香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

COMP 5212
Machine Learning
Lecture 22

Generative Adversarial Networks, Reinforcement Learning

Junxian He

Nov 26, 2024

Announcement

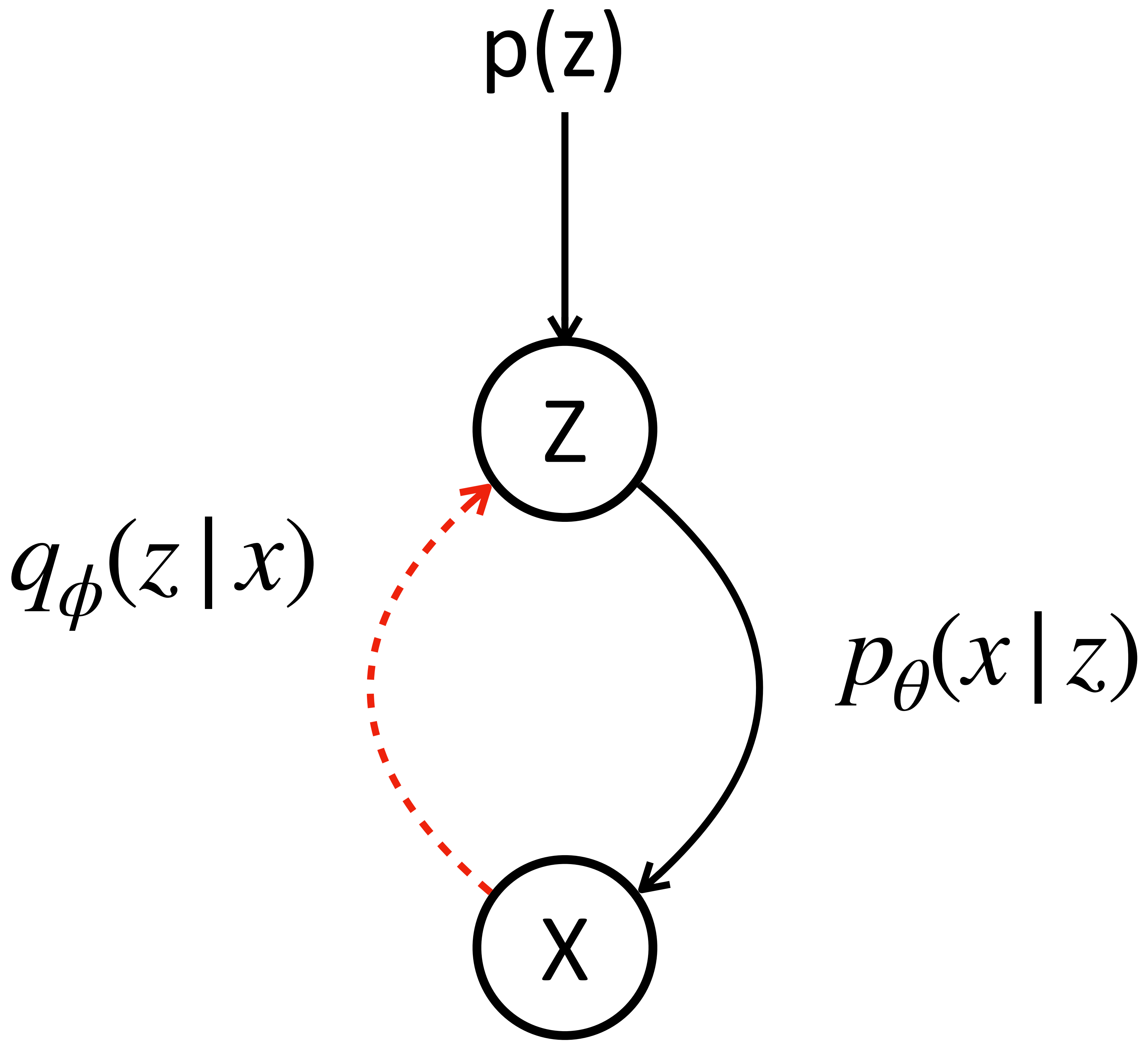
- HW4 is out, it is fairly easy, mainly a reflection of all the COMP5212 contents with only multi-choice questions

- The first round of Kaggle private leaderboard was released last night — do not overoptimize the public leaderboard too much

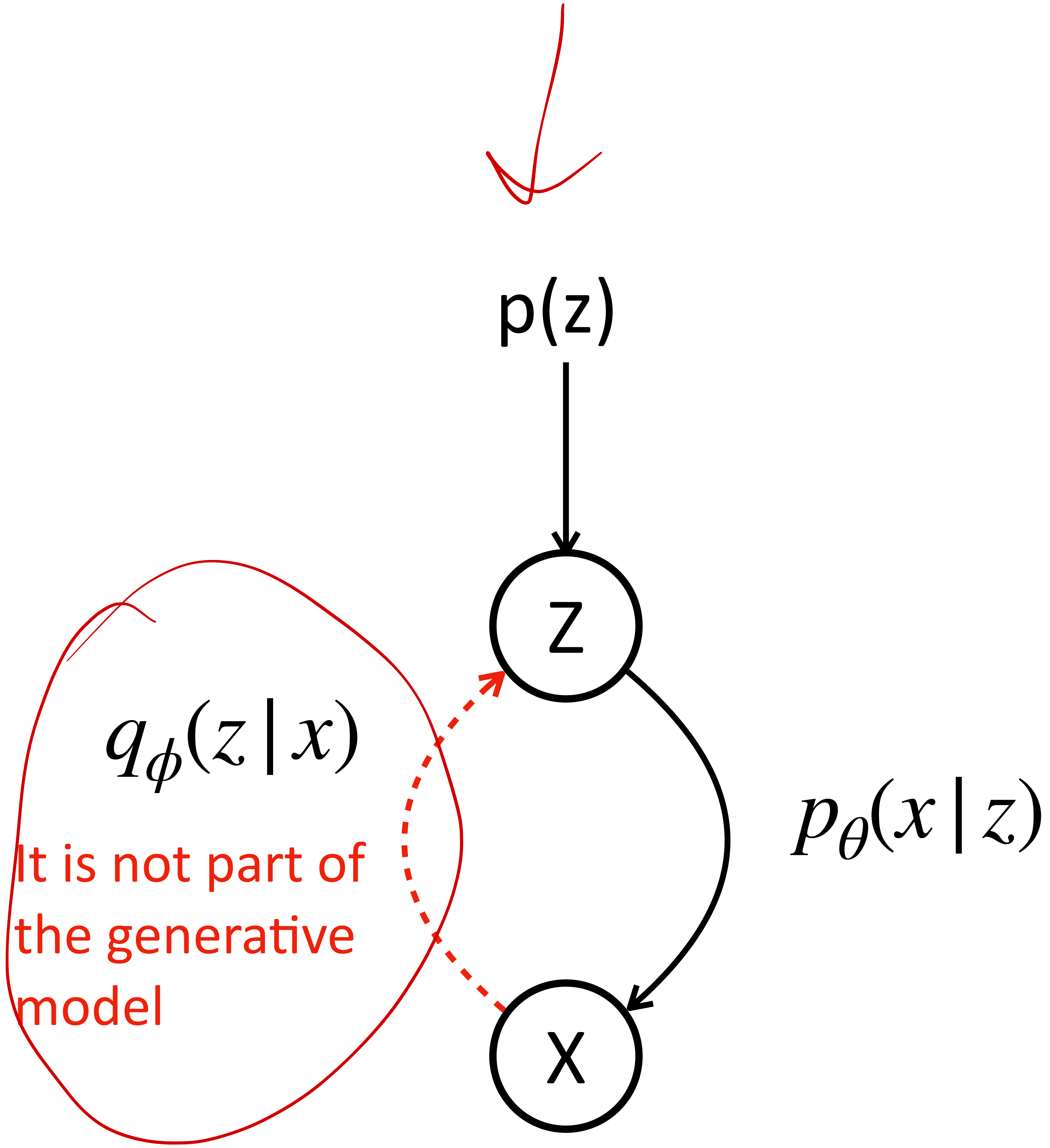
local valid

Pec

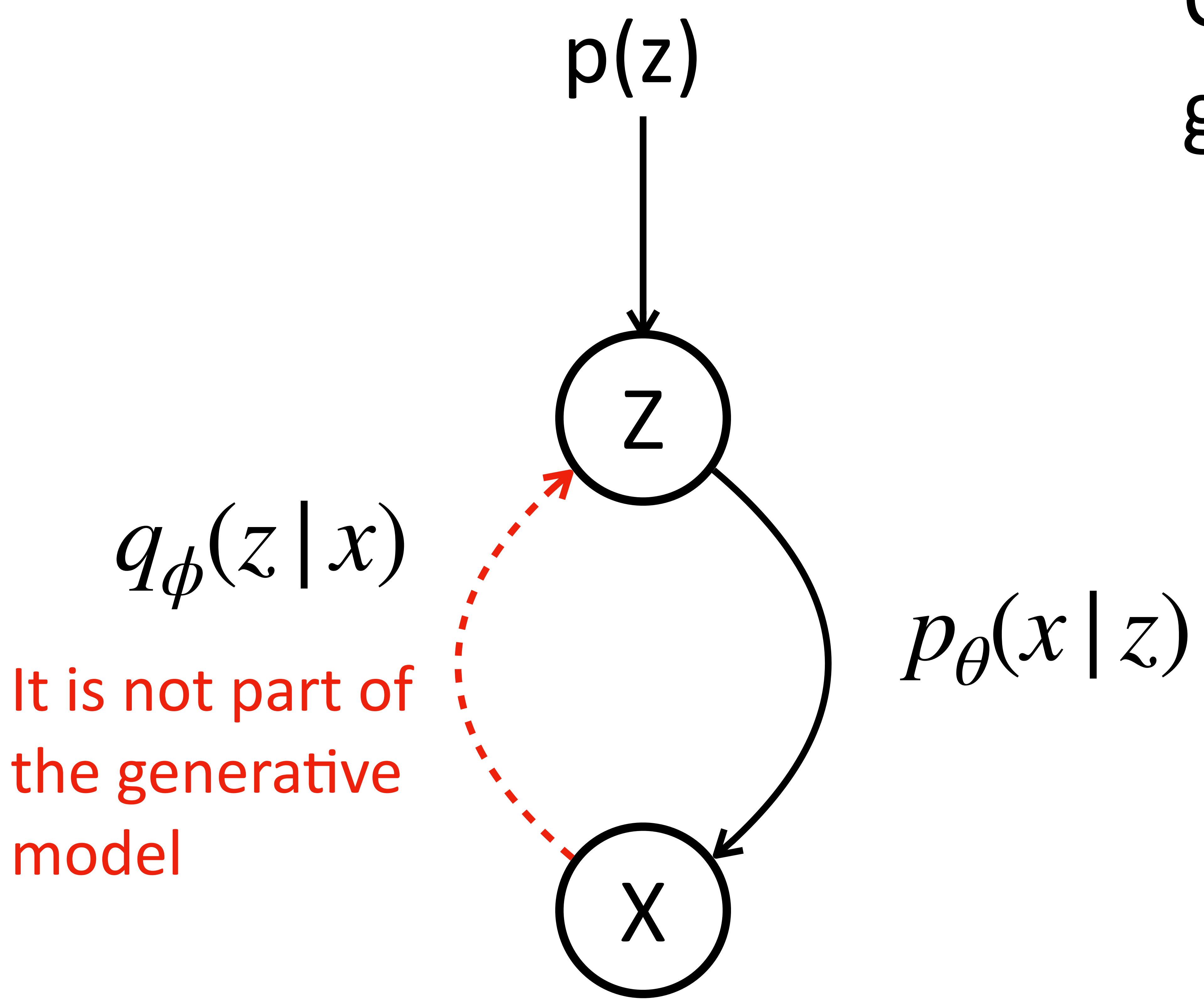
Recap: VAEs



Recap: VAEs

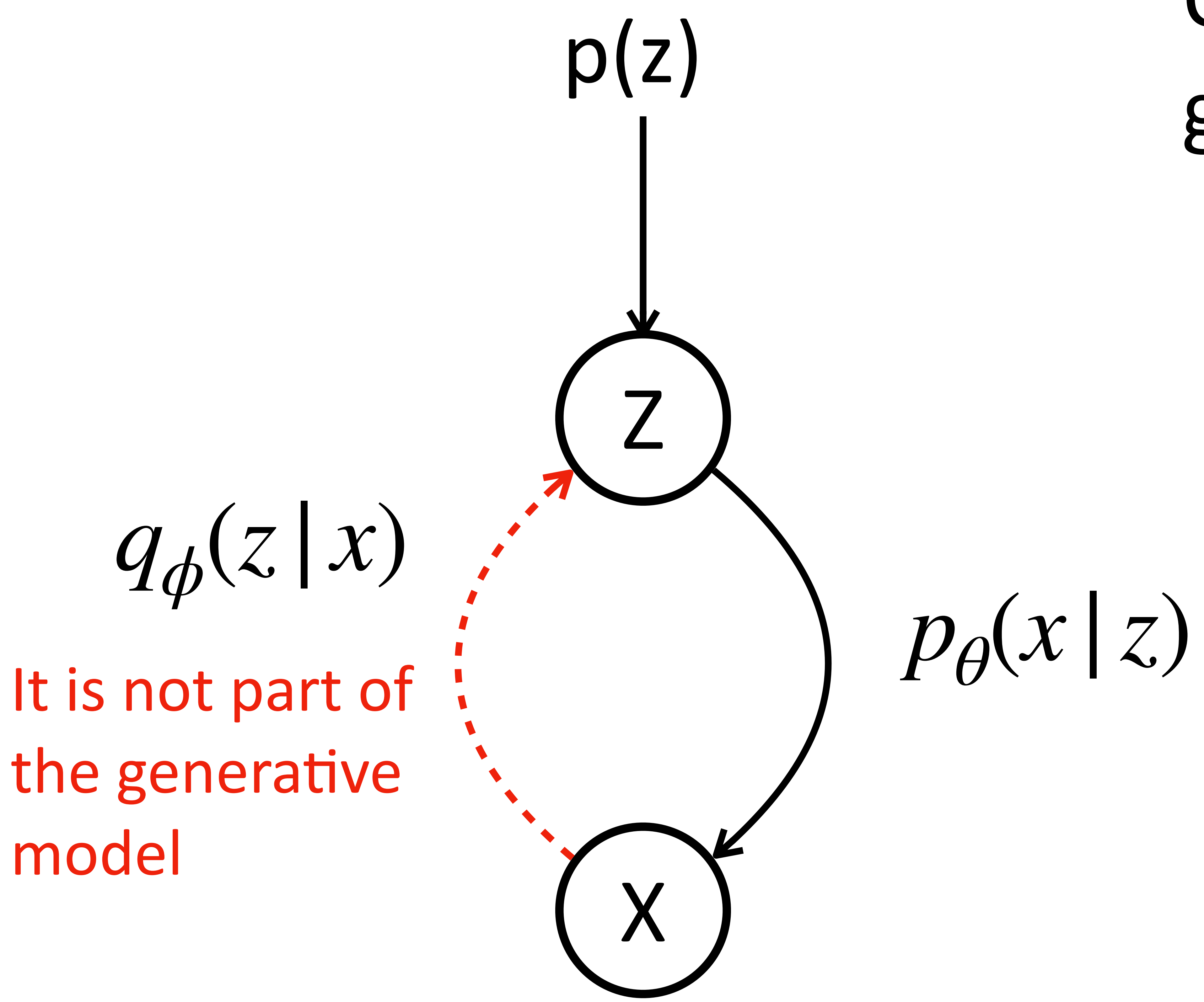


Recap: VAEs



Only the right (black) part defines the generative model, and the distribution

Recap: VAEs

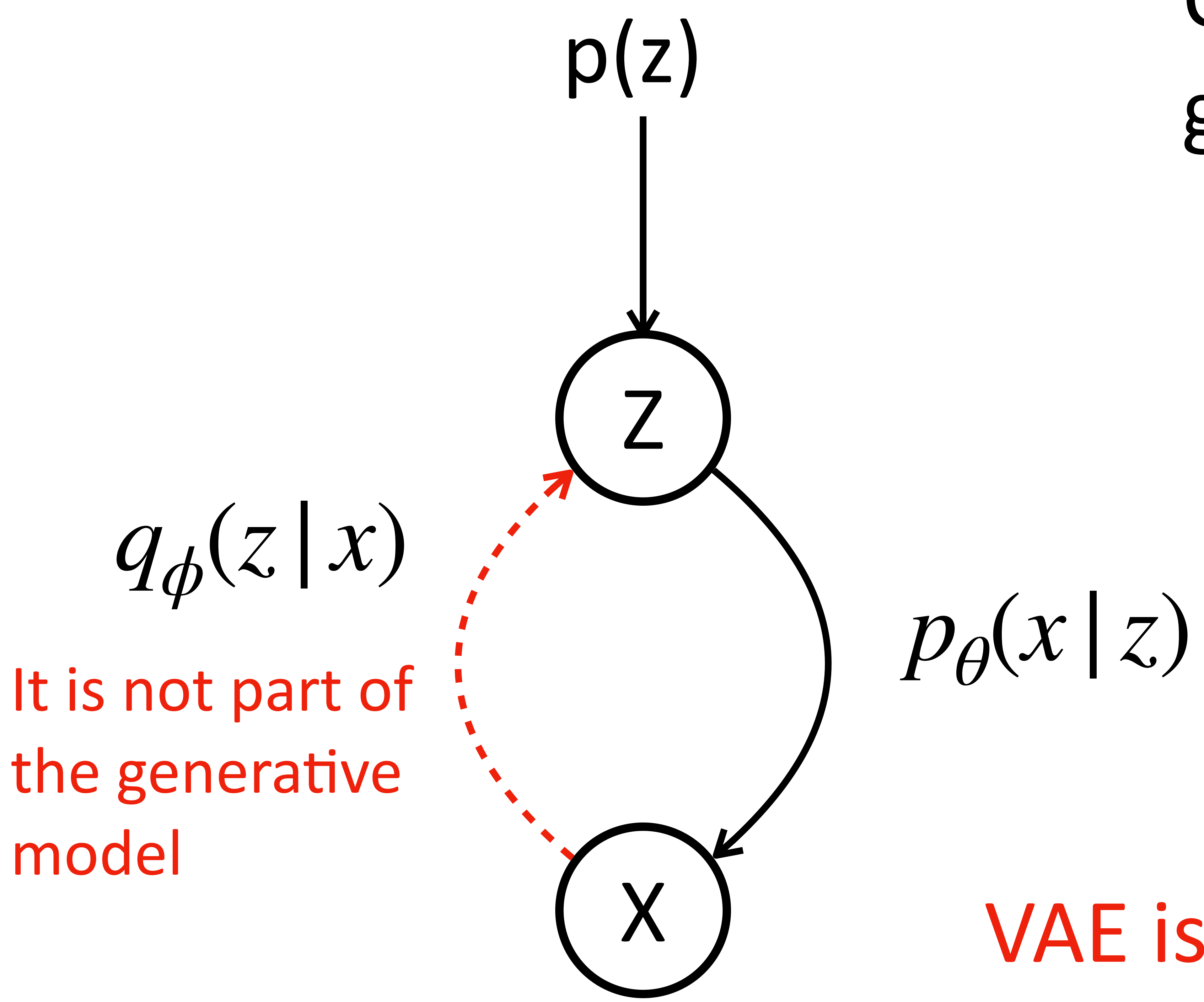


Only the right (black) part defines the generative model, and the distribution

$p_\theta(x|z)$: generative network/decoder

$q_\phi(z|x)$: inference network/encoder

Recap: VAEs



Only the right (black) part defines the generative model, and the distribution

$p_\theta(x|z)$: generative network/decoder

$q_\phi(z|x)$: inference network/encoder

VAE is a name to represent both the model p(x) and the inference network that is used to train the model, but do not confuse them together

Training VAEs

E-Step:

$$\operatorname{argmax}_{\phi} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

M-Step:

ELBO

$$\operatorname{argmax}_{\theta} \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

