural network
  - *train* parameters θ to sample from correct distribution.
- Well-known neural network architectures:
  - **Variational Autoencoders** (inference net + generator net)
  - **Generative Adversarial Networks** (generator network + discriminator network)

Differentiable Generator Nets

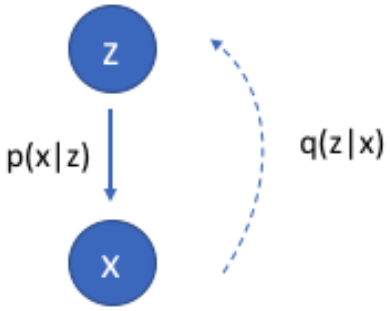
- **Idea**: Learn to sample intractable $p(\mathbf{x})$ by sampling tractable latent distribution

$$\mathbf{z} \sim p(\mathbf{z})$$

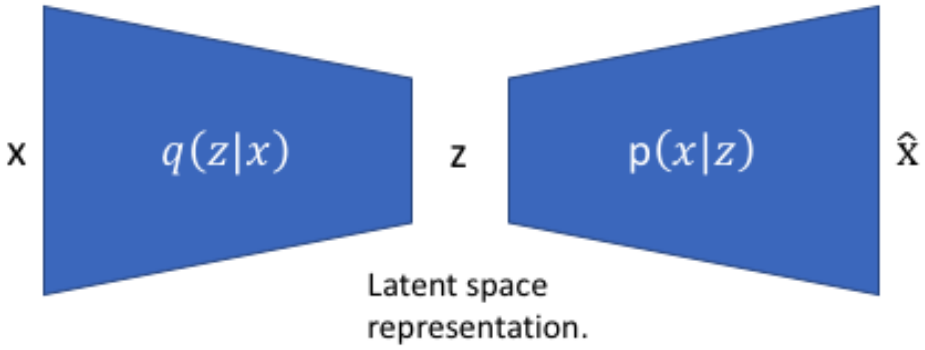
and perform a linear transformation to a desired distribution:

$$\mathbf{x} = G(\mathbf{z}, \theta) \sim p(\mathbf{x}).$$

- **Complex Distribution**:
  - G feedforward neural network
  - *train* parameters θ to sample from correct distribution.
- Well-known neural network architectures:
  - **Variational Autoencoders** (inference net + generator net)
  - **Generative Adversarial Networks** (generator network + discriminator network)

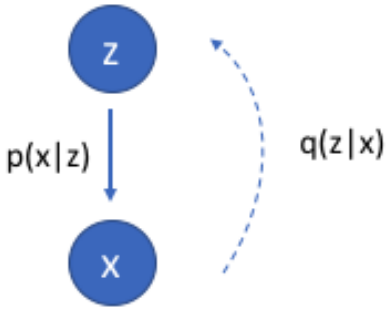


We'd like to use our observations to understand the hidden variable.

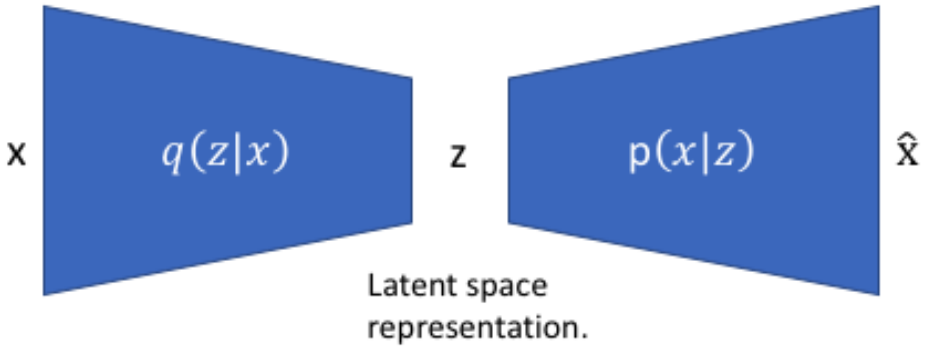


$$\min \left\{ -\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x} \mid \mathbf{z}) + D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})) \right\}$$

- **Encoder** $q(\mathbf{z} \mid \mathbf{x})$ (inference network, recognition model):
  - Maps to latent space
  - Models approximate posterior distribution q.
  - $\mathscr{D}_{\mathrm{KL}}\left[q(\mathbf{z} \mid \mathbf{x}) \parallel p_{\mathrm{model}}(\mathbf{z})\right]$ tries to make $q(\mathbf{z} \mid \mathbf{x})$ and $p_{\mathrm{model}}(\mathbf{z})$ similar.
- **Decoder** $p(\mathbf{x} \mid \mathbf{z})$.
  - Decodes $\mathbf{z} \rightarrow \hat{\mathbf{x}}$ with the aim to reconstruct the input $\mathbf{x}$.
  - $\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|x)} \log p_{\mathrm{model}}(\mathbf{x} \mid \mathbf{z})$ reconstruction log-likelihood



We'd like to use our observations to understand the hidden variable.

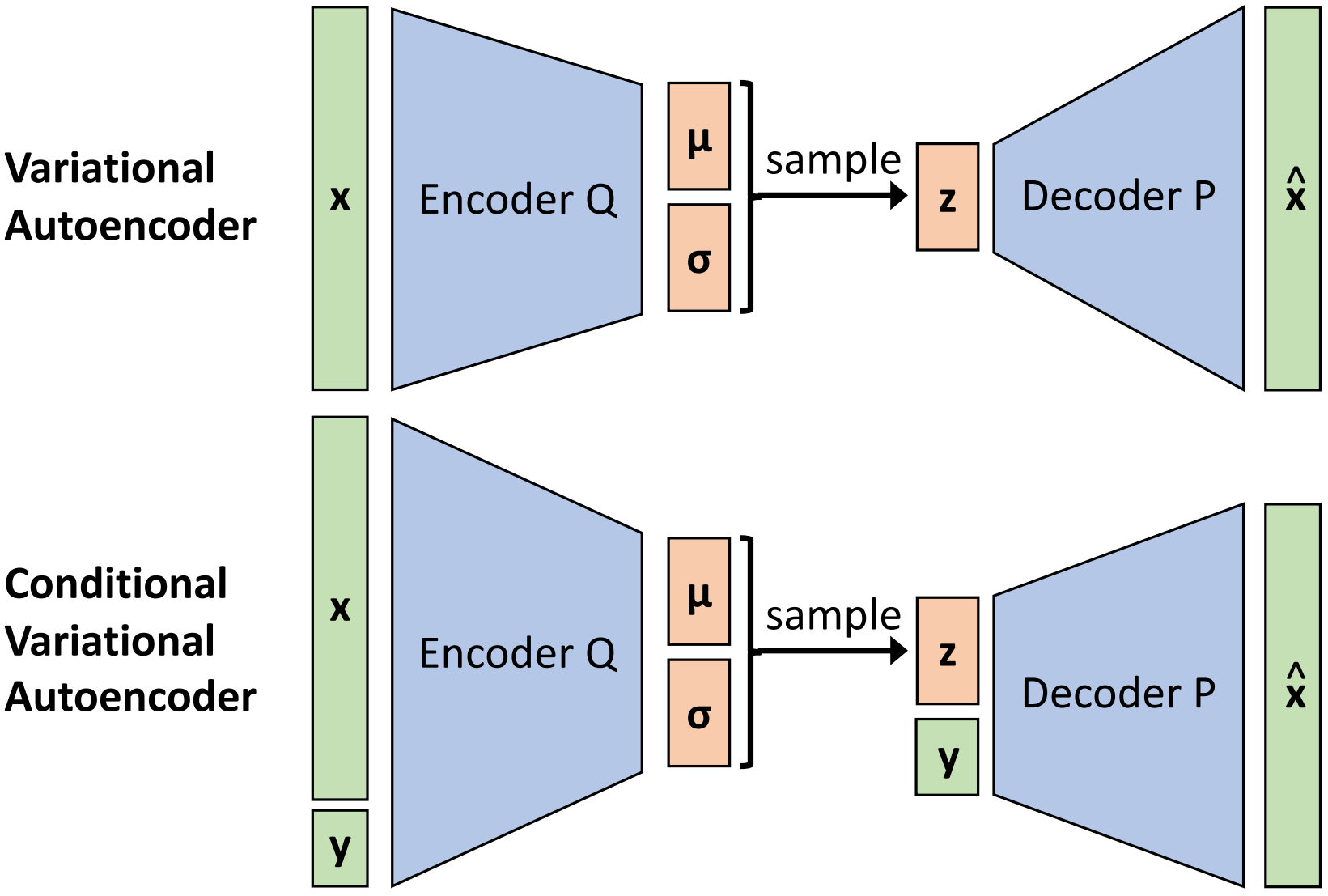


$$\min \left\{ -\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x} \mid \mathbf{z}) + D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})) \right\}$$