Training VAEs

E-Step:

$$\text{argmax}_\phi \quad \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

M-Step:

$$\text{argmax}_\theta \quad \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

Intuitively we hope to approximate p(z|x) with q(z|x) accurately in the E-step, to approximate the true EM algorithm

Training VAEs

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8)) ELBO
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

Training VAEs

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

Training VAEs

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$ ← re-parameter-trick
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}, \boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

End-to-end, because the objectives are the same (ELBO)

Training VAEs

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

End-to-end, because the objectives are the same (ELBO)

VAE training is optimizing ELBO with gradient descent

AutoEncoders

AutoEncoders

VAE: $$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

AutoEncoders

VAE: $$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

AE: $$\log p_{\theta}(x \mid q(x))$$

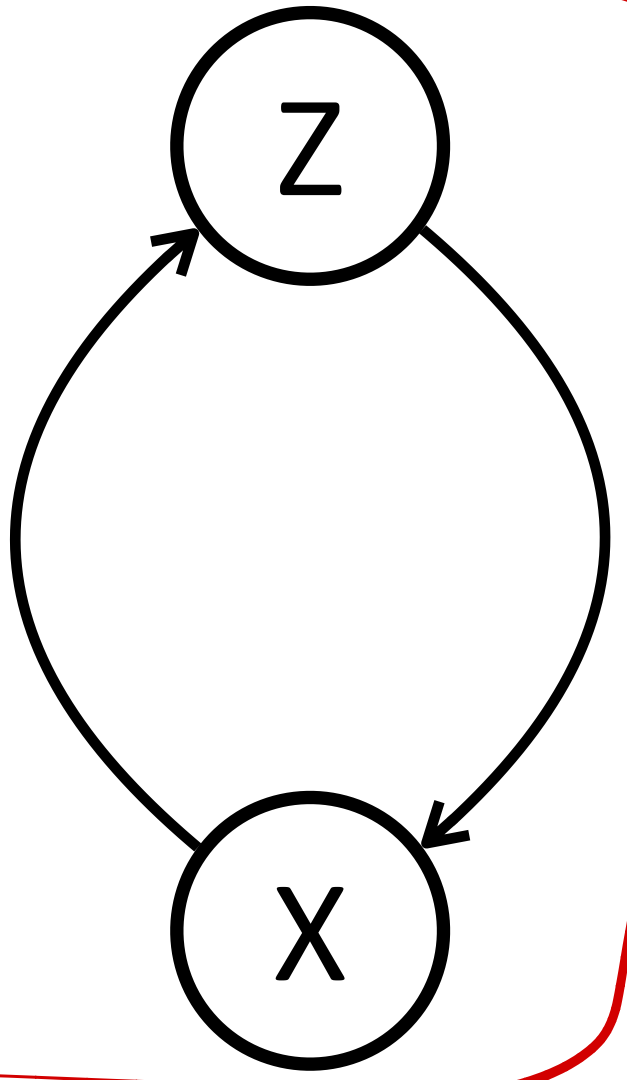
$z = q(x)$ deterministic

$z \sim q(z; f(x))$

AutoEncoders

VAE: $$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

AE: $$\log p_\theta(x \mid q(x))$$



AutoEncoders

AE is not generative model!

$p(x)$ $p(x,z)$

VAE: $$\underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol\theta}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{KL Regularizer}}$$

$N(0,1)$

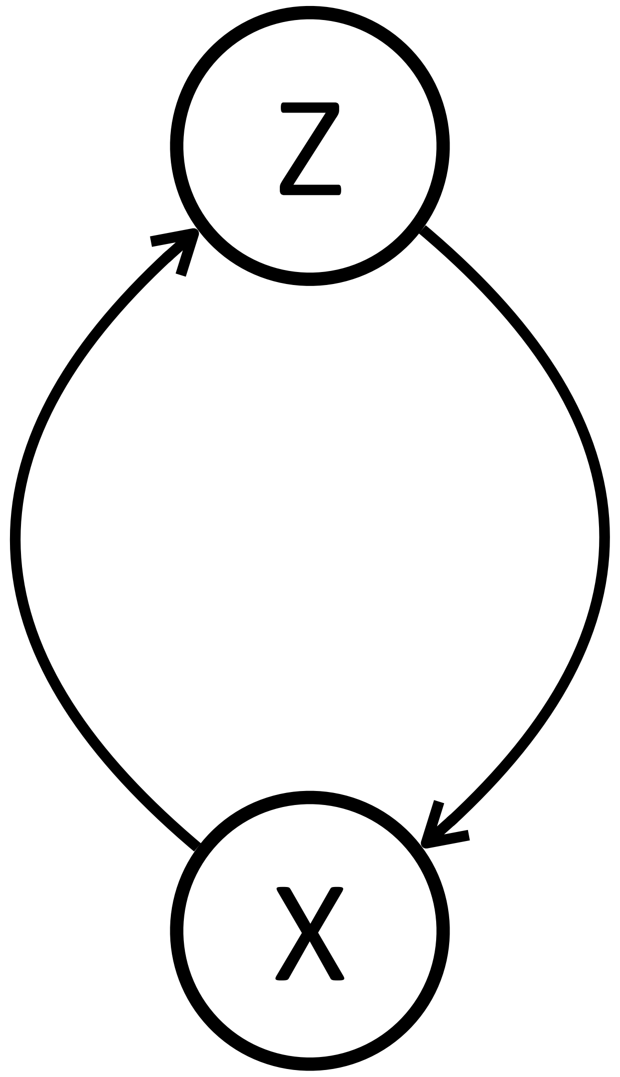
$p(z)$ $N(0,1)$

AE: $$\log p_\theta(x \mid q(x))$$

$p(x)$

$z \sim p(z)$

$x \sim p(x|z)$



No

1. Can we generate X samples from an autoencoder?
2. Can we approximate p(x) given x with an autoencoder?
3. What is the difference between the representation space from AE and VAE?

z

$p(z)$

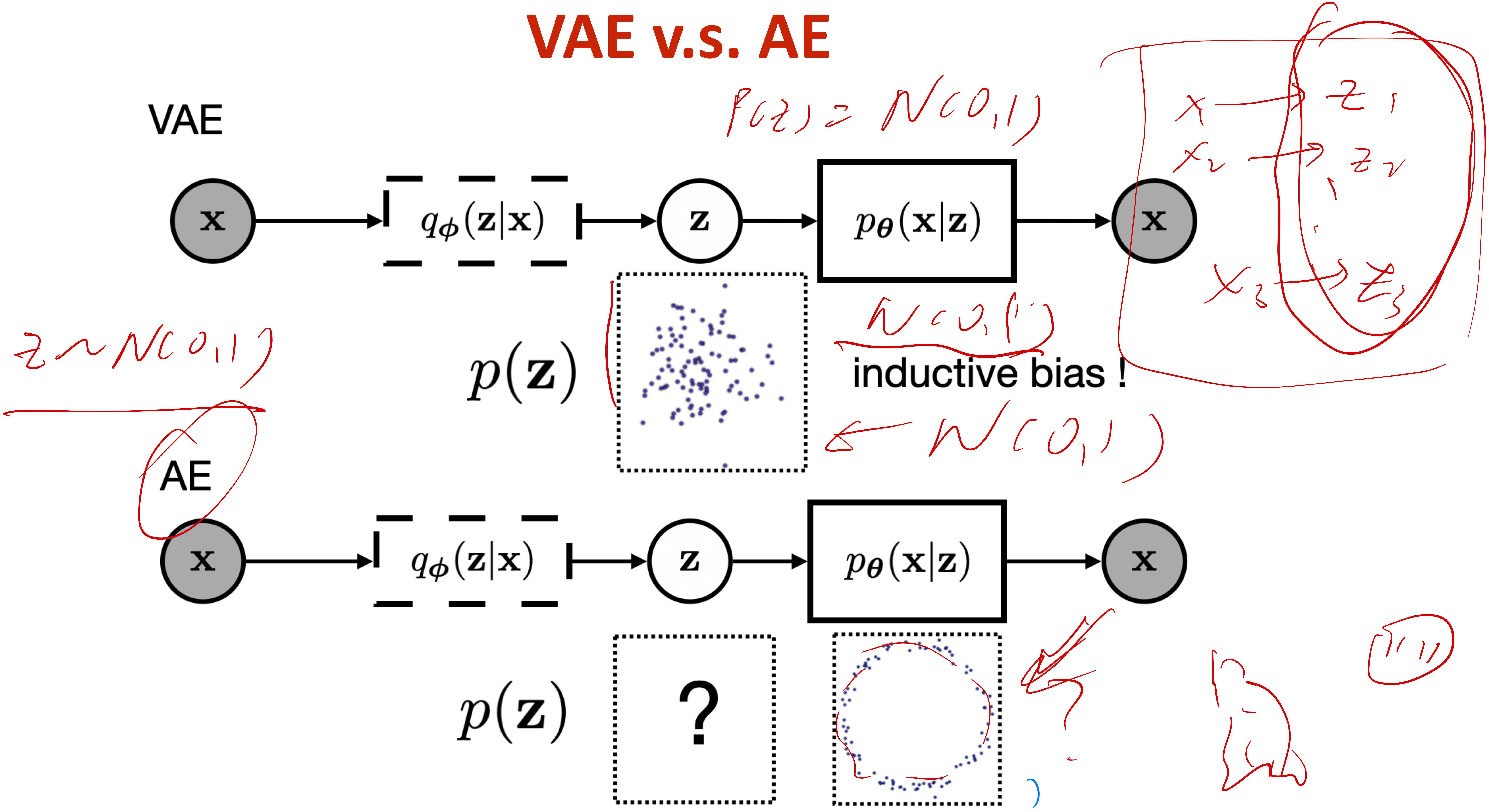
discriminative:

$p(z|x)$ $p(y|x)$

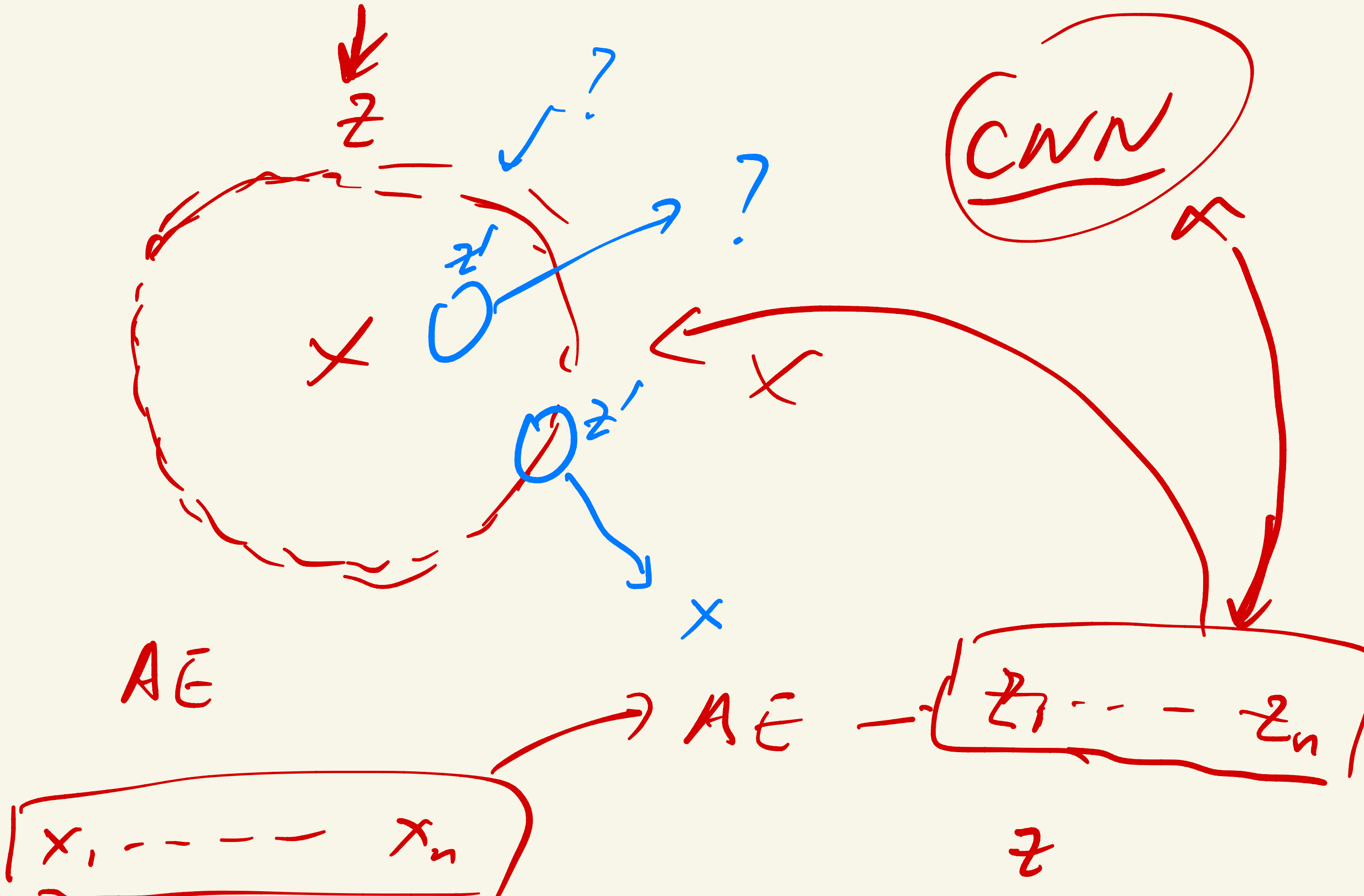
$z \sim N(0,1)$

AE: z?

VAE v.s. AE



VAE
$\mathbf{x}$
$q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}$
$p(\mathbf{z})$
inductive bias !
AE
$\mathbf{x}$
$q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}$
$p(\mathbf{z})$
?
p(z) = N(0,1)
z ~ N(0,1)
N(0,1)
N(0,1)
$x_1 \to z_1$
$x_2 \to z_2$
$x_3 \to z_3$


z
?
z'
?
x
z'
x
x
AE
CNN
AE
z_1 - - - z_n
x_1 - - - x_n
z

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie,[*] Mehdi Mirza, Bing Xu, David Warde-Farley,
Sherjil Ozair,[†] Aaron Courville, Yoshua Bengio[‡]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Generative Adversarial Networks

MLE

2017

VAEs

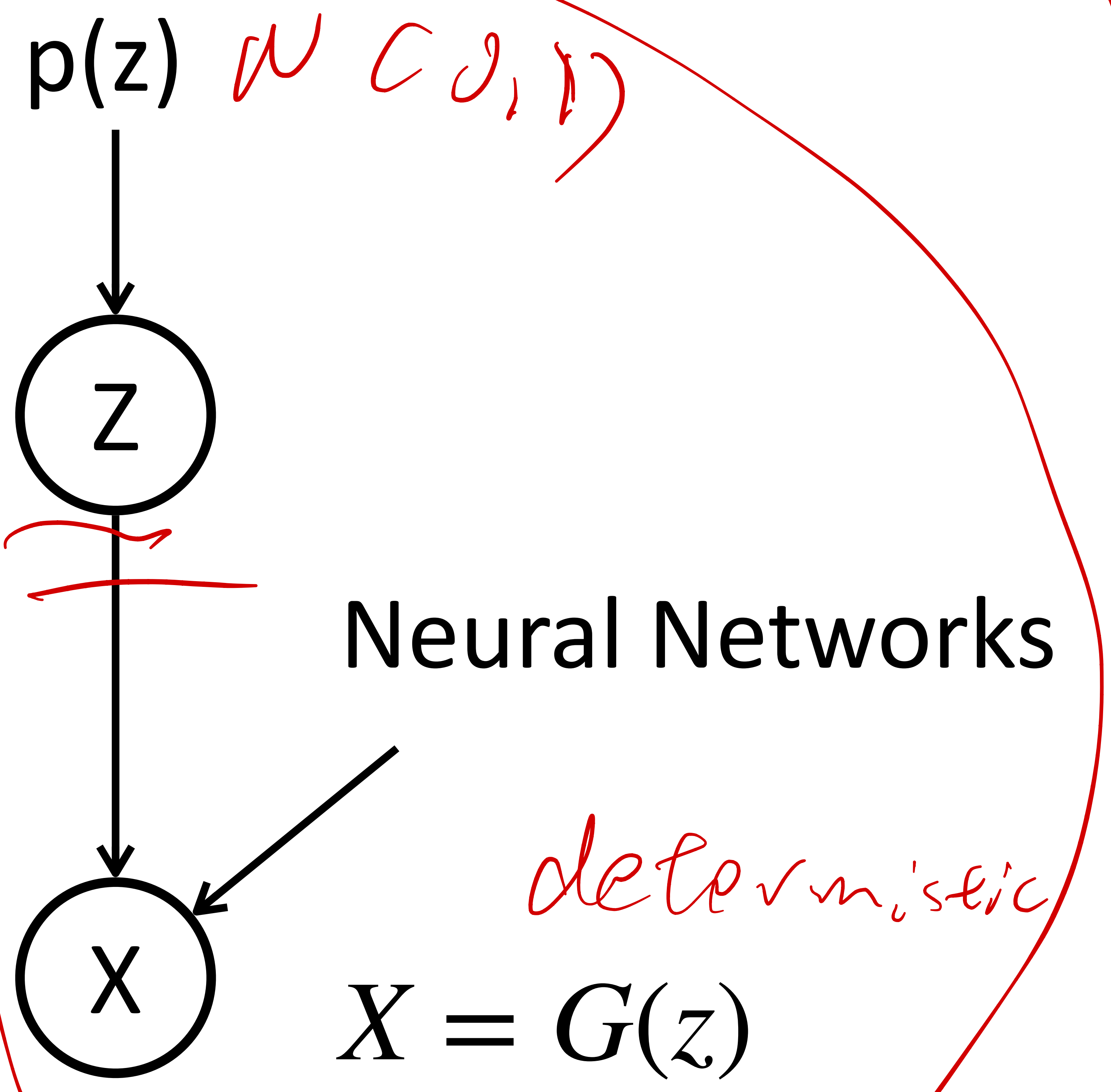
GANs

The GAN Model

$[P(x|z)] \sim N$

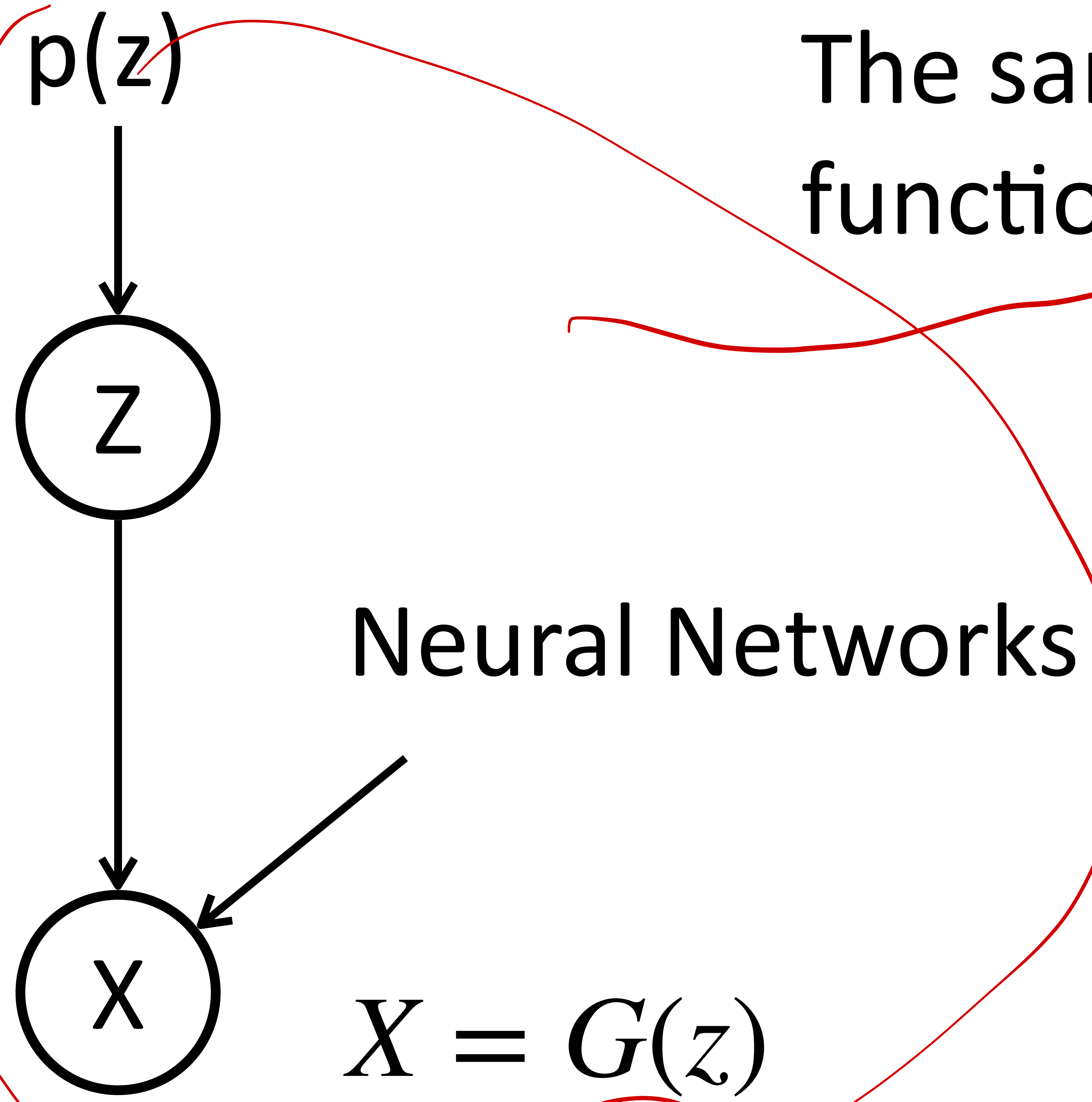
VAE: $x \sim N(\mu(z), \Sigma(z))$

GAN: $X = G(z; \theta)$



$P(x|z)$

The GAN Model



$$X = G(z)$$

The same as the VAE model, except that x is a deterministic function of z, but it can be a distribution as well