- **Encoder** $q(\mathbf{z} \mid \mathbf{x})$ (inference network, recognition model):
  - Maps to latent space
  - Models approximate posterior distribution q.
  - $\mathscr{D}_{\mathrm{KL}}\left[q(\mathbf{z} \mid \mathbf{x}) \parallel p_{\mathrm{model}}(\mathbf{z})\right]$ tries to make $q(\mathbf{z} \mid \mathbf{x})$ and $p_{\mathrm{model}}(\mathbf{z})$ similar.
- **Decoder** $p(\mathbf{x} \mid \mathbf{z})$.
  - Decodes $\mathbf{z} \rightarrow \hat{\mathbf{x}}$ with the aim to reconstruct the input $\mathbf{x}$.
  - $\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|x)} \log p_{\mathrm{model}}(\mathbf{x} \mid \mathbf{z})$ reconstruction log-likelihood

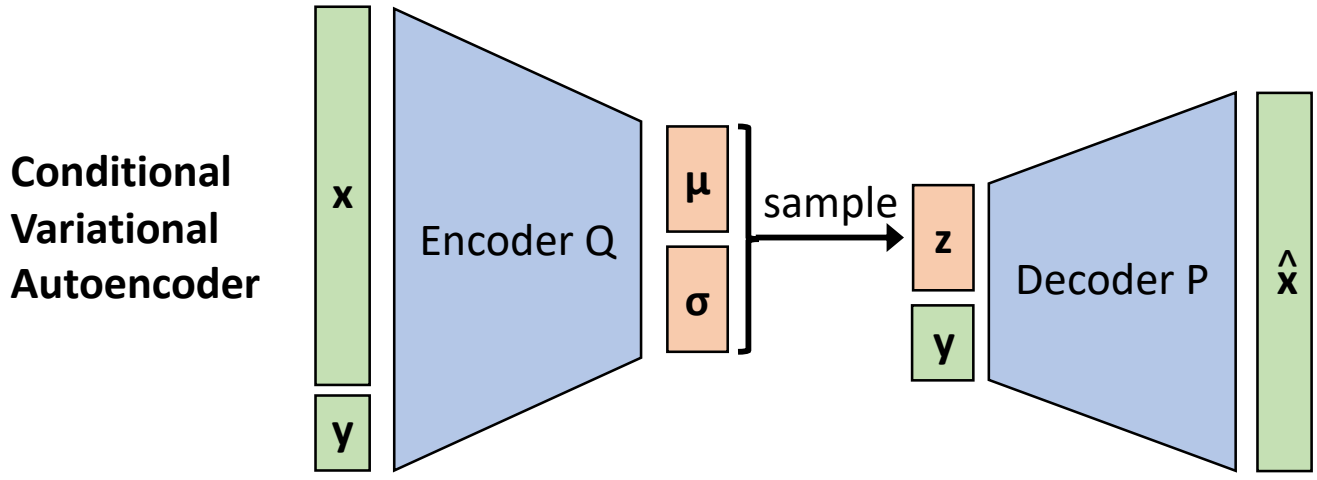
Conditional Variational Autoencoders

Structure



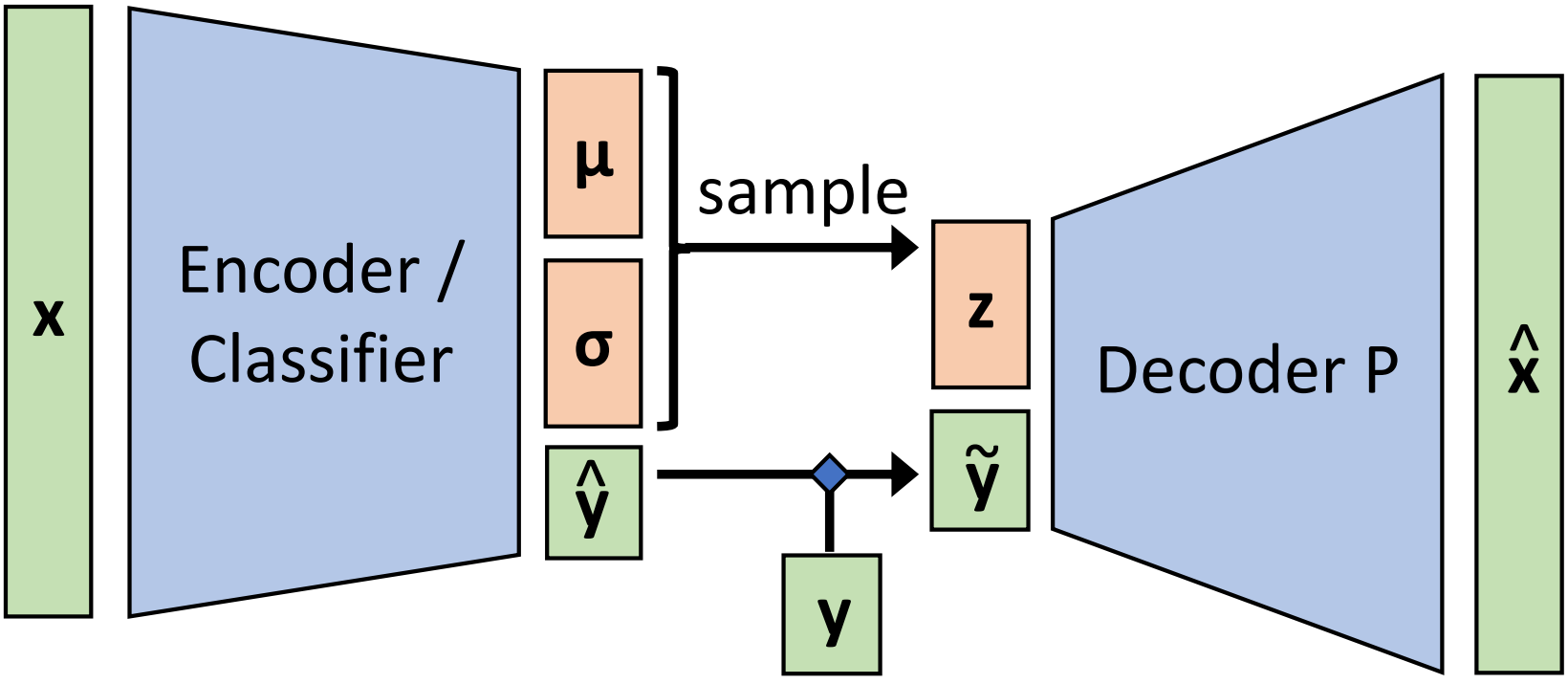
Conditional Variational Autoencoders

Structure



Classification Variational Autoencoder

Structure



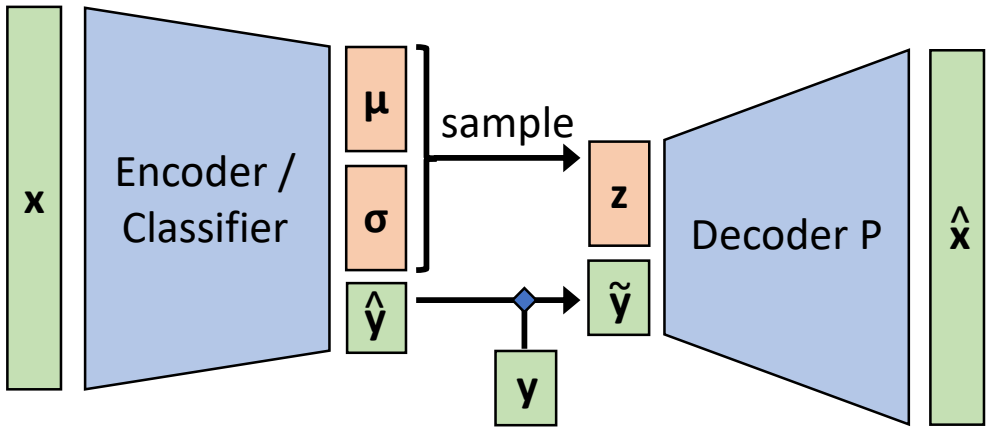
- **Supervised training**:

$$\min \left\{ \underbrace{-\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x} \mid \mathbf{z})}_{\text{Reconstruction loss}} + \underbrace{D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))}_{\text{Regularization loss}} + \underbrace{\|\mathbf{y} - \hat{\mathbf{y}}\|^2}_{\text{Classification loss}} \right\}$$