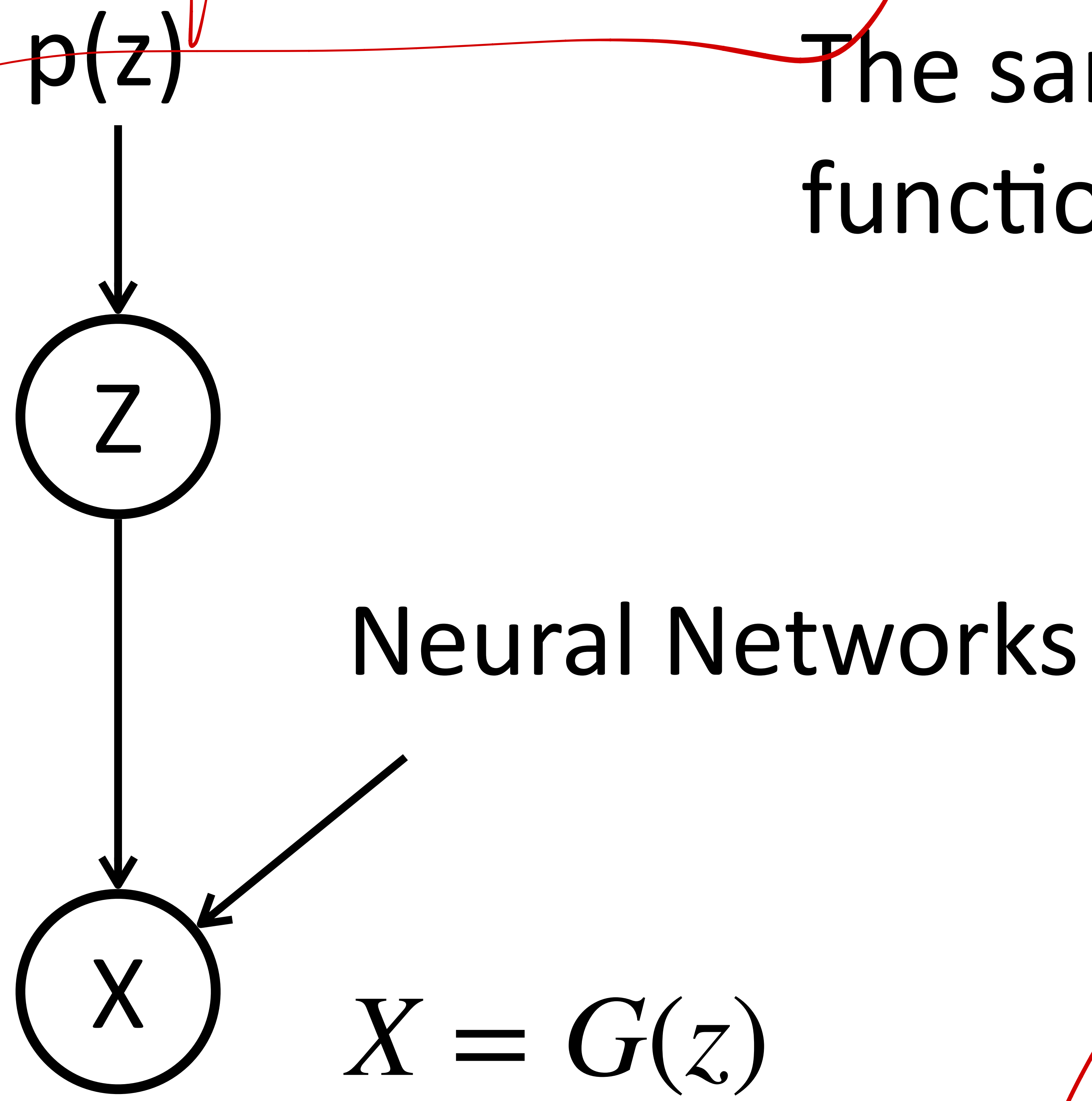
$x \sim N(\mu(z), \sigma^2(z))$

The GAN Model

$x_1 \to z$

$x_2 \to z$

$x = G(z)$ $\quad$ $z = G^{-1}$



$$X = G(z)$$

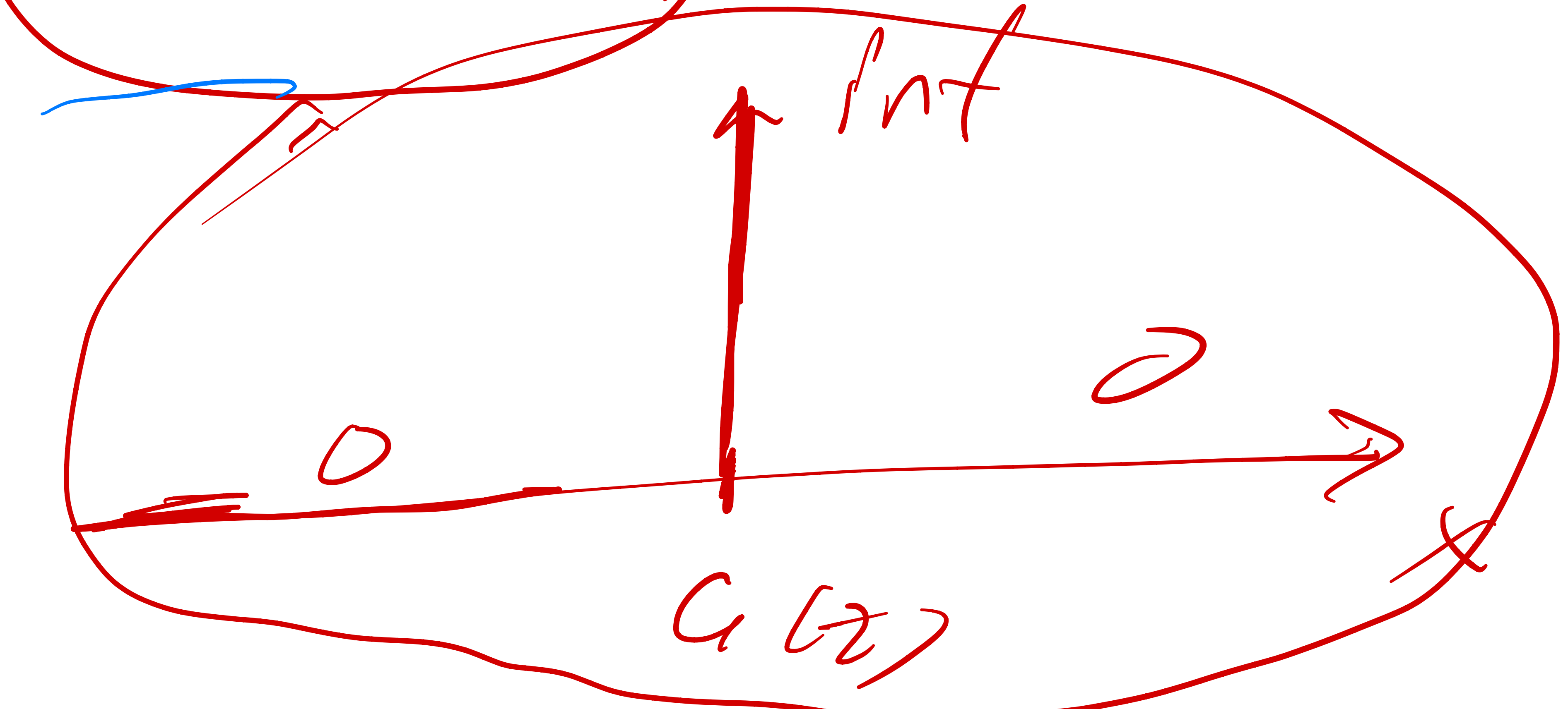
The same as the VAE model, except that x is a deterministic function of z, but it can be a distribution as well

Generative flow model

Can VAE use a deterministic x = G(z)?

$(x = G(z))$ $\quad$ $z \sim N(0,1)$

$p(x|z)$



$z = G^{-1}(x)$

flow:

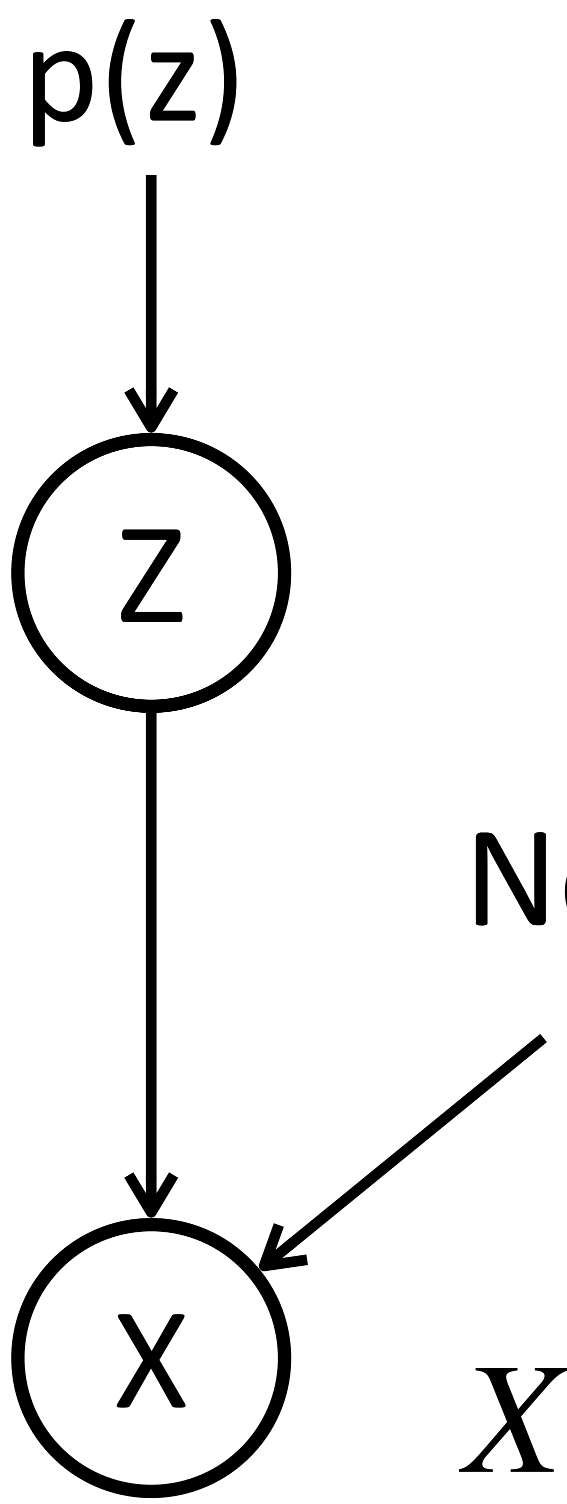
$p(z)$

z

$x \quad x = G(z)$

$\boxed{z = G^{-1}(x)}$

The GAN Model



Neural Networks

$X = G(z)$

The same as the VAE model, except that x is a deterministic function of z, but it can be a distribution as well

Can VAE use a deterministic x = G(z)?

explicit gen—

VAEs

Sometimes we call GANs *implicit* generative models

$p(x)$

You can draw samples, but hard to evaluate p(x)

$z \sim N(0,1)$

$x = G(z)$

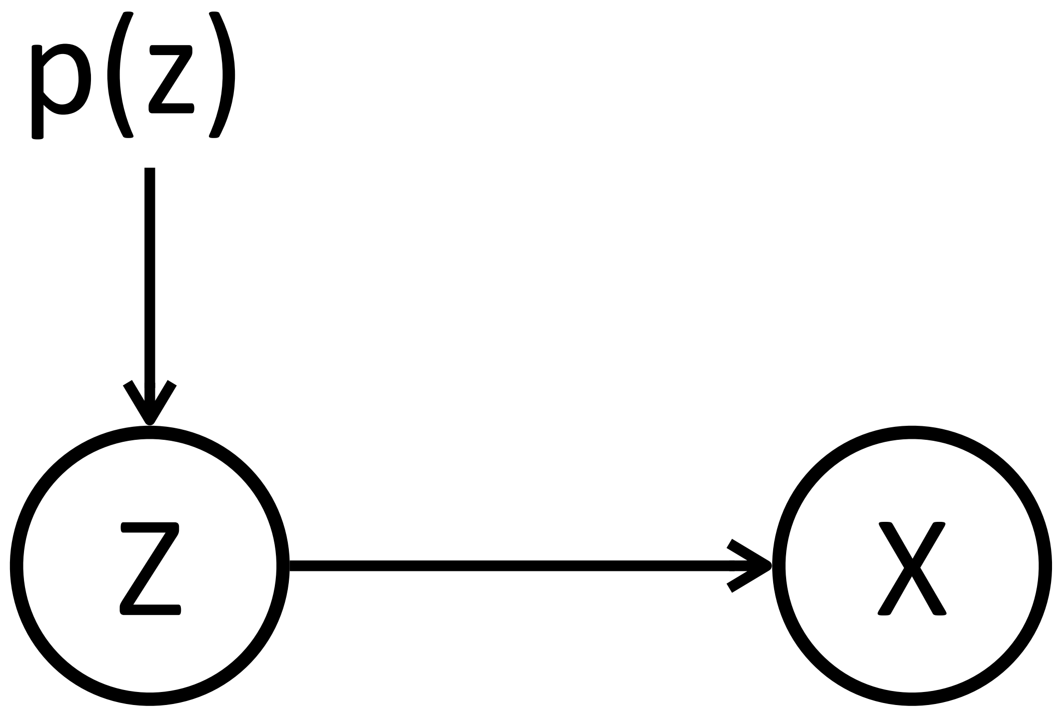
$x = G(z)$

x_0

$\boxed{ELBO} \leq \log p(x_0)$?

Training GANs

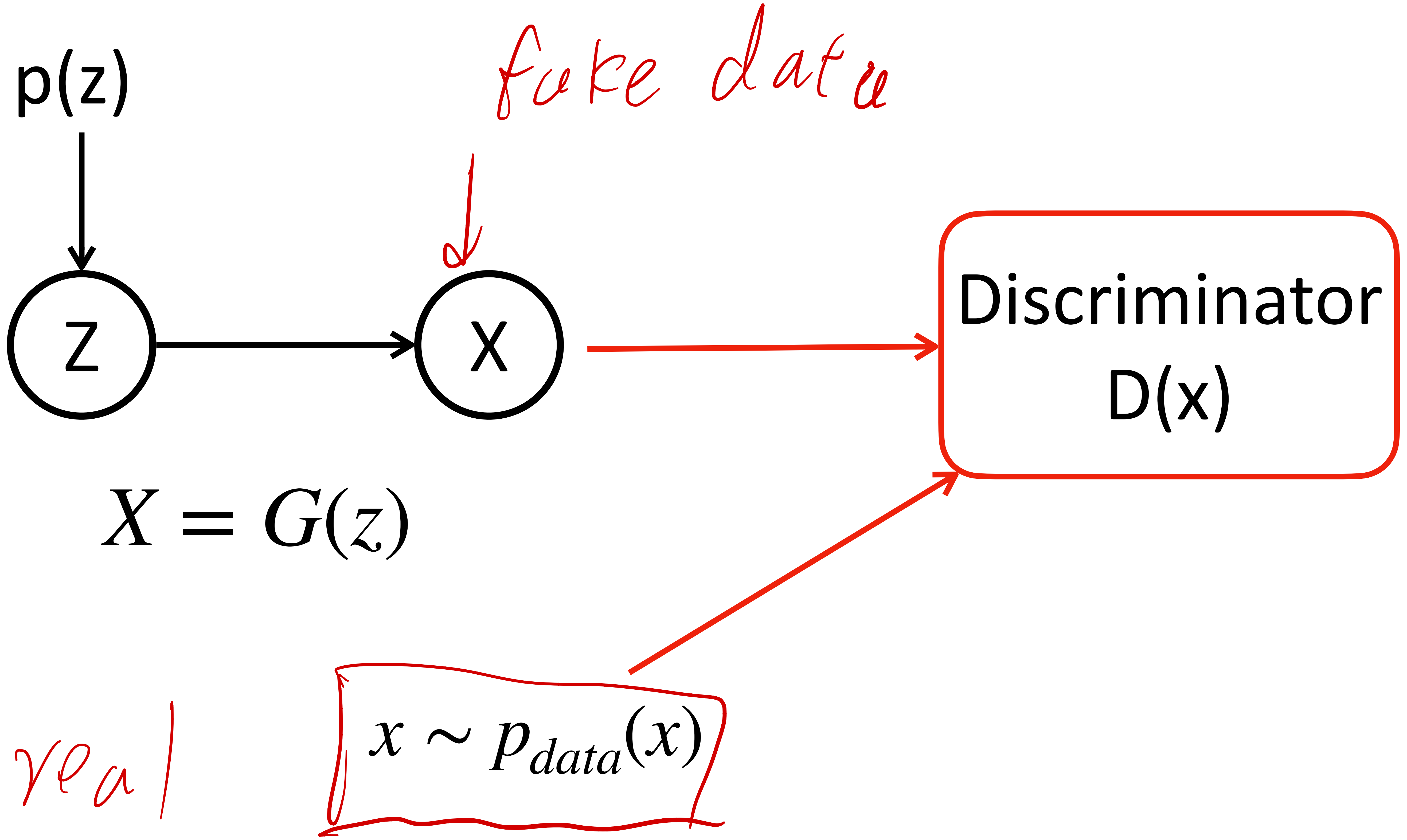
Computation Graph



$$X = G(z)$$

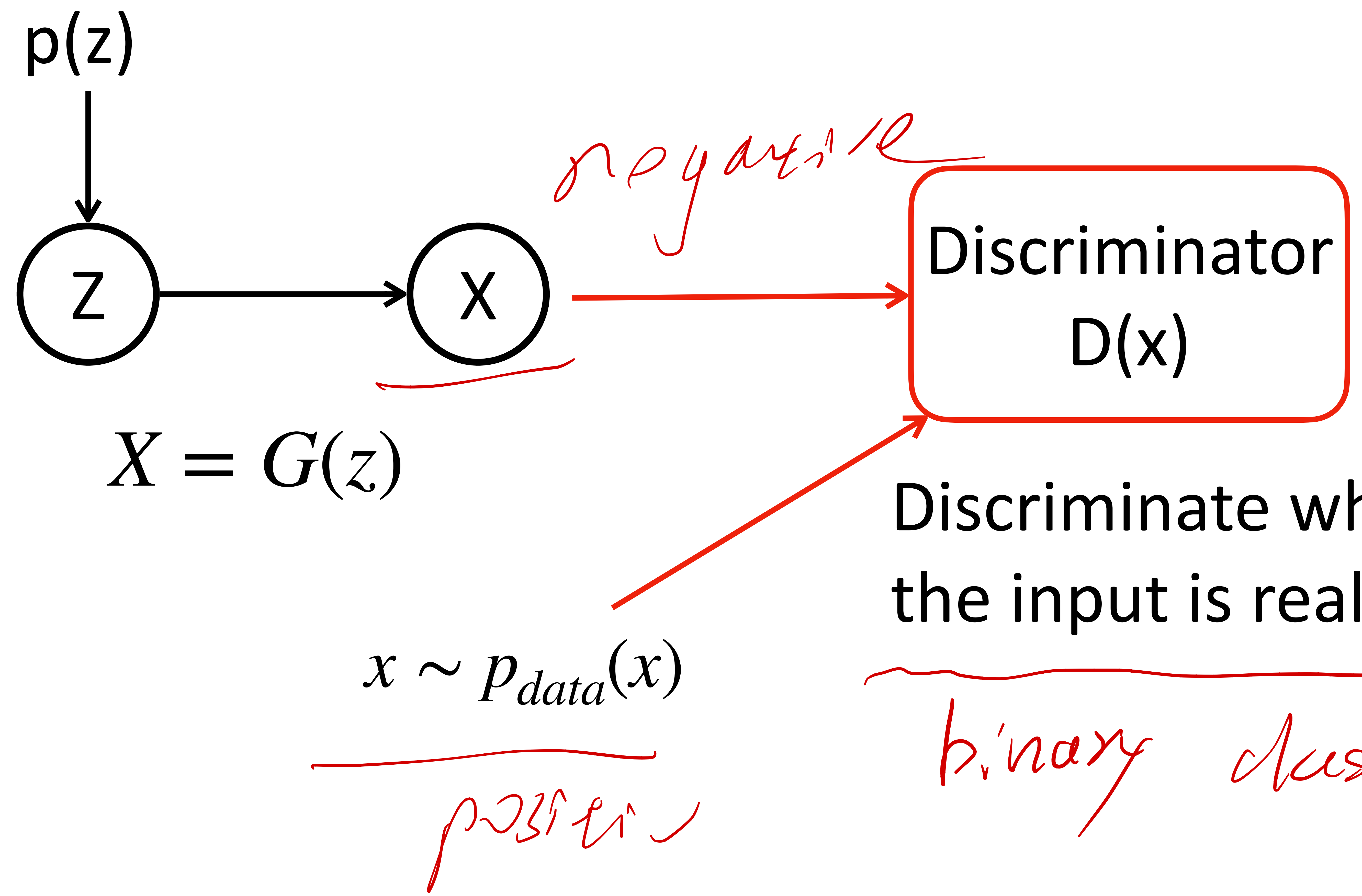
Training GANs

Computation Graph



Training GANs

Computation Graph

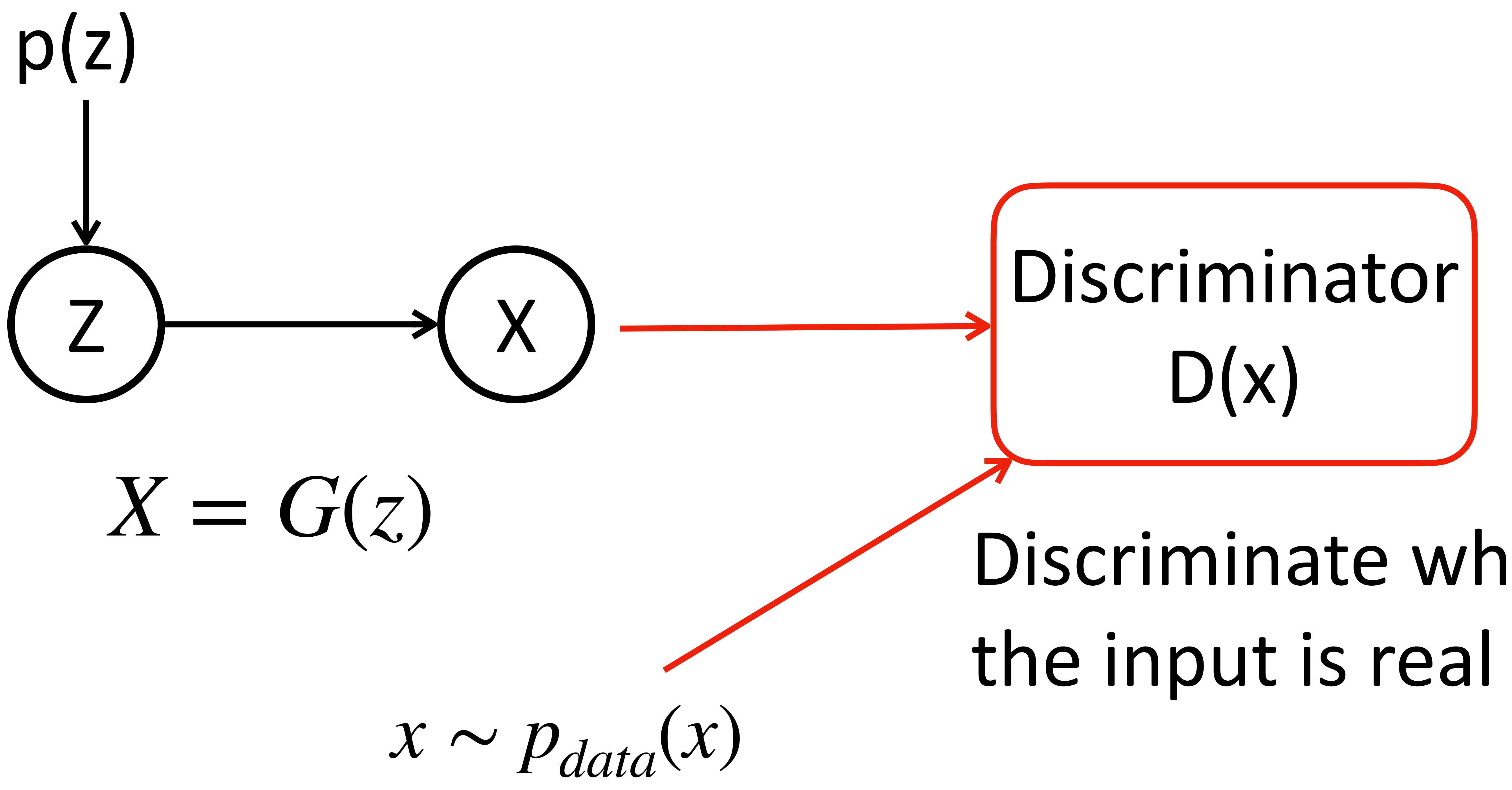


Discriminate whether the input is real or fake

binary classifier

Training GANs

Computation Graph

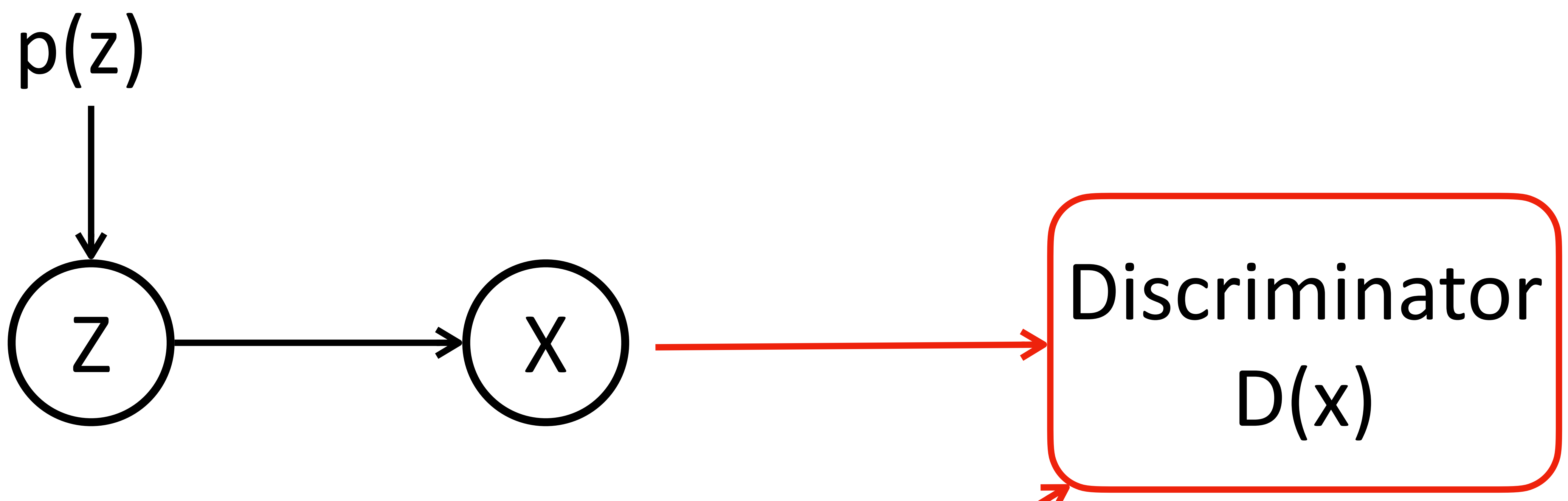


Discriminate whether
the input is real or fake

1. Generator is trained to produce realistic examples to fool the discriminator

Training GANs

Computation Graph



$$X = G(z)$$

Discriminate whether the input is real or fake

$$x \sim p_{data}(x)$$

1. Generator is trained to produce realistic examples to fool the discriminator
2. Discriminator is trained to discriminate real and fake examples

Training GANs

1. Generator is trained to produce realistic examples to fool the discriminator
2. Discriminator is trained to discriminate real and fake examples

Training GANs

1. Generator is trained to produce realistic examples to fool the discriminator
2. Discriminator is trained to discriminate real and fake examples

The two objectives are against each other

Adversarial Game

Generative Adversarial Net

Training GANs

1. Generator is trained to produce realistic examples to fool the discriminator
2. Discriminator is trained to discriminate real and fake examples

The two objectives are against each other

Adversarial Game

D(x) prob of x being real

P(x)

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

1 − D(G(z))

1 − D(G(z)): prob G(z) being fake

Training GANs

1. Generator is trained to produce realistic examples to fool the discriminator
2. Discriminator is trained to discriminate real and fake examples

The two objectives are against each other

Adversarial Game

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

Classification loss

Training GANs

1. Generator is trained to produce realistic examples to fool the discriminator
2. Discriminator is trained to discriminate real and fake examples

The two objectives are against each other

Adversarial Game

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

Classification loss

G(z) is trained to minimize the probability of G(z) recognized as “fake” by D

Training GANs

1. Generator is trained to produce realistic examples to fool the discriminator
2. Discriminator is trained to discriminate real and fake examples

The two objectives are against each other

Adversarial Game

optimization

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

Classification loss

G(z) is trained to minimize the probability of G(z) recognized as “fake” by D

D(x) is trained with a standard classification loss

Training GANs

1. GAN is a new algorithm to train a common generative model (VAE as well)
2. GAN training is not MLE

Theory of GANs

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

Theory of GANs

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

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

$\min_G \max_D$

G^*

$p_{data}(x)$

Theory of GANs

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

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

$$\begin{aligned} C(G) &= \max_D V(G,D) \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1-D_G^*(G(\boldsymbol{z})))] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1-D_G^*(\boldsymbol{x}))] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] \end{aligned}$$

Theory of GANs

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

Theory of GANs

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

Theorem 1. *The global minimum of the virtual training criterion* $C(G)$ *is achieved if and only if* $p_g = p_{data}$. *At that point,* $C(G)$ *achieves the value* $-\log 4$.

$P_g = P_{data}$

$P_g(x) = P_{data}(x)$

MLE: $\max E_{x \sim P_{data}(x)} \log P_g(x)$

MLE: $\arg\min_{P_g(x)} KL(P_{data}(x) \| P_g(x)) \geq 0$

$$\arg\max_{P_g(x)} E_{x\sim P_{data}(x)} \log P_g(x)$$

$P_g^*(x) = P_{data}(x)$? Jensen inequality

not

$$\arg\max_{P_g(x)} E_{x\sim P_{data}(x)} \log P_g(x) - E_{x\sim P_{data}(x)} \log P_{data}(x)$$

?

$$\Leftrightarrow \arg\min_{P_g(x)} E_{x\sim P_{data}} \log P_{data}(x) - E_{x\sim P_{data}(x)} \log P_g(x)$$

Theory of GANs

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

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) + KL\left(p_g \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right)$$

Theory of GANs

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

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) + KL\left(p_g \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right)$$

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

Theory of GANs

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

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) + KL\left(p_g \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right)$$

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data}

Theory of GANs

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

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right)$$

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

d

min max
G D

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data}

Training GANs

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

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

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

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

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Training GANs

Inner loop to update discriminator first

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
  - Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
  - Update the discriminator by ascending its stochastic gradient:

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

  end for

- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Update the generator by descending its stochastic gradient:

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

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

→ discriminator

Generator

?

Training GANs

Training GANs

1. GAN is a new algorithm to train a common generative model (like VAE)

Training GANs

1. GAN is a new algorithm to train a common generative model (like VAE)

2. GAN training is not MLE

Training GANs

1. GAN is a new algorithm to train a common generative model (like VAE)
2. GAN training is not MLE

What is it then?

$P_g(x) \longrightarrow P_{data}(x)$

?

MLE: $P_g(x) \longrightarrow P_{data}(x)$

$KL(P_{data} \| P_g(x))$

Training GANs

1. GAN is a new algorithm to train a common generative model (like VAE)
2. GAN training is not MLE

What is it then?

Suppose the generator G(x) is parameterized by θ, then what is the gradient when updating G(x)?

Training GANs

1. GAN is a new algorithm to train a common generative model (like VAE)
2. GAN training is not MLE

What is it then?

Suppose the generator G(x) is parameterized by θ, then what is the gradient when updating G(x)?

$$C(G) = -\log(4) + KL(p_{data} || \frac{p_{data} + p_g^*}{2}) + KL(p_g || \frac{p_{data} + p_g^*}{2})$$

Training GANs

1. GAN is a new algorithm to train a common generative model (like VAE)
2. GAN training is not MLE

What is it then?

Suppose the generator G(x) is parameterized by θ, then what is the gradient when updating G(x)?

θ

min max

G D

$$C(G) = -\log(4) + KL(p_{data}||\frac{p_{data}+p_g^*}{2}) + KL(p_g||\frac{p_{data}+p_g^*}{2})$$

p_g^* is from the solution of the discriminator, which is fixed when optimizing θ

Training GANs

$d(x_1, x_2) = d(x_2, x_1)$

$\|x_1 - x_2\|^2 = \|x_2 - x_1\|^2$

KL distance is not symmetric

1. GAN is a new algorithm to train a common generative model (like VAE)

2. GAN training is not MLE

What is it then?

Suppose the generator G(x) is parameterized by θ, then what is the gradient when updating G(x)?

$$C(G) = -\log(4) + KL(p_{data}||\frac{p_{data}+p_g^*}{2}) + KL(p_g||\frac{p_{data}+p_g^*}{2})$$

MLE:

p_g^* is from the solution of the discriminator, which is fixed when optimizing θ

$$\nabla_\theta C(G) = \nabla_\theta KL(p_g(x;\theta)||\frac{p_{data}+p_g^*(x)}{2})$$

$KL(P_{data}(x)||P_g(x))$

Training GANs

Training GANs

Recall that MLE is equivalent to minimizing $KL(p_{data}(x)||p_g(x))$

Training GANs

Recall that MLE is equivalent to minimizing $KL(p_{data}(x) || p_g(x))$

For GANs, the generator is to minimize $KL(p_g(x;\theta) || \frac{p_{data} + p_g^*(x)}{2})$

Training GANs

Recall that MLE is equivalent to minimizing $KL(p_{data}(x)\,||\,p_g(x))$

For GANs, the generator is to minimize $KL(p_g(x;\theta)\,||\,\frac{p_{data}+p_g^*(x)}{2})$

$$KL(p\,||\,q) \neq KL(q\,||\,p)$$

$KL(p_g(x)\,||\,p_{data}) = \mathbb{E}_{x\sim p_g(x)} \log p_g(x)$?

$- \mathbb{E}_{x\sim p_g(x)} \log p_{data}(x)$

Training GANs

Recall that MLE is equivalent to minimizing $KL(p_{data}(x)||p_g(x))$

For GANs, the generator is to minimize $KL(p_g(x;\theta)||\frac{p_{data}+p_g^*(x)}{2})$

$$\boxed{KL(p||q) \neq KL(q||p)}$$

KL divergence is asymmetric, and GANs' KL divergence is in the opposite direction with respect to MLE

GANs v.s. VAEs

Brock et al. LARGE SCALE GAN TRAINING FOR HIGH FIDELITY NATURAL IMAGE SYNTHESIS. ICLR 2019.

GANs v.s. VAEs

GANs are widely demonstrated to show superiority to VAEs on generating realistic, vivid images. In contrast, VAEs' generation is more blurred

Brock et al. LARGE SCALE GAN TRAINING FOR HIGH FIDELITY NATURAL IMAGE SYNTHESIS. ICLR 2019.

GANs v.s. VAEs

GANs are widely demonstrated to show superiority to VAEs on generating realistic, vivid images. In contrast, VAEs' generation is more blurred

GANs' generated images

Brock et al. LARGE SCALE GAN TRAINING FOR HIGH FIDELITY NATURAL IMAGE SYNTHESIS. ICLR 2019.

GANs v.s. VAEs

GANs are widely demonstrated to show superiority to VAEs on generating realistic, vivid images. In contrast, VAEs' generation is more blurred

GANs' generated images

GANs' generation can "miss mode" of the data distribution, where the generated images are not diverse to cover all the data distributions (VAEs do not have this issue)

Brock et al. LARGE SCALE GAN TRAINING FOR HIGH FIDELITY NATURAL IMAGE SYNTHESIS. ICLR 2019.