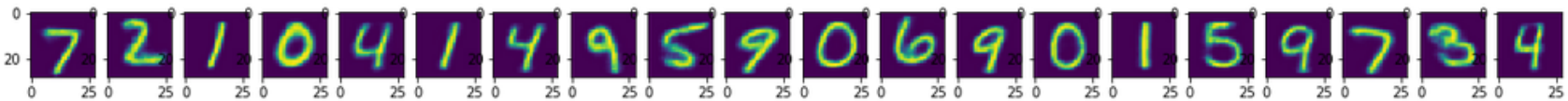
- **Unsupervised training** without classification loss.

Classification Variational Autoencoder

Input-output encoding (not optimized...)

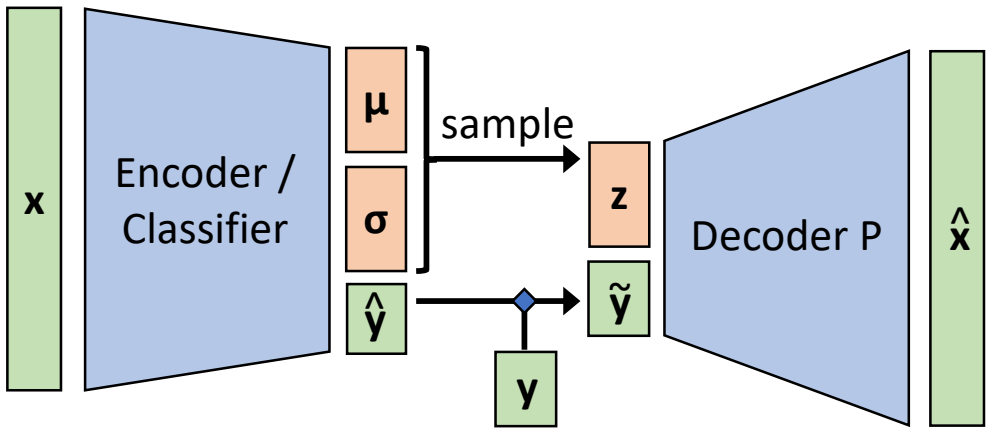


Reconstruction:

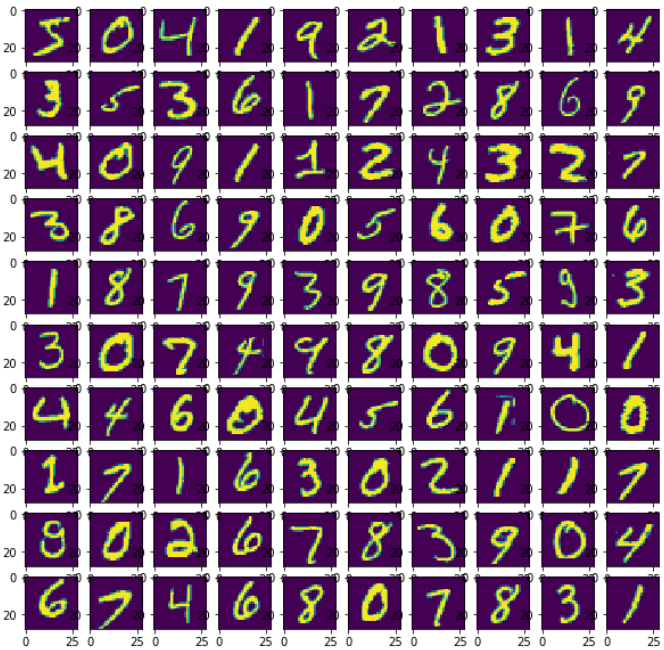


Classification Variational Autoencoder

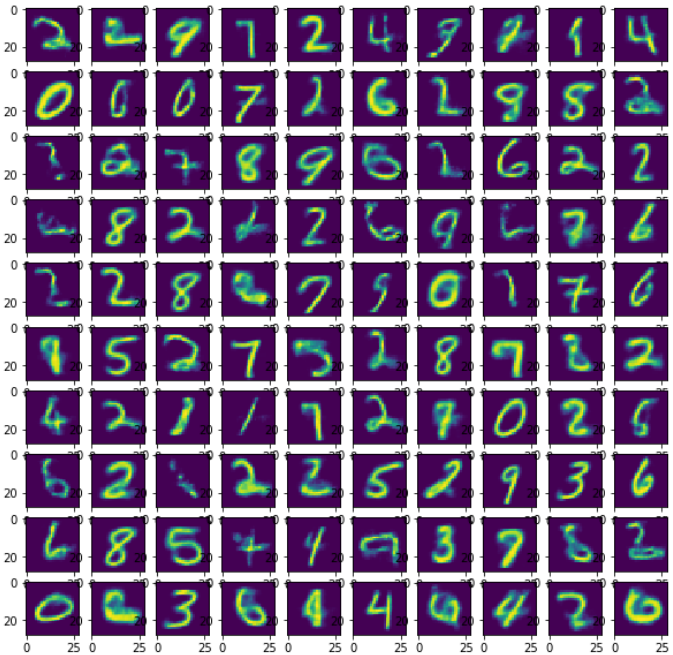
Sampling (not optimized...)