Implication of the KL divergence

Implication of the KL divergence

$$KL(p_{data}(x)||p_g(x)) \text{ v.s. } KL(p_g(x)||p_{data}(x))$$

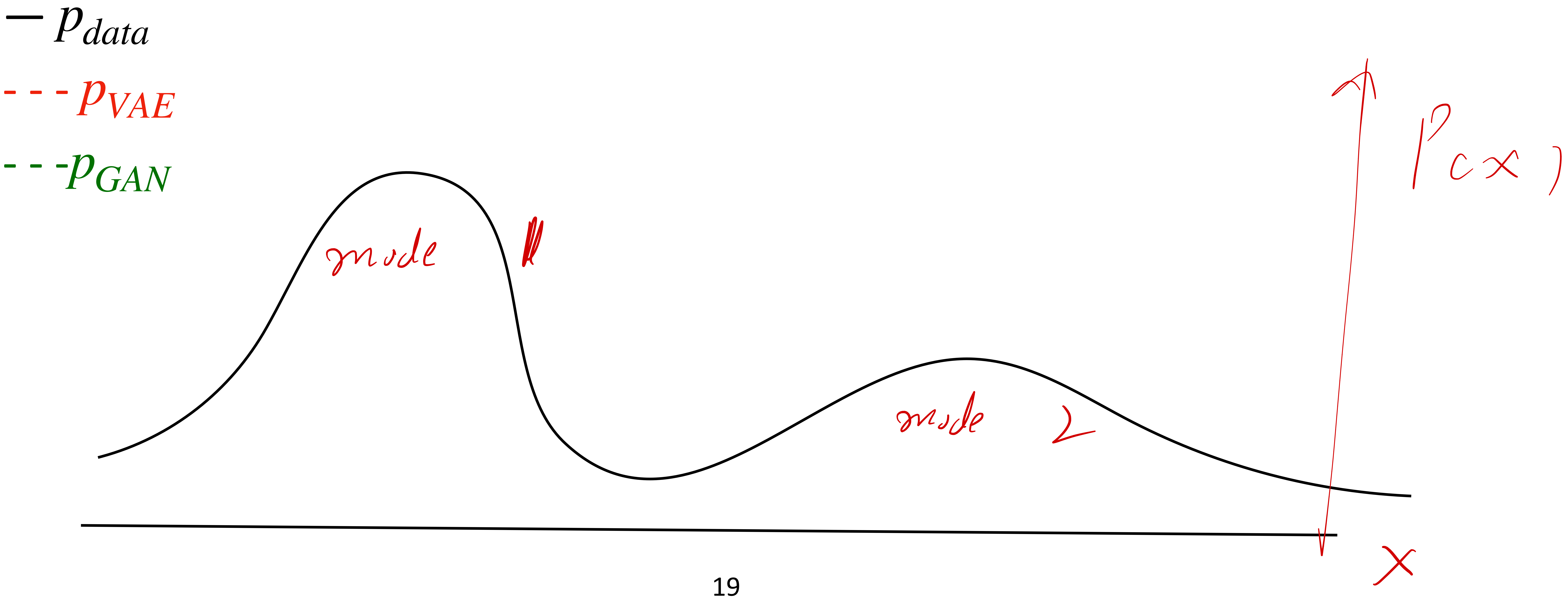
VAEs

GANs (approximately)

Implication of the KL divergence

$$KL(p_{data}(x)||p_g(x)) \text{ v.s. } KL(p_g(x)||p_{data}(x))$$

VAEs GANs (approximately)



Implication of the KL divergence

$$KL(p_{data}(x)||p_g(x)) \text{ v.s. } KL(p_g(x)||p_{data}(x))$$

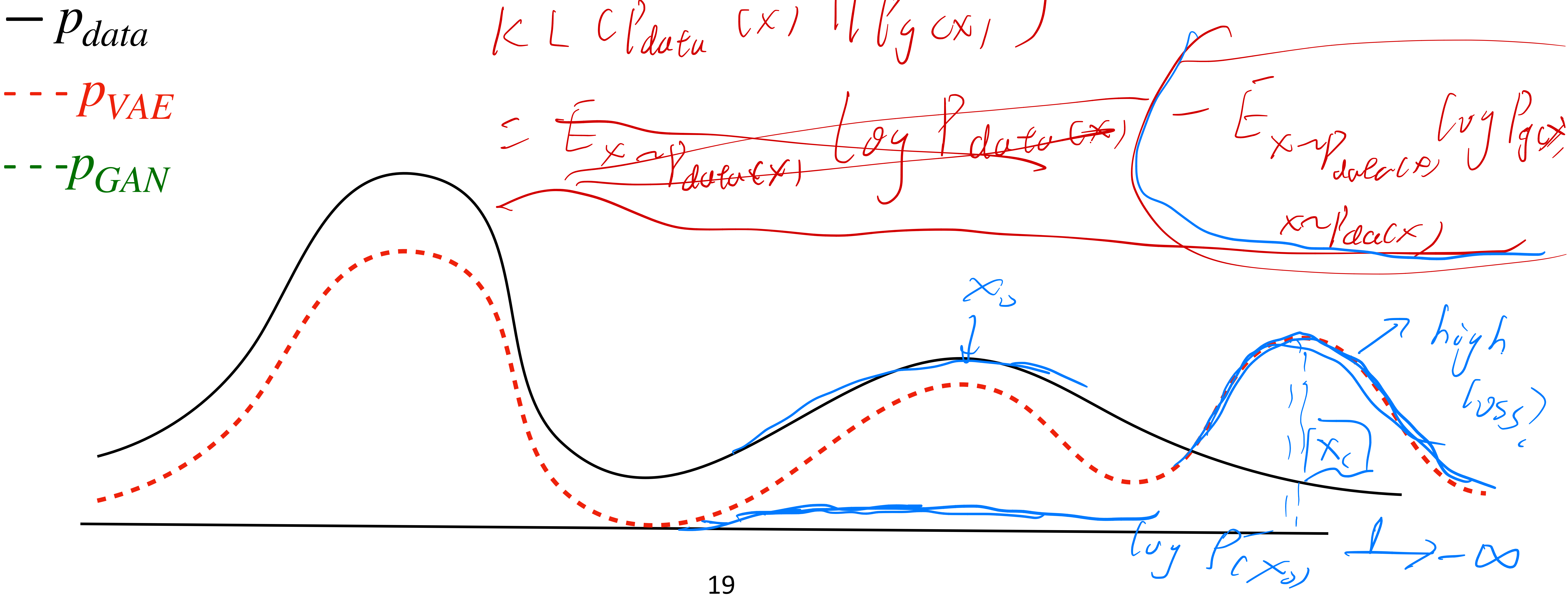
VAEs GANs (approximately)

$p_g(x)$

$KL(p_{data}(x) || p_g(x))$

$= \cancel{E_{x\sim p_{data}(x)} \log p_{data}(x)} - E_{x\sim p_{data}(x)} \log p_g(x)$

$x \sim p_{data}(x)$

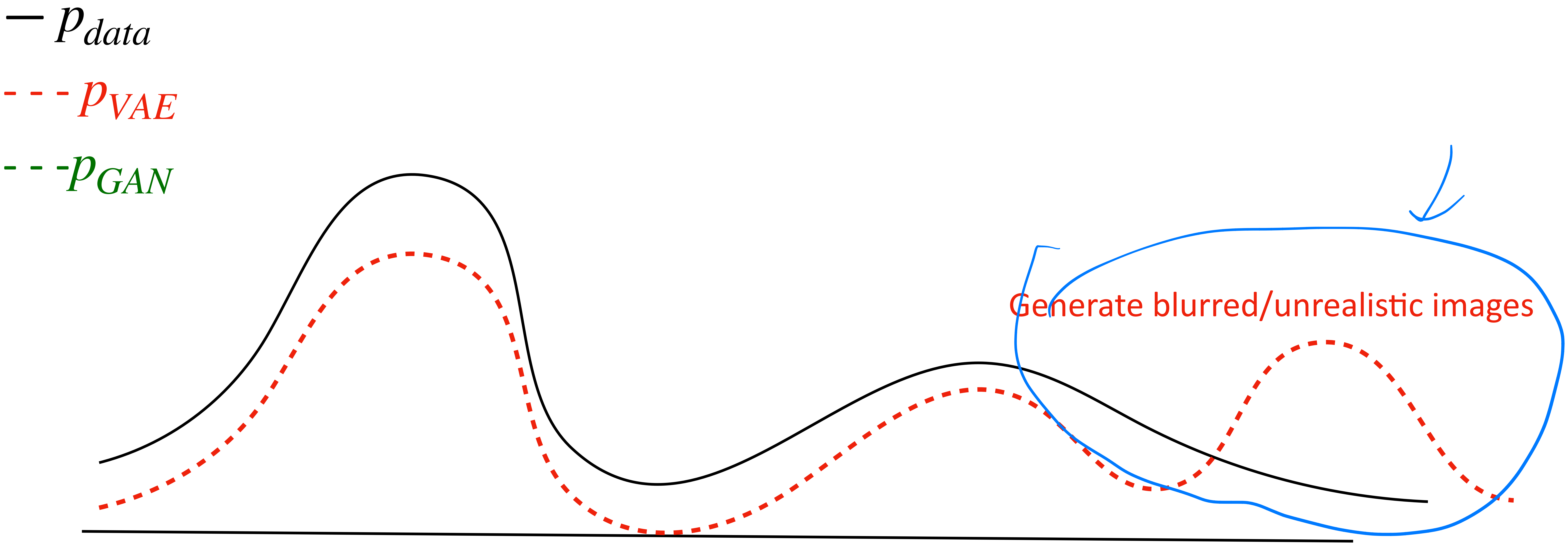


Implication of the KL divergence

$$KL(p_{data}(x)||p_g(x)) \text{ v.s. } KL(p_g(x)||p_{data}(x))$$

VAEs

GANs (approximately)



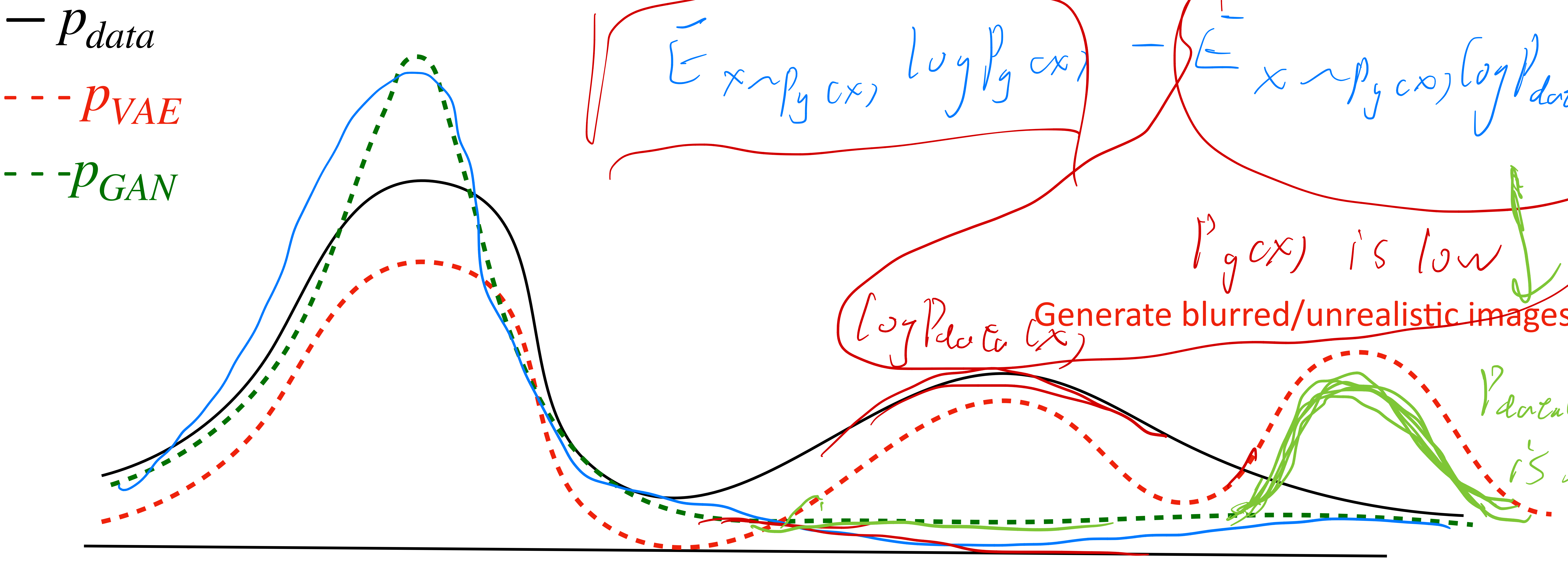
Implication of the KL divergence

$$KL(p_{data}(x)||p_g(x)) \text{ v.s. } KL(p_g(x)||p_{data}(x))$$

VAEs — ~~GANs (approximately)~~

$$E_{x\sim p_g(x)} \log p_g(x) - E_{x\sim p_g(x)} \log p_{data}(x)$$

$p_g(x)$ is low



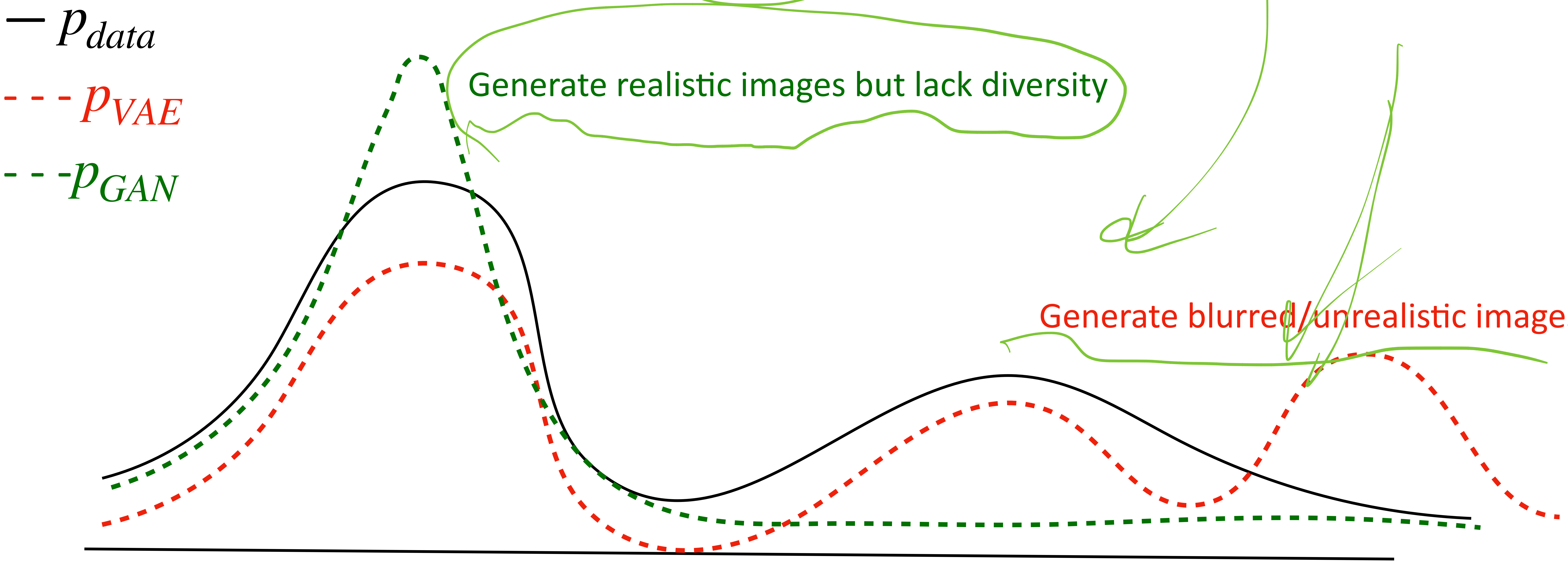
Implication of the KL divergence

$$KL(p_{data}(x)||p_g(x)) \text{ v.s. } KL(p_g(x)||p_{data}(x))$$

VAEs

GANs (approximately)

Generate realistic images but lack diversity



Reinforcement Learning

Learning Tasks

- Supervised learning - $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
  - Regression - $y^{(i)} \in \mathbb{R}$
  - Classification - $y^{(i)} \in \{1, \dots, C\}$

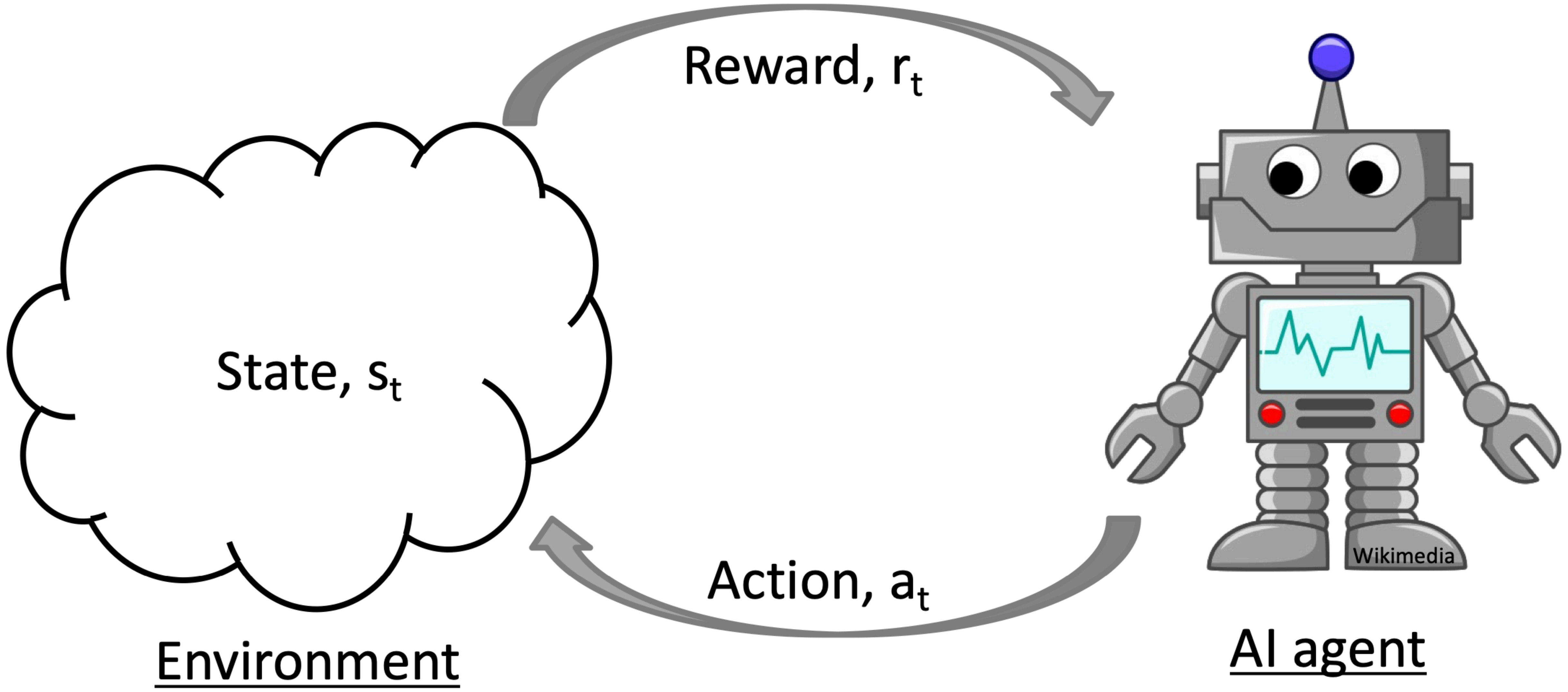
- Unsupervised learning - $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$
  - Clustering
  - Dimensionality reduction

Learning Tasks

- Supervised learning - $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$
  - Regression - $y^{(i)} \in \mathbb{R}$
  - Classification - $y^{(i)} \in \{1, \dots, C\}$
- Unsupervised learning - $\mathcal{D} = \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{N}$
  - Clustering
  - Dimensionality reduction
- Reinforcement learning - $\mathcal{D} = \left\{\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}, \boldsymbol{r}^{(t)}\right\}_{t=1}^{T}$