Input Data

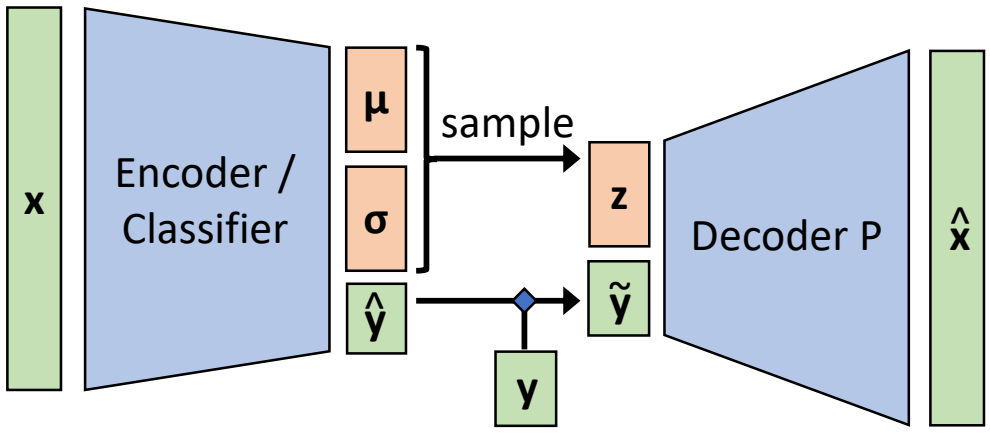


Generated Data

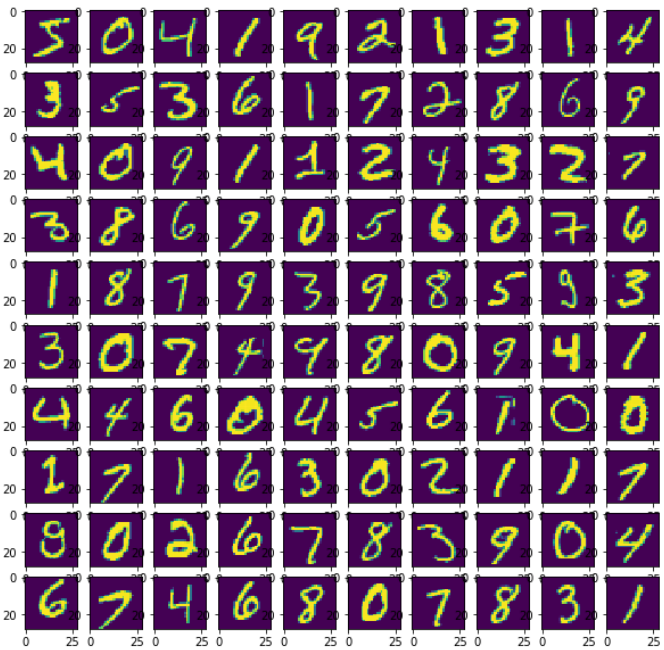


Classification Variational Autoencoder

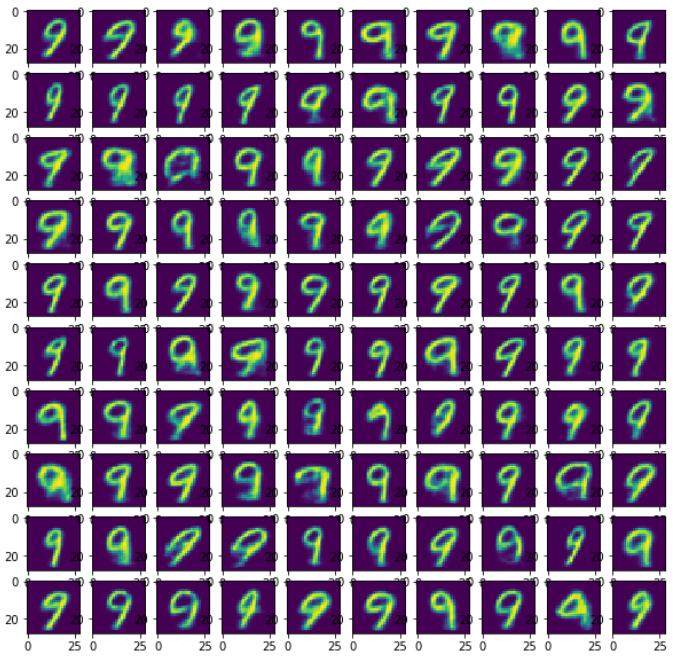
Conditional Sampling (not optimized...)



Input Data

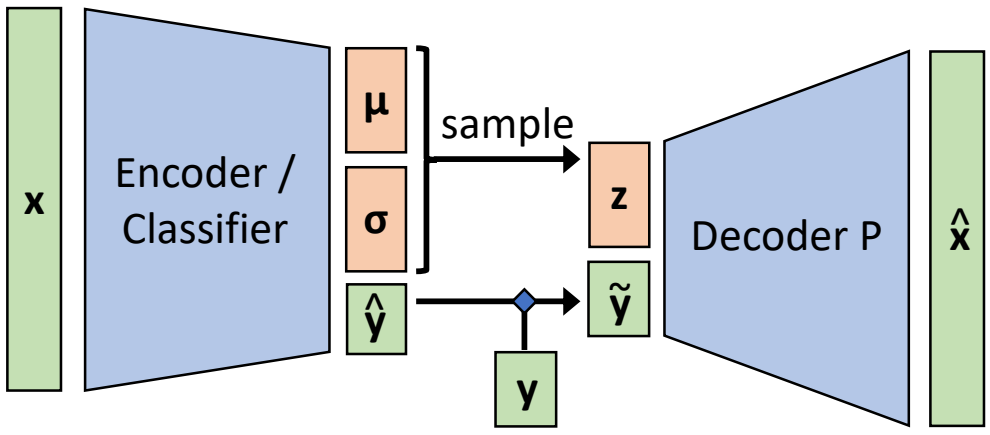


Generated Data



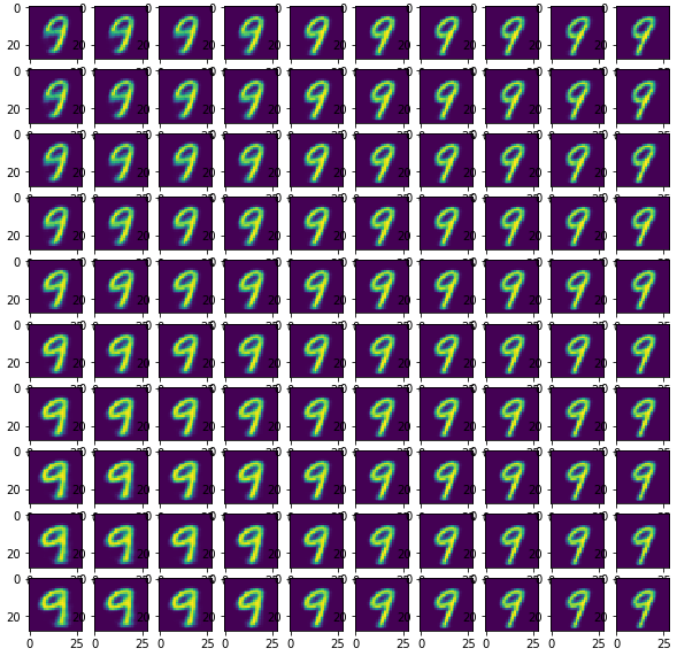
Classification Variational Autoencoder

Interpolation (not optimized...)



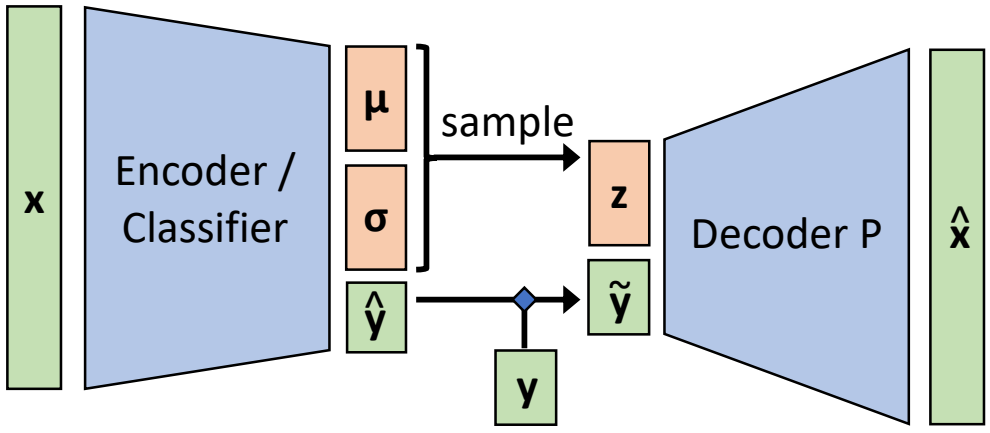
Input Data

Generated Data



Classification Variational Autoencoder

Semi-supervised learning (not optimized...)



Semi-supervised learning ($N_{\text{train}}^{\text{supervised}} + N_{\text{train}}^{\text{unsupervised}} = 60,000$)

$N_{\text{train}}^{\text{supervised}}$	100	400	1000	4000	10000	40000	60000
Test error	0.2569	0.3577	0.7047	0.9291	0.9697	0.9878	0.9920

- **Idea**: Game in which the Generator network competes against a Discriminator network, i.e. these two networks are adversaries.
- **Generator network** G: directly produces samples

$$\mathbf{x} = G(\mathbf{z}; \theta_G)$$

- **Discriminator network** D:
  - Attempts to distinguish between samples drawn from the training data and samples drawn from the generator.
  - Emits a probability value that $\mathbf{x}$ is a true sample and not a fake:

$$p_{\text{true}}(\mathbf{x}) = D(\mathbf{x}; \theta_D)$$

- Simplest formulation: **Zero Sum Game**
  - Discriminator receives payoff $v(\theta_G, \theta_D)$
  - Generator receives payoff $-v(\theta_G, \theta_D)$

Generative Adversarial Network

GAN (Goodfellow et al., 2014)

- **Idea**: Game in which the Generator network competes against a Discriminator network, i.e. these two networks are adversaries.
- **Generator network** G: directly produces samples

$$\mathbf{x} = G(\mathbf{z}; \theta_G)$$

- **Discriminator network** D:
  - Attempts to distinguish between samples drawn from the training data and samples drawn from the generator.
  - Emits a probability value that $\mathbf{x}$ is a true sample and not a fake:

$$p_{\text{true}}(\mathbf{x}) = D(\mathbf{x}; \theta_D)$$

- Simplest formulation: **Zero Sum Game**
  - Discriminator receives payoff $v(\theta_G, \theta_D)$
  - Generator receives payoff $-v(\theta_G, \theta_D)$

- **Idea**: Game in which the Generator network competes against a Discriminator network, i.e. these two networks are adversaries.
- **Generator network** G: directly produces samples

$$\mathbf{x} = G(\mathbf{z}; \theta_G)$$

- **Discriminator network** D:
  - Attempts to distinguish between samples drawn from the training data and samples drawn from the generator.
  - Emits a probability value that $\mathbf{x}$ is a true sample and not a fake:

$$p_{\text{true}}(\mathbf{x}) = D(\mathbf{x}; \theta_D)$$

- Simplest formulation: **Zero Sum Game**
  - Discriminator receives payoff $v(\theta_G, \theta_D)$
  - Generator receives payoff $-v(\theta_G, \theta_D)$

- **Idea**: Game in which the Generator network competes against a Discriminator network, i.e. these two networks are adversaries.
- **Generator network** G: directly produces samples

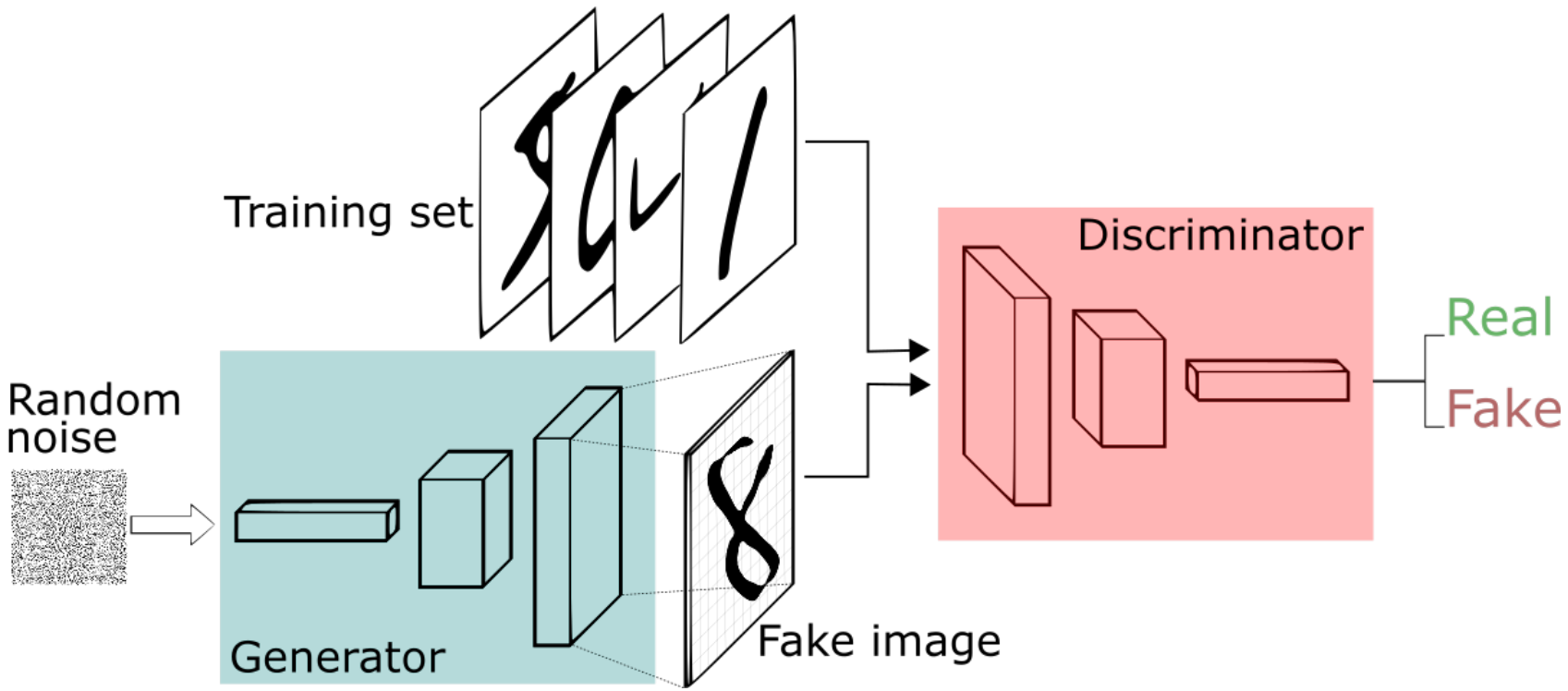
$$\mathbf{x} = G(\mathbf{z}; \theta_G)$$

- **Discriminator network** D:
  - Attempts to distinguish between samples drawn from the training data and samples drawn from the generator.
  - Emits a probability value that $\mathbf{x}$ is a true sample and not a fake:

$$p_{\text{true}}(\mathbf{x}) = D(\mathbf{x}; \theta_D)$$

- Simplest formulation: **Zero Sum Game**
  - Discriminator receives payoff $v(\theta_G, \theta_D)$
  - Generator receives payoff $-v(\theta_G, \theta_D)$

- Discrimiator randomly receives either generated (fake) or training (real) sample as input.
- Generator tries to fake a sample and trick Discriminator into believing it, Discriminator tries to reveal the truth.



https://deeplearning4j.org/generative-adversarial-network

- During learning, **each player attempts to maximize its own payoff**, so that at convergence:

$$\hat{\theta}_G = \arg\min_{\theta_G} \max_{\theta_D} v(\theta_G, \theta_D).$$

- **Default choice**

$$v(\theta_G, \theta_D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}; \theta_D) + \mathbb{E}_{\mathbf{x} \sim p_{\text{model}}} \log (1 - D(\mathbf{x}; \theta_D))$$

  - Discriminator gets reward for correctly classifying samples as real or fake.
  - Generator gets reward when fooling the classifier into believing its samples are real.

- **At convergence**:
  - The Generator's samples are indistinguishable from real data
  - Discriminator outputs 0.5 everywhere. The discriminator may then be discarded.

- During learning, **each player attempts to maximize its own payoff**, so that at convergence:

$$\hat{\theta}_G = \arg\min_{\theta_G} \max_{\theta_D} v(\theta_G, \theta_D).$$

- **Default choice**

$$v(\theta_G, \theta_D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}; \theta_D) + \mathbb{E}_{\mathbf{x} \sim p_{\text{model}}} \log (1 - D(\mathbf{x}; \theta_D))$$

  - Discriminator gets reward for correctly classifying samples as real or fake.
  - Generator gets reward when fooling the classifier into believing its samples are real.

- **At convergence**:
  - The Generator's samples are indistinguishable from real data
  - Discriminator outputs 0.5 everywhere. The discriminator may then be discarded.

- During learning, **each player attempts to maximize its own payoff**, so that at convergence:

$$\hat{\theta}_G = \arg\min_{\theta_G} \max_{\theta_D} v(\theta_G, \theta_D).$$

- **Default choice**

$$v(\theta_G, \theta_D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}; \theta_D) + \mathbb{E}_{\mathbf{x} \sim p_{\text{model}}} \log (1 - D(\mathbf{x}; \theta_D))$$

  - Discriminator gets reward for correctly classifying samples as real or fake.
  - Generator gets reward when fooling the classifier into believing its samples are real.
- **At convergence**:
  - The Generator's samples are indistinguishable from real data
  - Discriminator outputs 0.5 everywhere. The discriminator may then be discarded.