RL Setup

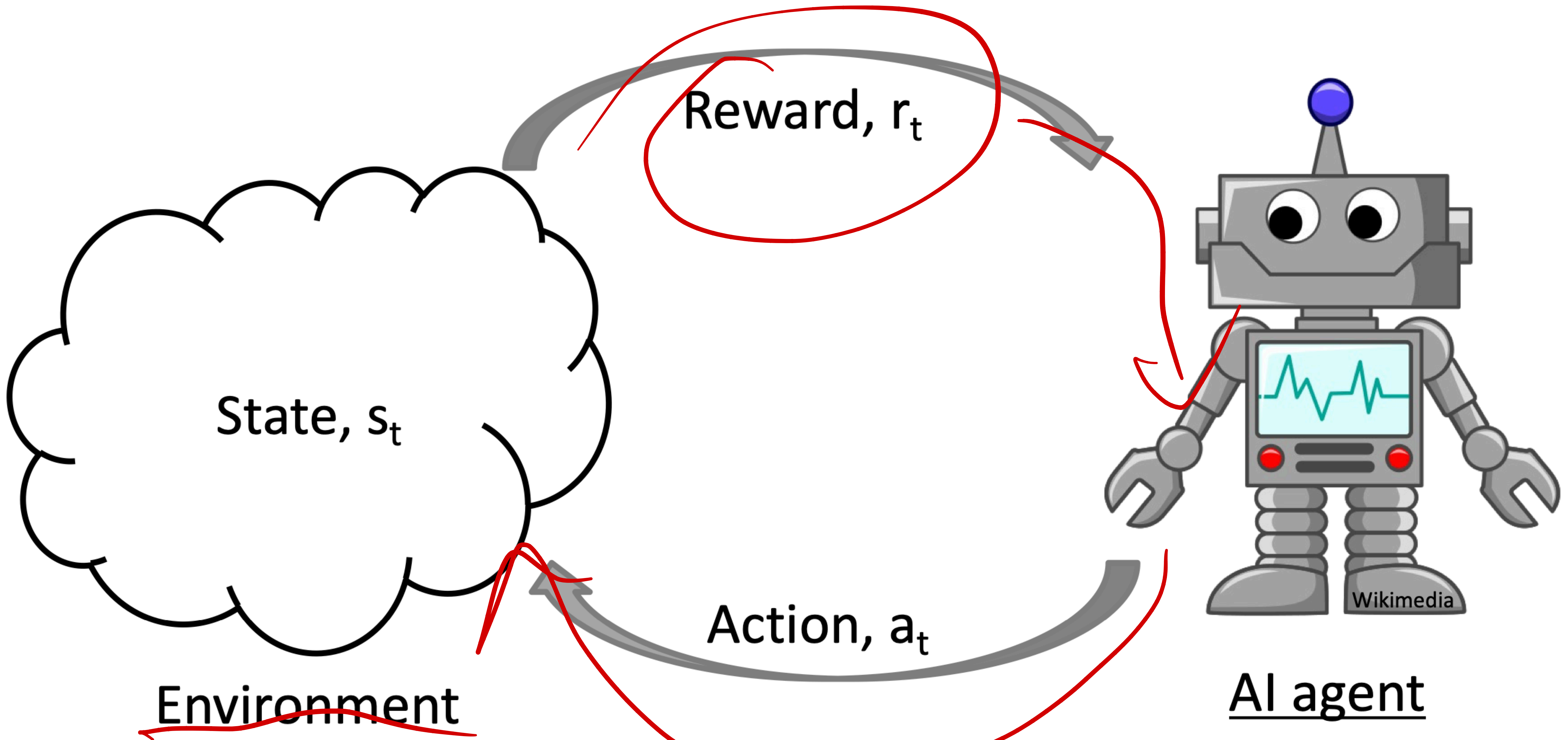
In many cases, we cannot precisely define what the correct output is (think of we want to train a robot to walk)



supervised ⟺ imitation learning

RL Setup

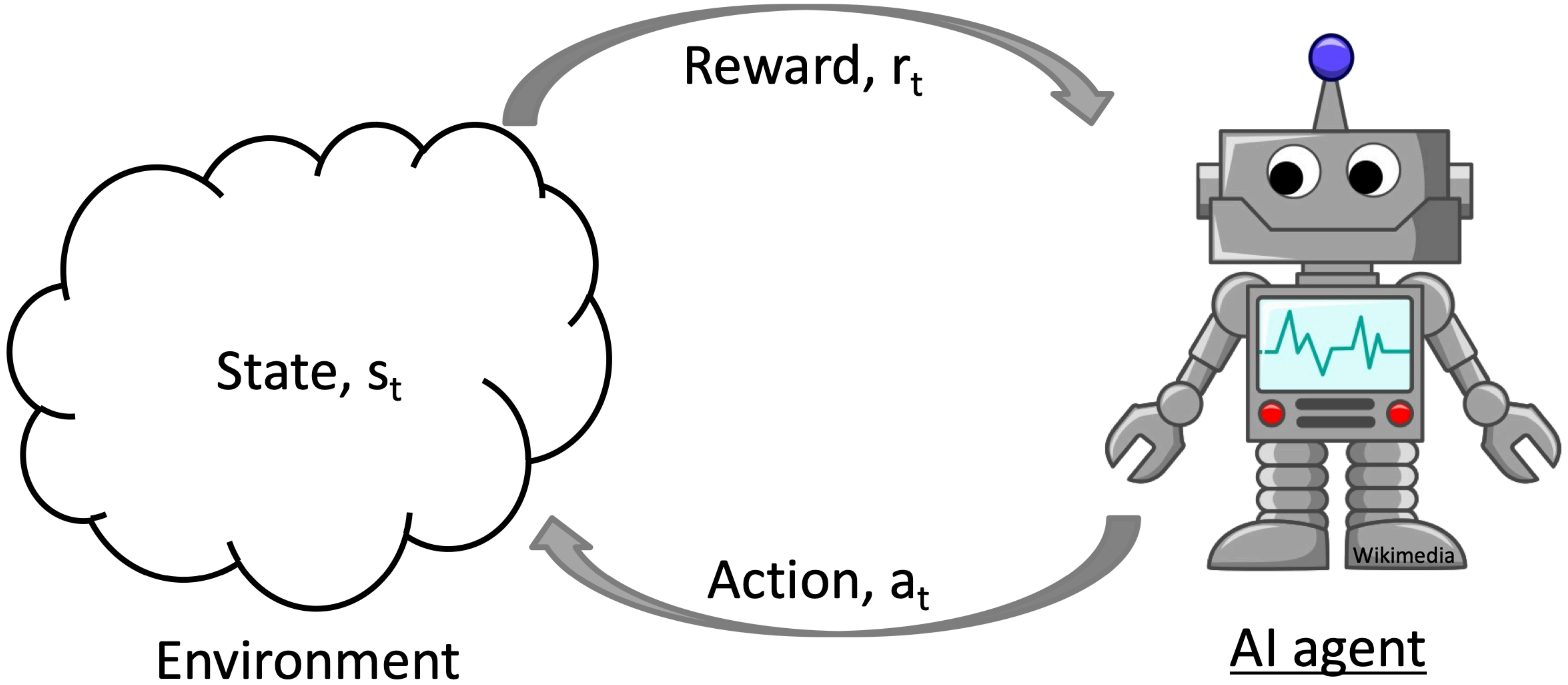
In many cases, we cannot precisely define what the correct output is (think of we want to train a robot to walk)



Agent chooses **actions** which can depend on past

RL Setup

In many cases, we cannot precisely define what the correct output is (think of we want to train a robot to walk)

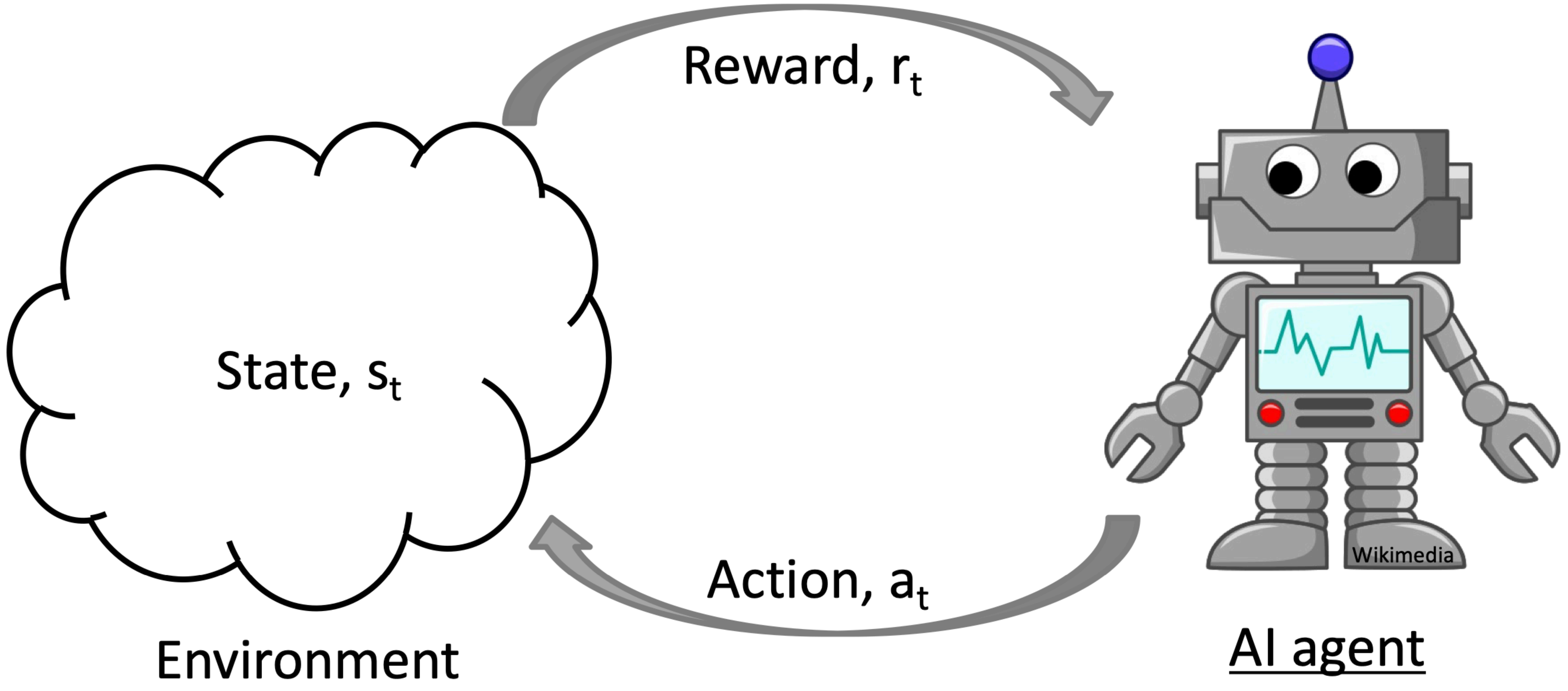


Agent chooses **actions** which can depend on past

Environment can change **state** with each action

RL Setup

In many cases, we cannot precisely define what the correct output is (think of we want to train a robot to walk)



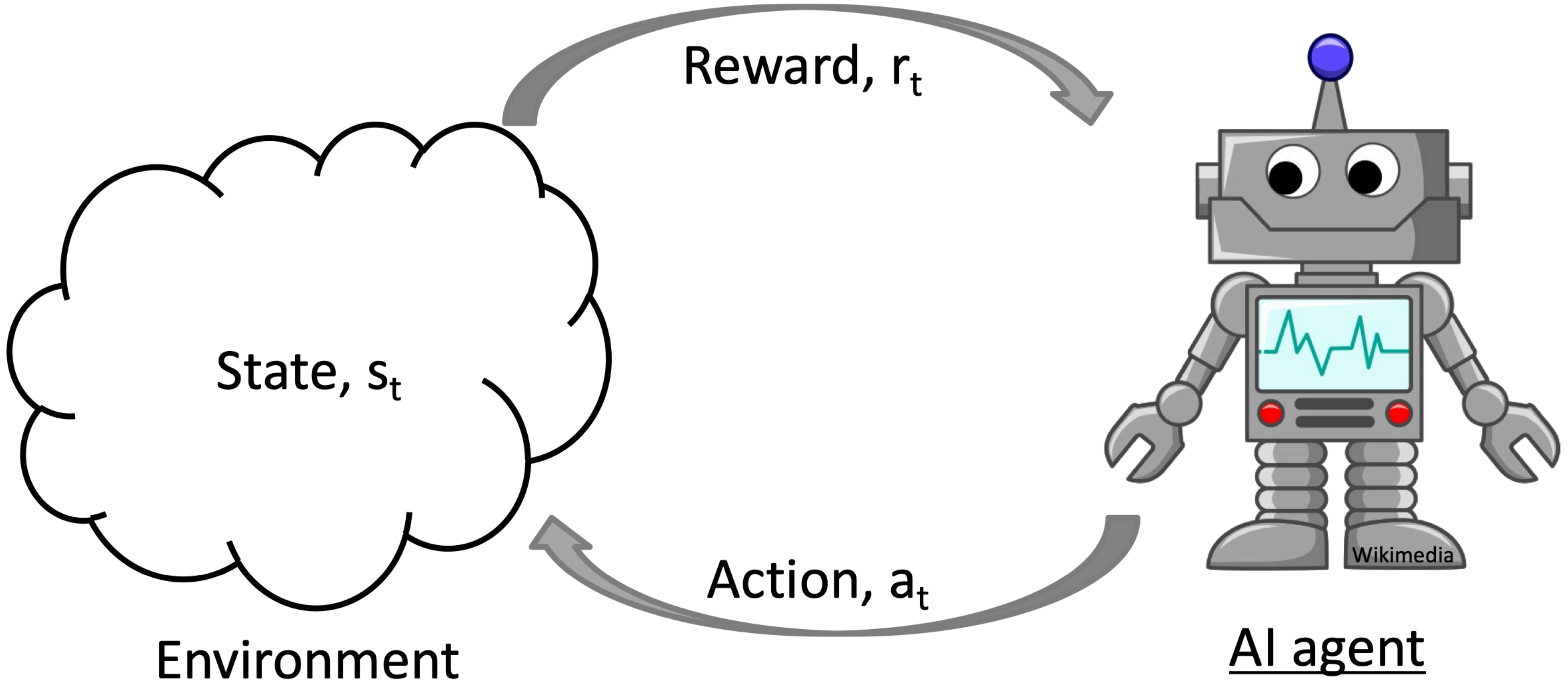
Agent chooses **actions** which can depend on past

Environment can change **state** with each action

Reward (Output) depends on (Inputs) action and state of environment

RL Setup

In many cases, we cannot precisely define what the correct output is (think of we want to train a robot to walk)



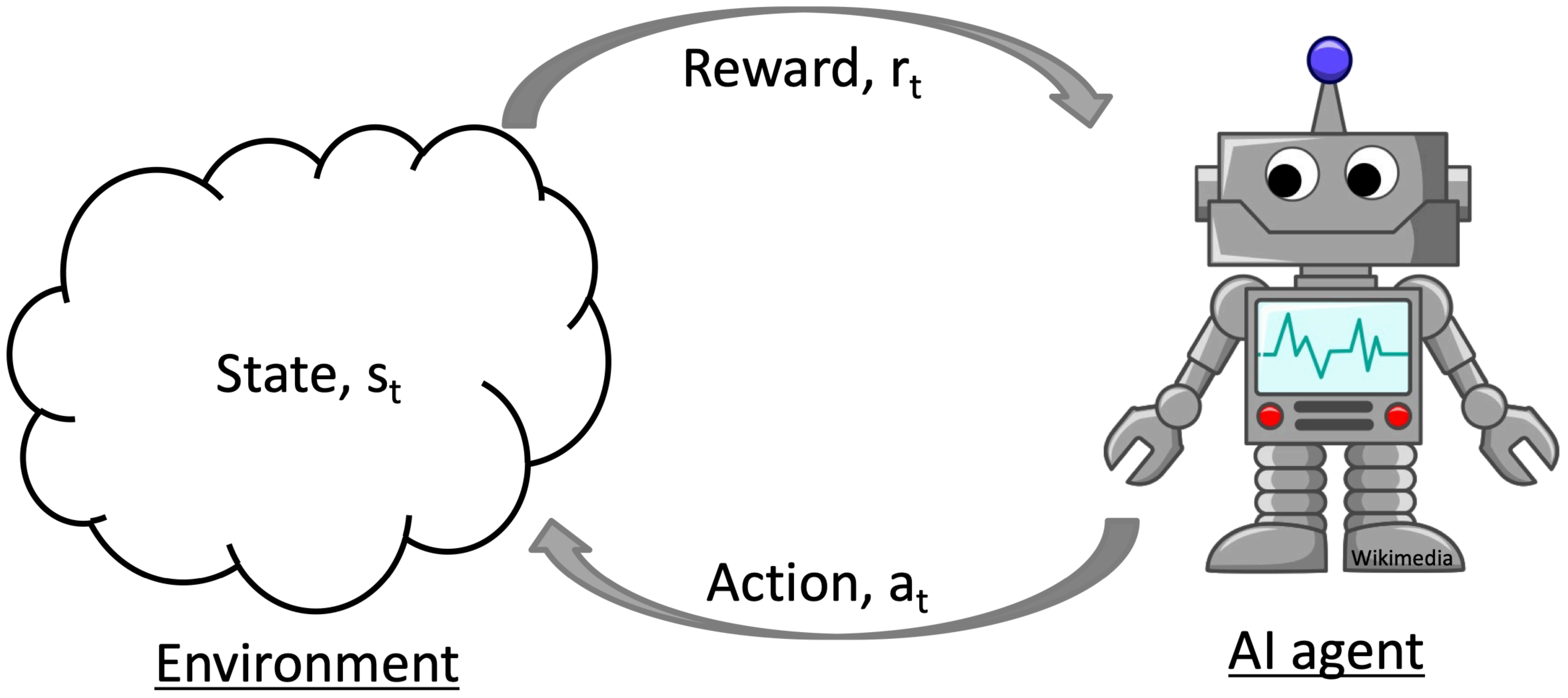
Agent chooses **actions** which can depend on past

Environment can change **state** with each action

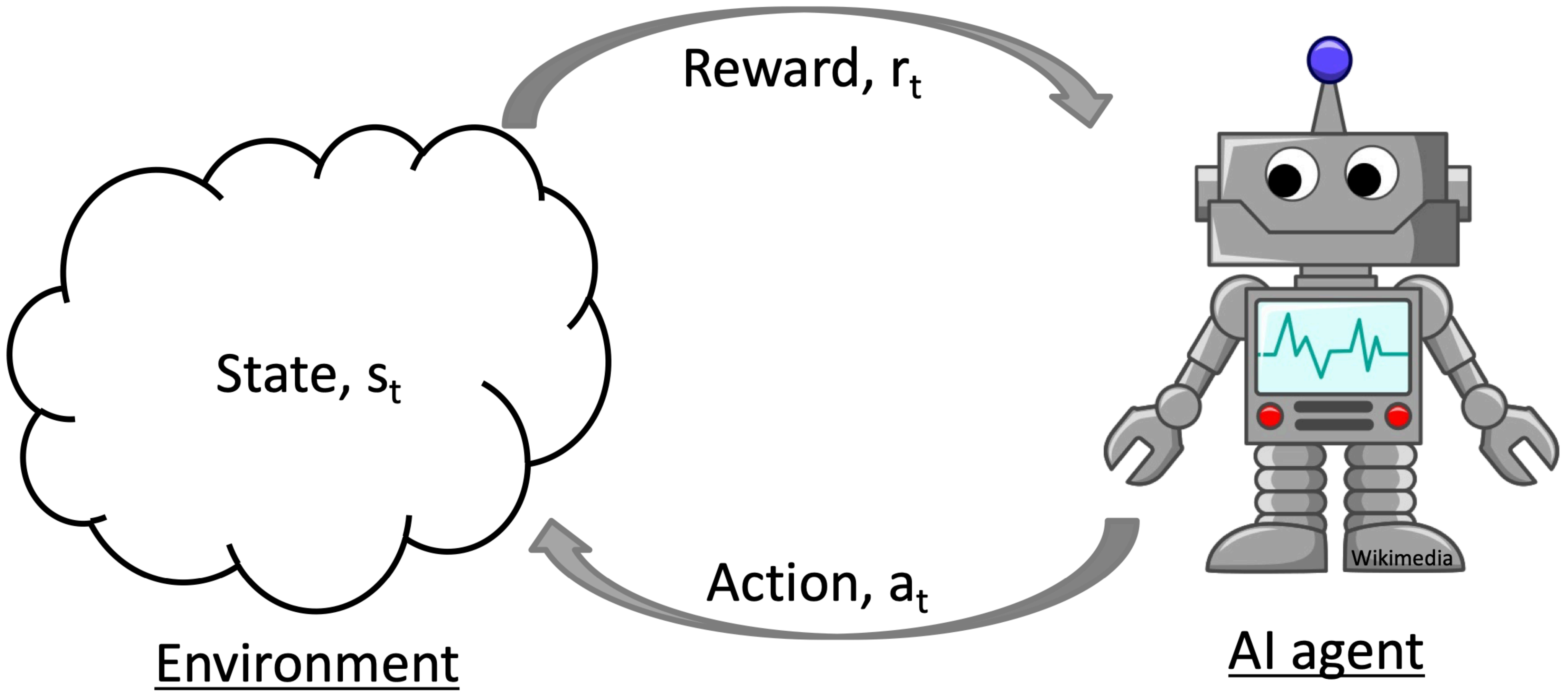
Reward (Output) depends on (Inputs) action and state of environment