- **Pros**:
  - Learning process does not require approximations such as variational inference.
  - When $\max_{\theta_D} v(\theta_G, \theta_D)$ is convex in θ_D, the procedure is guaranteed to converge.
- **Cons**:
  - Learning in GANs can be difficult in practice when G and D are represented by neural networks and $\max_{\theta_D} v(\theta_G, \theta_D)$ is not convex.
  - In general, simultaneous gradient descent on two players' costs is not guaranteed to reach an equilibrium.

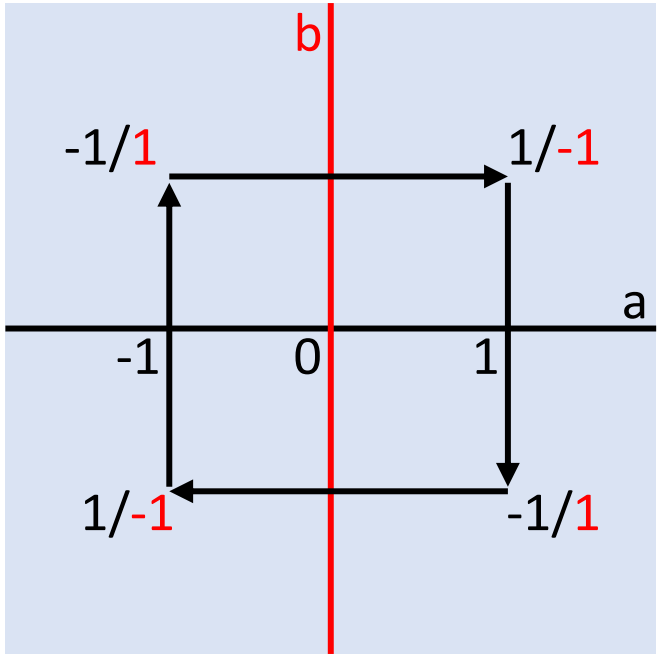
- **Pros**:
  - Learning process does not require approximations such as variational inference.
  - When $\max_{\theta_D} v(\theta_G, \theta_D)$ is convex in θ_D, the procedure is guaranteed to converge.
- **Cons**:
  - Learning in GANs can be difficult in practice when G and D are represented by neural networks and $\max_{\theta_D} v(\theta_G, \theta_D)$ is not convex.
  - In general, simultaneous gradient descent on two players' costs is not guaranteed to reach an equilibrium.

- **Pros**:
  - Learning process does not require approximations such as variational inference.
  - When $\max_{\theta_D} v(\theta_G, \theta_D)$ is convex in θ_D, the procedure is guaranteed to converge.
- **Cons**:
  - Learning in GANs can be difficult in practice when G and D are represented by neural networks and $\max_{\theta_D} v(\theta_G, \theta_D)$ is not convex.
  - In general, simultaneous gradient descent on two players' costs is not guaranteed to reach an equilibrium.

- **Pros**:
  - Learning process does not require approximations such as variational inference.
  - When $\max_{\theta_D} v(\theta_G, \theta_D)$ is convex in θ_D, the procedure is guaranteed to converge.
- **Cons**:
  - Learning in GANs can be difficult in practice when G and D are represented by neural networks and $\max_{\theta_D} v(\theta_G, \theta_D)$ is not convex.
  - In general, simultaneous gradient descent on two players' costs is not guaranteed to reach an equilibrium.

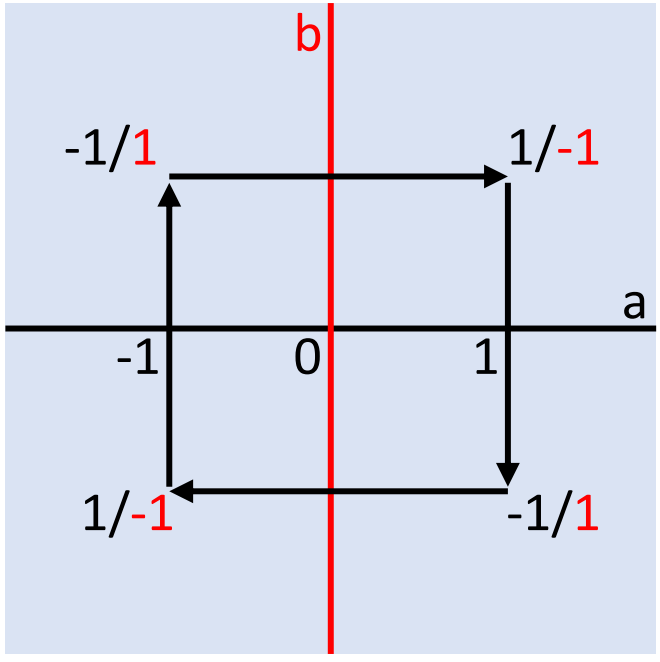
Example for convergence problems in zero-sum games:

- Value function $v(a, b) = ab$, where one player controls a and receives value ab, while the other player controls b and receives a value $-ab$.
- Each player makes gradient steps, increasing their own value at the expense of the other player
- a and b can go into a stable, circular orbit, rather than arriving at the equilibrium point at the origin.



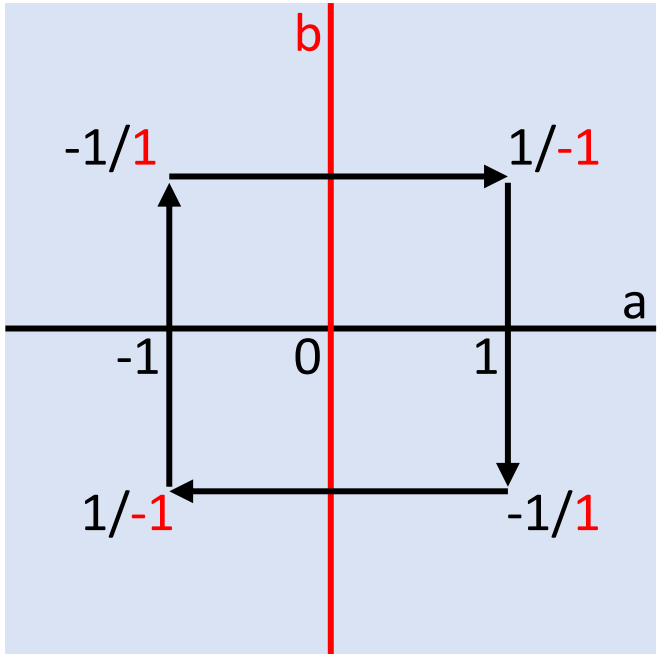
Example for convergence problems in zero-sum games:

- Value function $v(a, b) = ab$, where one player controls a and receives value ab, while the other player controls b and receives a value $-ab$.
- Each player makes gradient steps, increasing their own value at the expense of the other player
- a and b can go into a stable, circular orbit, rather than arriving at the equilibrium point at the origin.



Example for convergence problems in zero-sum games:

- Value function $v(a, b) = ab$, where one player controls a and receives value ab, while the other player controls b and receives a value $-ab$.
- Each player makes gradient steps, increasing their own value at the expense of the other player
- a and b can go into a stable, circular orbit, rather than arriving at the equilibrium point at the origin.



- The equilibria for a minimax game are not local minima of v, but the points that are simultaneously minima for both players' costs.
  - $\rightarrow$ Equilibrium points are saddle points of v (local minima wrt player 1 parameters and local maxima wrt player 2 parameters).
  - $\rightarrow$ There are oscillatory solutions that never relax to a saddle point.
- Dropout seems to be important in the discriminator network.
- Improving the convergence of GANs is a **very active** area of research – since 2013 hundreds of papers have been written on that.

- The equilibria for a minimax game are not local minima of v, but the points that are simultaneously minima for both players' costs.
  - $\rightarrow$ Equilibrium points are saddle points of v (local minima wrt player 1 parameters and local maxima wrt player 2 parameters).
  - $\rightarrow$ There are oscillatory solutions that never relax to a saddle point.
- Dropout seems to be important in the discriminator network.
- Improving the convergence of GANs is a **very active** area of research – since 2013 hundreds of papers have been written on that.

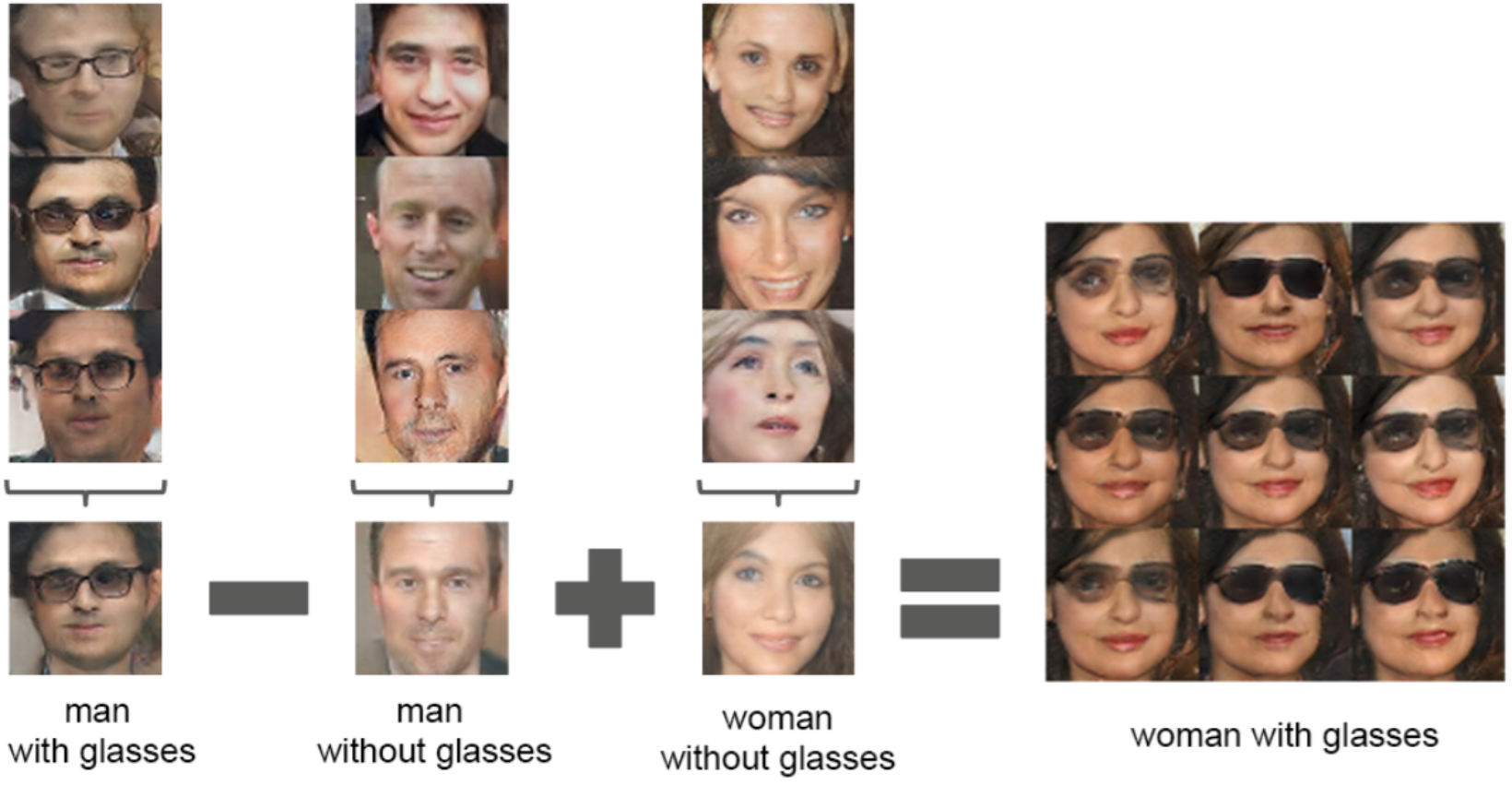
- The equilibria for a minimax game are not local minima of v, but the points that are simultaneously minima for both players' costs.
  - $\rightarrow$ Equilibrium points are saddle points of v (local minima wrt player 1 parameters and local maxima wrt player 2 parameters).
  - $\rightarrow$ There are oscillatory solutions that never relax to a saddle point.
- Dropout seems to be important in the discriminator network.
- Improving the convergence of GANs is a **very active** area of research – since 2013 hundreds of papers have been written on that.

- The equilibria for a minimax game are not local minima of v, but the points that are simultaneously minima for both players' costs.
  - $\rightarrow$ Equilibrium points are saddle points of v (local minima wrt player 1 parameters and local maxima wrt player 2 parameters).
  - $\rightarrow$ There are oscillatory solutions that never relax to a saddle point.
- Dropout seems to be important in the discriminator network.
- Improving the convergence of GANs is a **very active** area of research – since 2013 hundreds of papers have been written on that.

- The equilibria for a minimax game are not local minima of v, but the points that are simultaneously minima for both players' costs.
  - $\rightarrow$ Equilibrium points are saddle points of v (local minima wrt player 1 parameters and local maxima wrt player 2 parameters).
  - $\rightarrow$ There are oscillatory solutions that never relax to a saddle point.
- Dropout seems to be important in the discriminator network.
- Improving the convergence of GANs is a **very active** area of research – since 2013 hundreds of papers have been written on that.

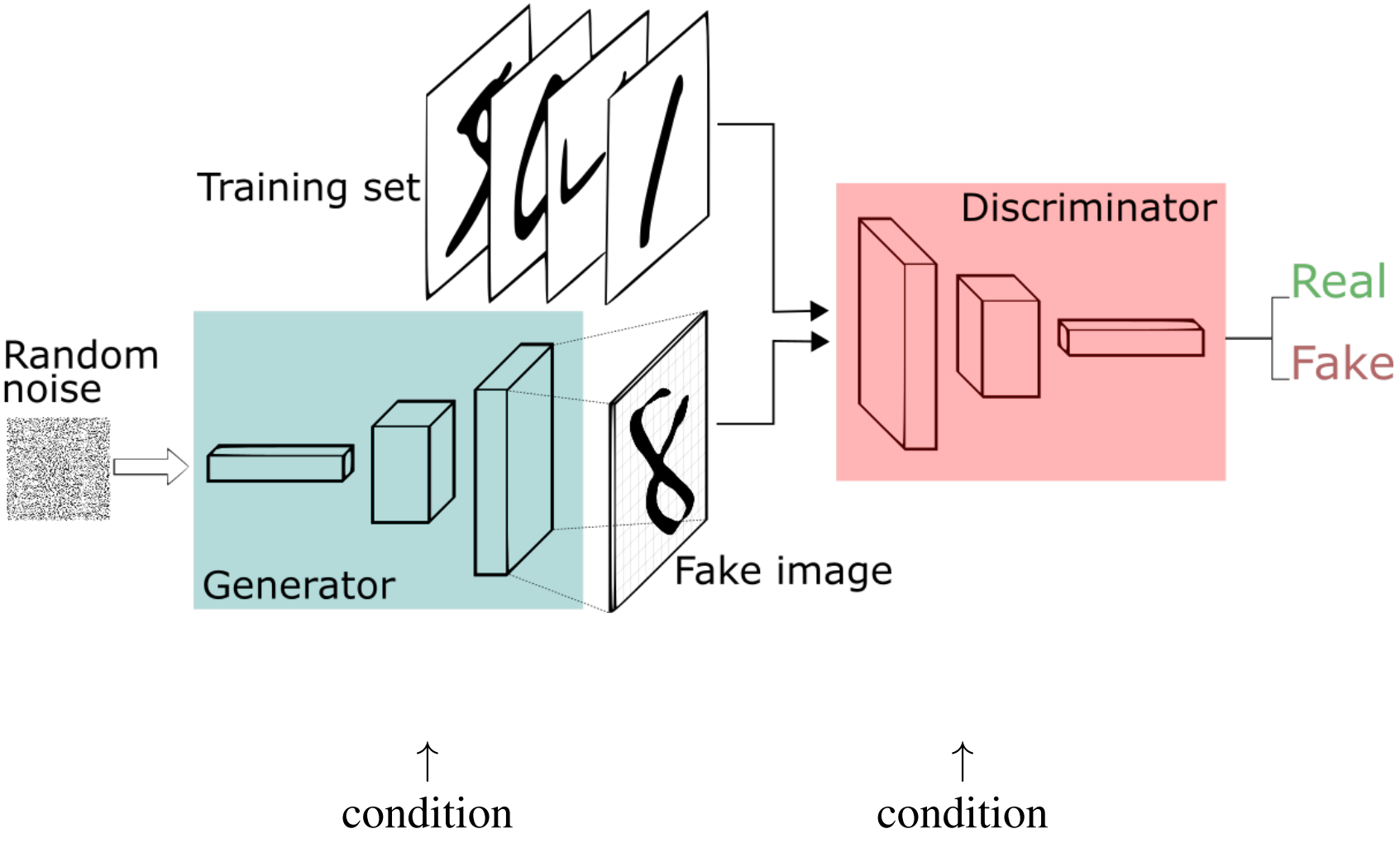
Generative Adversarial Network

GAN (Goodfellow et al., 2014)