Goal: maximize the total reward

Differences from Supervised Learning

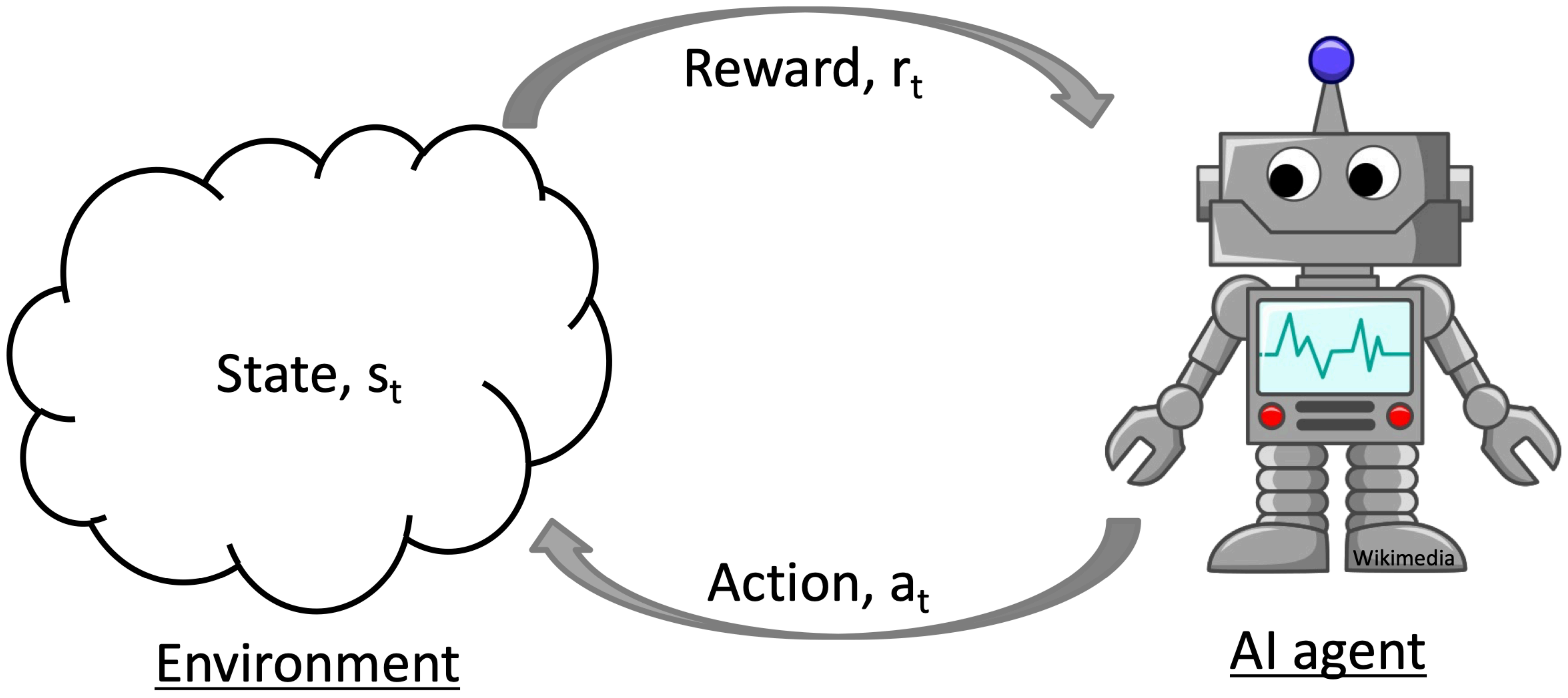


Differences from Supervised Learning



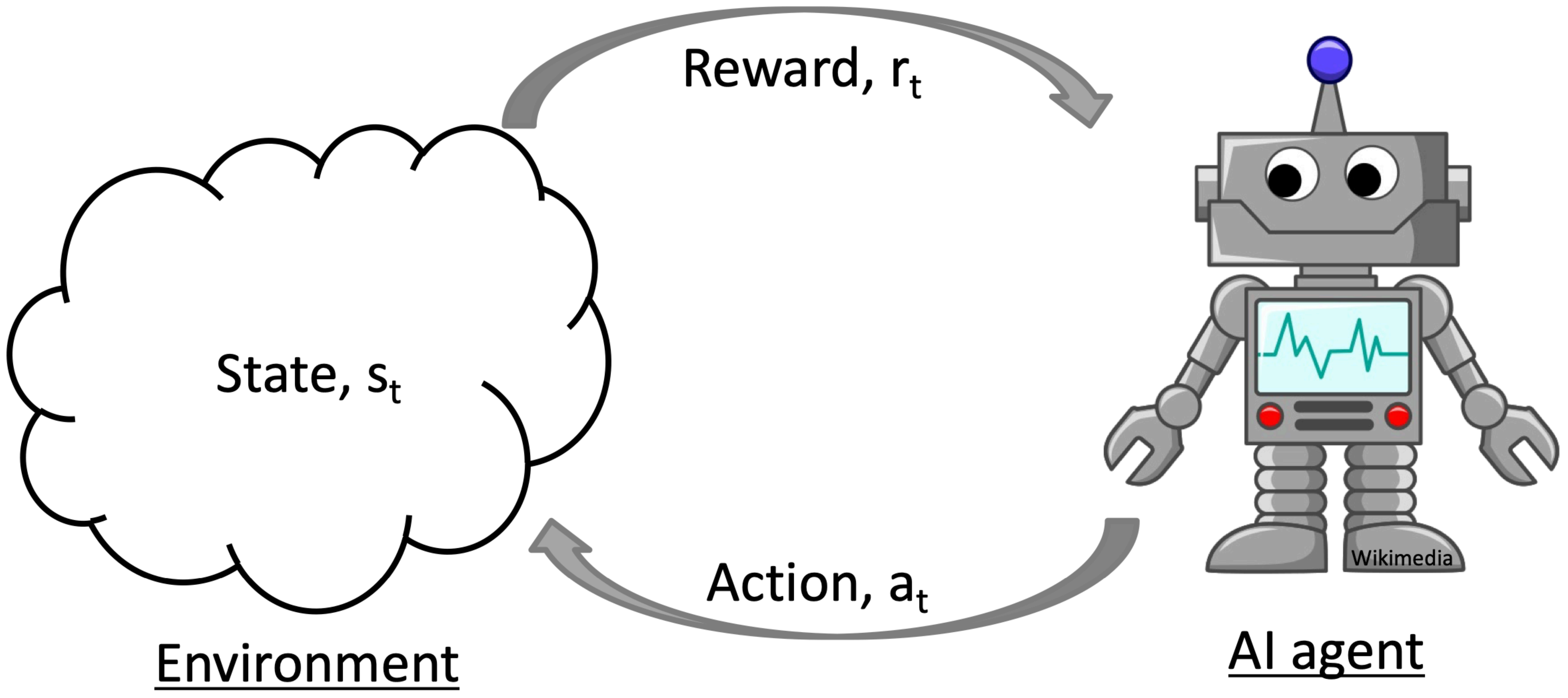
- Maximize reward (rather than learn reward)

Differences from Supervised Learning



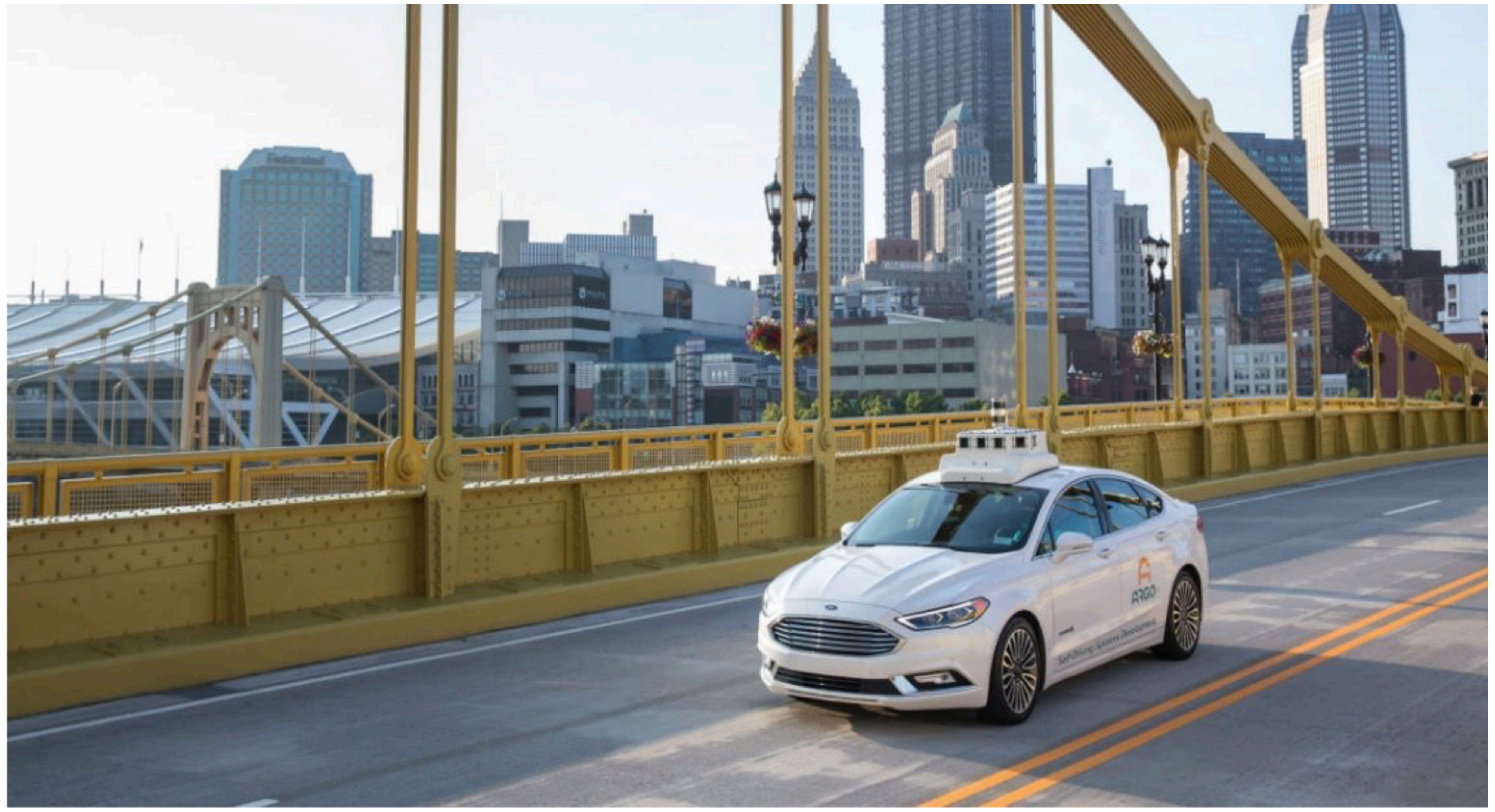
- Maximize reward (rather than learn reward) Supervised training is like imitation

Differences from Supervised Learning



- Maximize reward (rather than learn reward) Supervised training is like imitation
- Inputs are not iid – state & action depends on past

RL Examples



RL Setup

RL Setup

- State space, $\mathcal{S}$
- Action space, $\mathcal{A}$

RL Setup

- State space, $\mathcal{S}$
- Action space, $\mathcal{A}$
- Reward function
  - Stochastic, $p(r \mid s, a)$
  - Deterministic, $R\colon \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

RL Setup

- State space, $\mathcal{S}$
- Action space, $\mathcal{A}$
- Reward function
  - Stochastic, $p(r \mid s, a)$
  - Deterministic, $R \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- Transition function
  - Stochastic, $p(s' \mid s, a)$
  - Deterministic, $\delta \colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}$

RL Setup

- State space, $\mathcal{S}$
- Action space, $\mathcal{A}$
- Reward function
  - Stochastic, $p(r \mid s, a)$
  - Deterministic, $R\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- Transition function
  - Stochastic, $p(s' \mid s, a)$
  - Deterministic, $\delta\colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}$
- Reward and transition functions can be known or unknown

RL Setup

- State space, $\mathcal{S}$
- Action space, $\mathcal{A}$
- Reward function
  - Stochastic, $p(r \mid s, a)$
  - Deterministic, $R: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- Transition function
  - Stochastic, $p(s' \mid s, a)$
  - Deterministic, $\delta: \mathcal{S} \times \mathcal{A} \to \mathcal{S}$

In this lecture, we assume they are known

- Reward and transition functions can be known or unknown

RL Setup

- Policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$
  - Specifies an action to take in *every* state

RL Setup

- Policy, $\pi : \mathcal{S} \to \mathcal{A}$
  - Specifies an action to take in *every* state

- Value function, $V^{\pi}: \mathcal{S} \to \mathbb{R}$
  - Measures the expected total reward of starting in some state s and executing policy π, i.e., in every state, taking the action that π returns

RL Example - gridworld

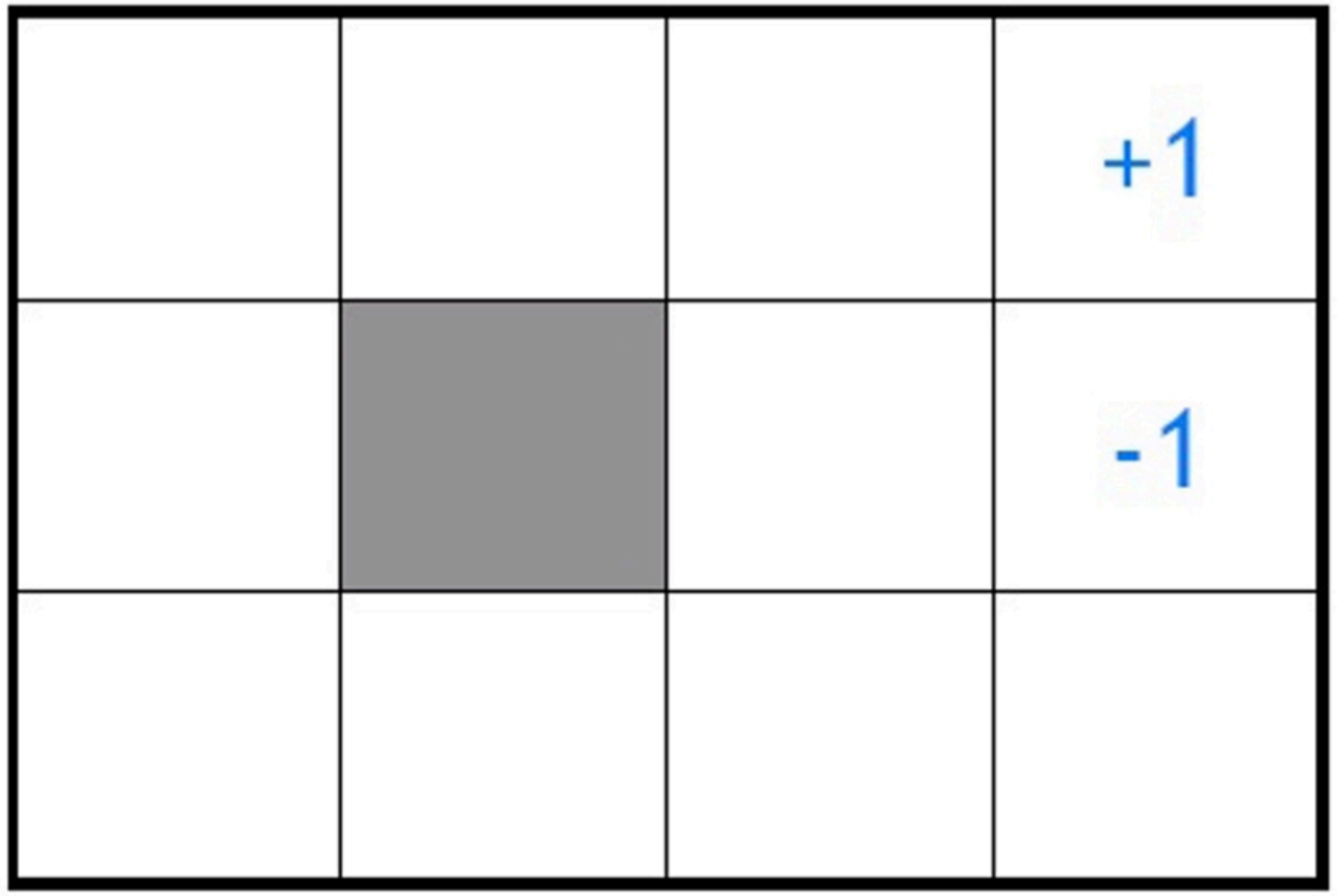
$\mathcal{S}$ = all empty squares in the grid