Conditional GAN

https://arxiv.org/pdf/1411.1784.pdf



Plug & Play Generative Networks:
Nguyen et al, 2016

Generative Face Completion

Li et al, 2017

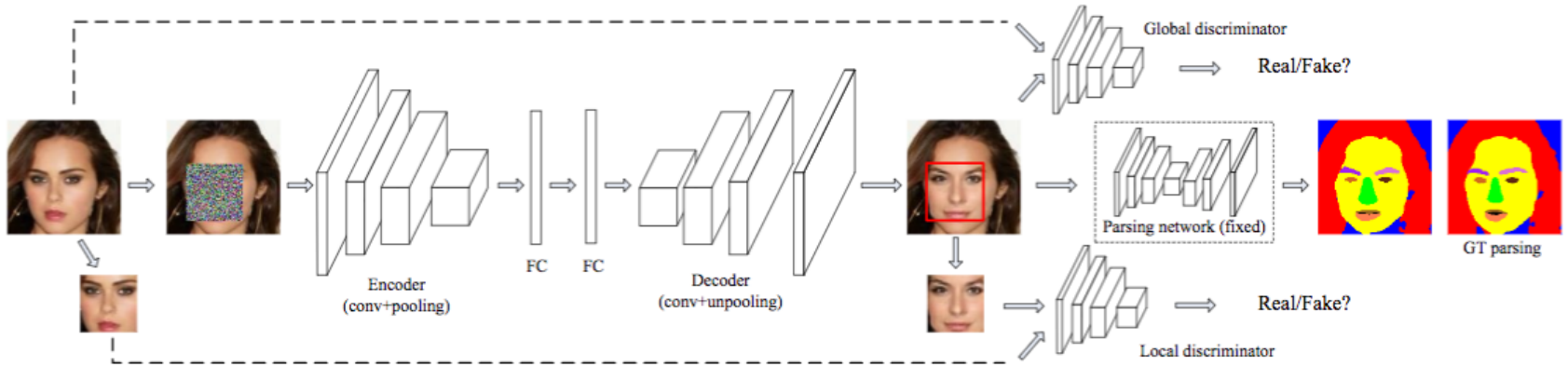


Image-to-Image Translation

Isola et al, 2017

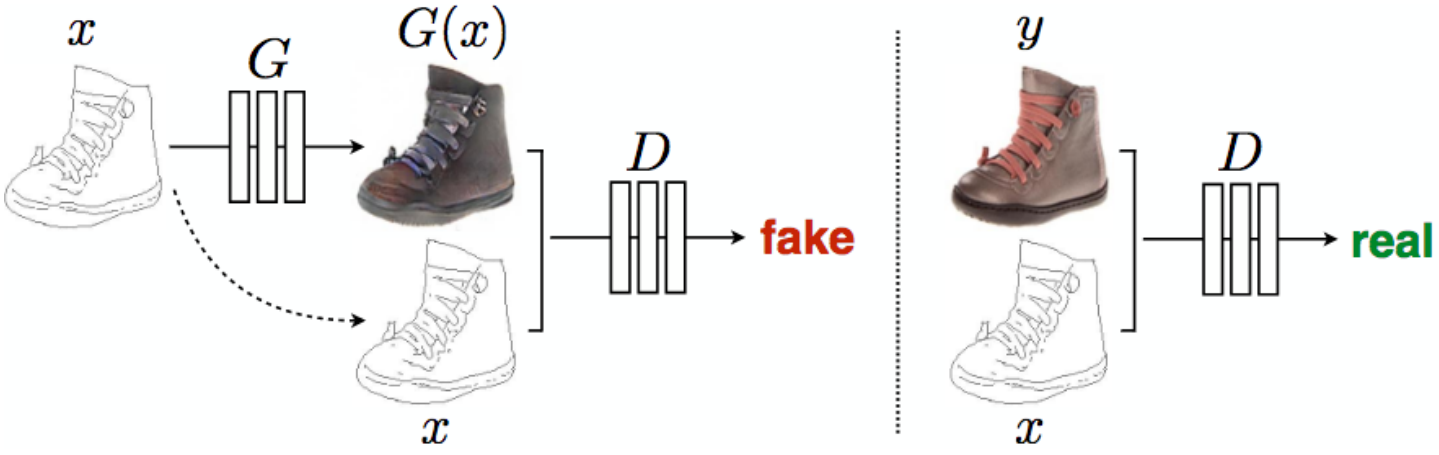


Image-to-Image Translation

Isola et al, 2017

Image-to-Image Translation

Isola et al, 2017

Progressive growing of GANs

Karras et al, 2018

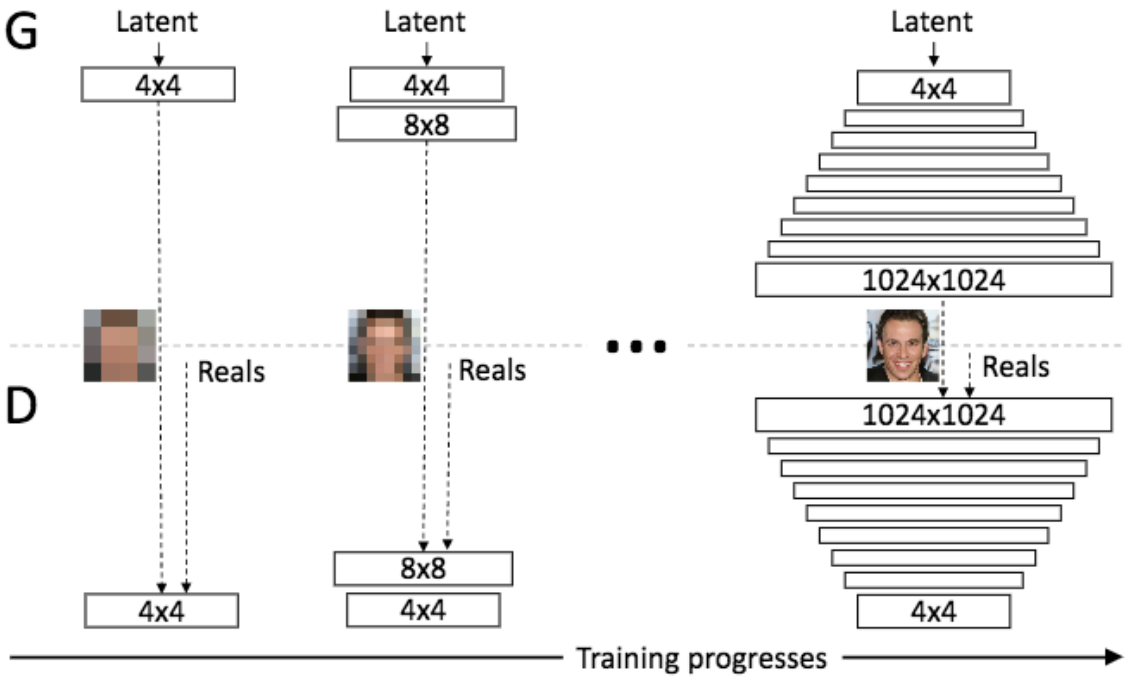


Image-to-Image Translation

Karras et al, 2018

Image-to-Image Translation

Karras et al, 2018

Image-to-Image Translation

Karras et al, 2018