$\mathcal{A}$ = {up, down, left, right}

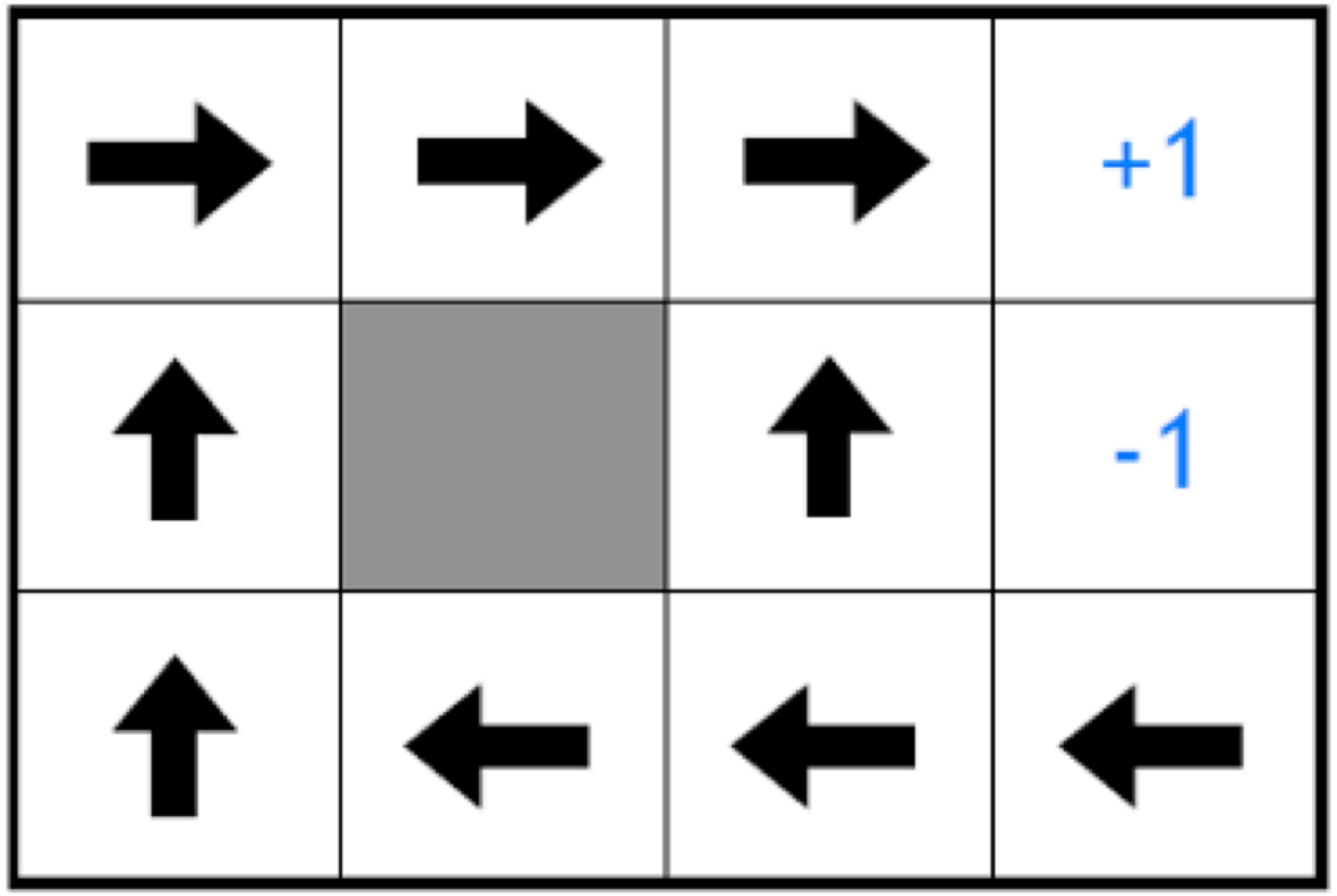
Deterministic transitions

Rewards of +1 and -1 for entering the labelled squares

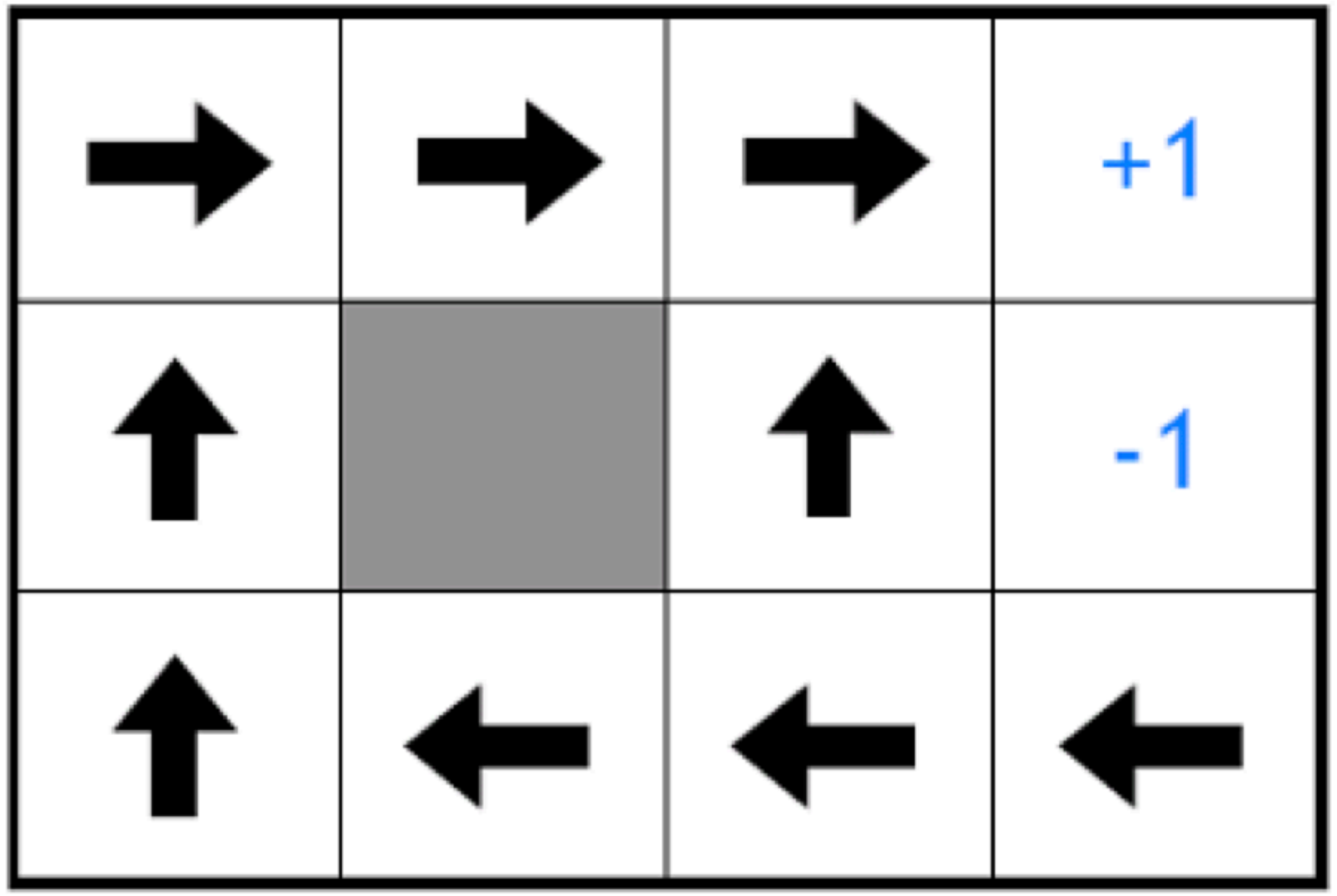
Terminate after receiving either reward



RL Example - gridworld



RL Example - gridworld



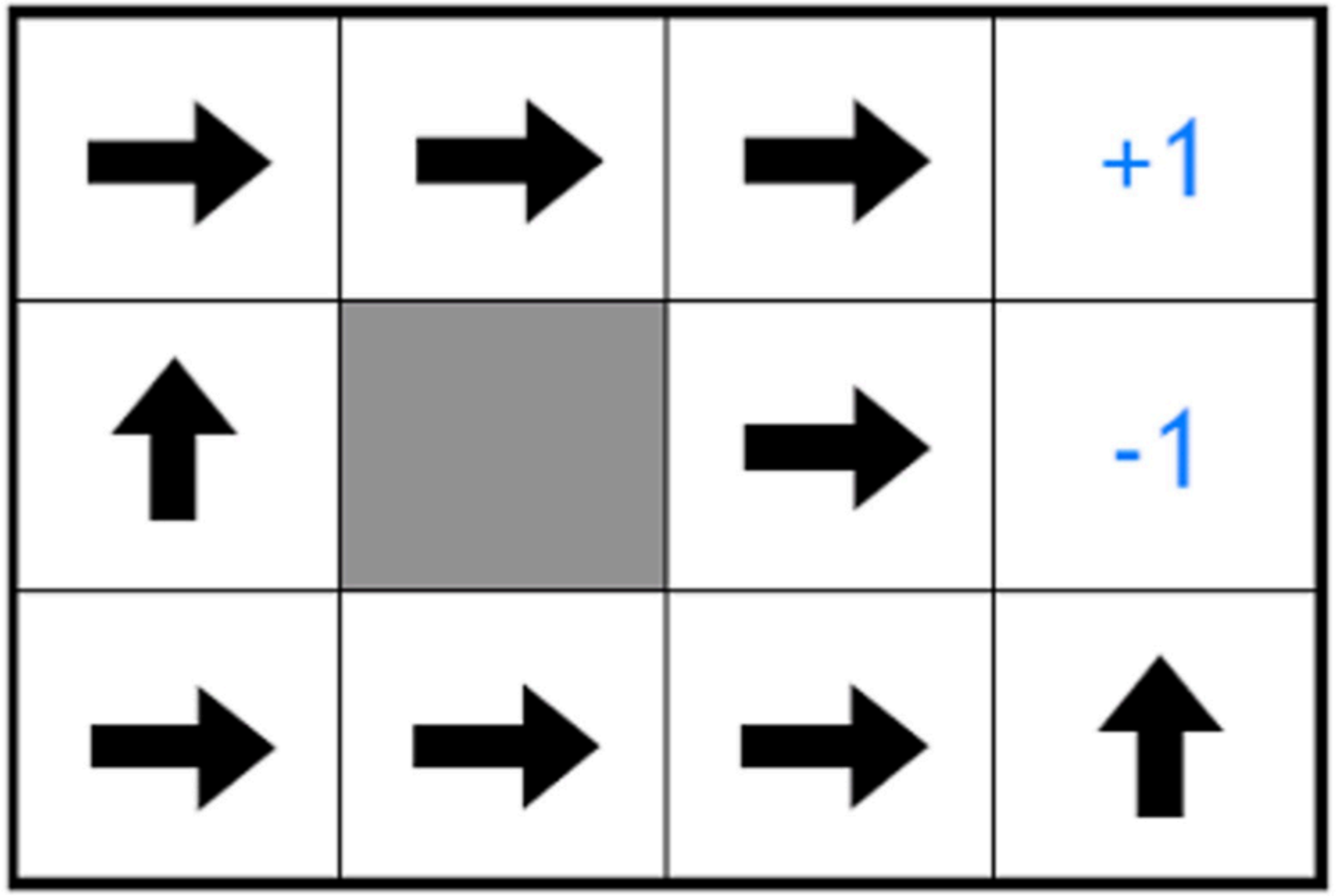
Is this policy optimal?

RL Example - gridworld

Optimal policy given a reward of -2 per step

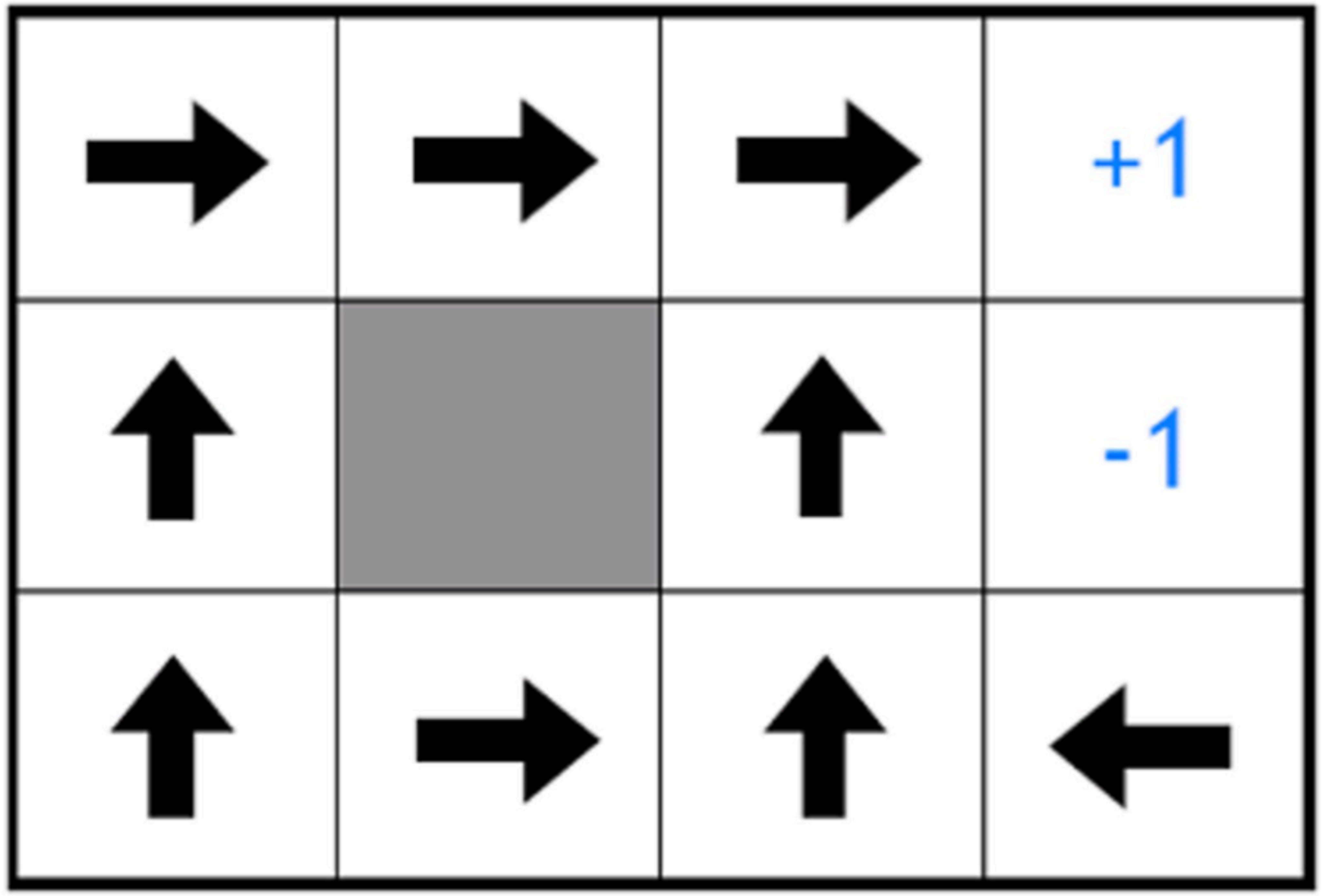
RL Example - gridworld

Optimal policy given a reward of -2 per step



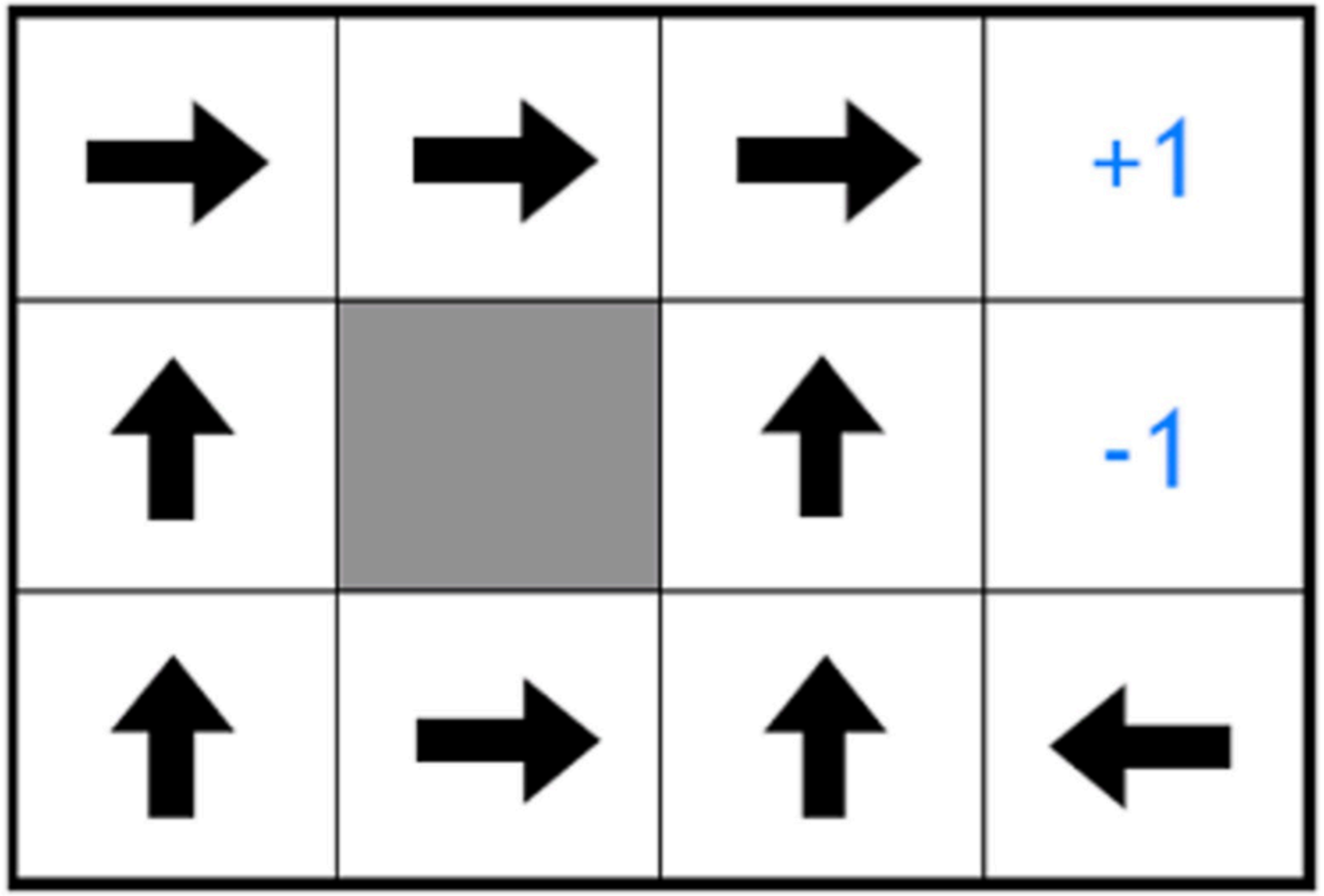
RL Example - gridworld

Optimal policy given a reward of -0.5 per step



RL Example - gridworld

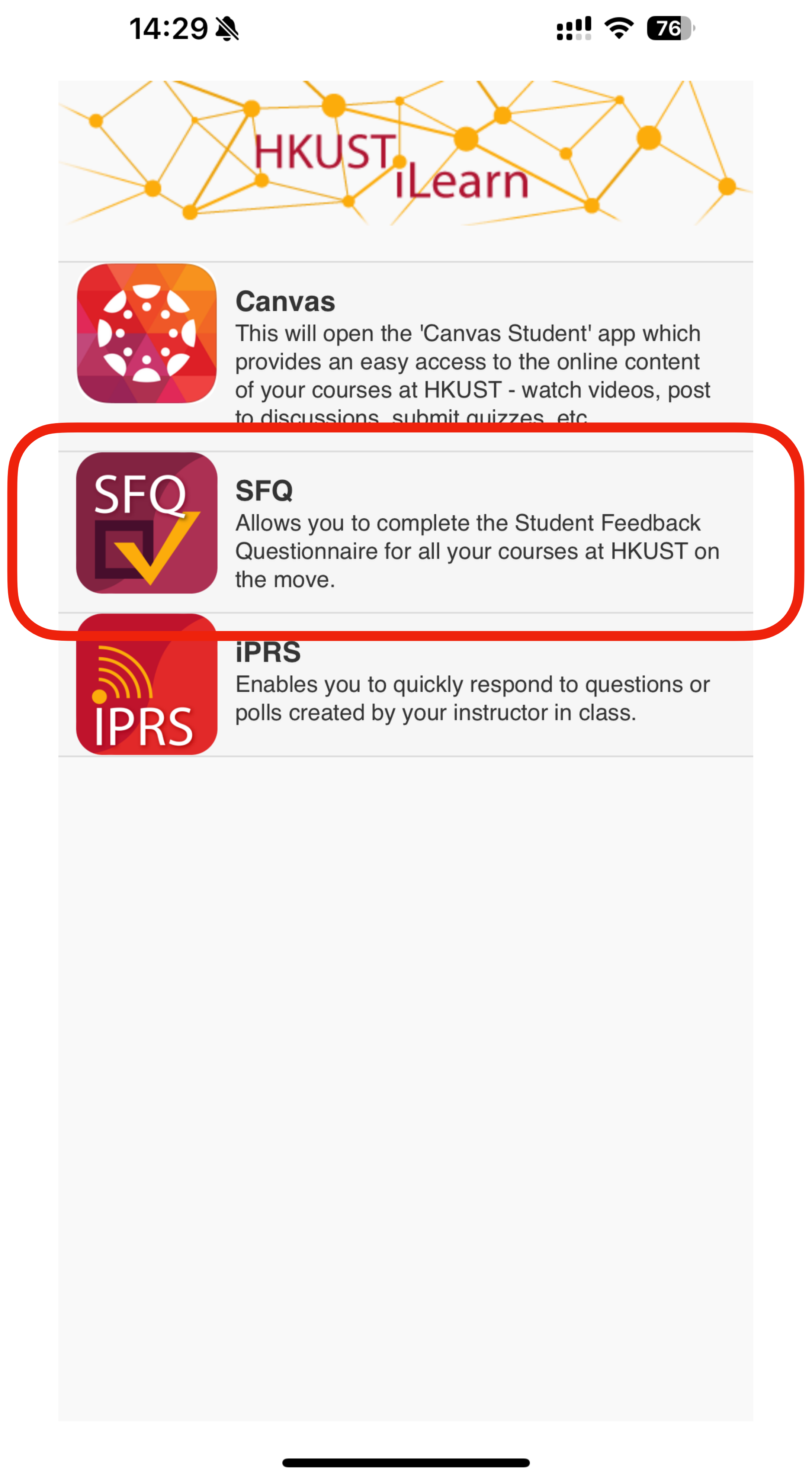
Optimal policy given a reward of -0.5 per step



What would be the algorithm to find the optimal policy automatically?

Course Evaluation

Anonymous to instructors



Or

https://survey.ust.hk/hkust/