Intro IA - Intelligent Systems

Deep Learning
AE, VAE, GAN, LSTM

Sergi Pujades

Autoencoders

What is an auto-encoder ?

Type of **artificial neural network** used to learn (efficient) data **encodings** of **unlabeled** data (in an unsupervised manner).

The goal is to learn a **representation**, typically dimensionality reduction, by training the network to ignore insignificant data.

Known techniques of dimensionality reduction?

https://en.wikipedia.org/wiki/Autoencoder
https://www.v7labs.com/blog/autoencoders-guide#autoencoders-intro

Dimensionality Reduction

Principal Component Analysis

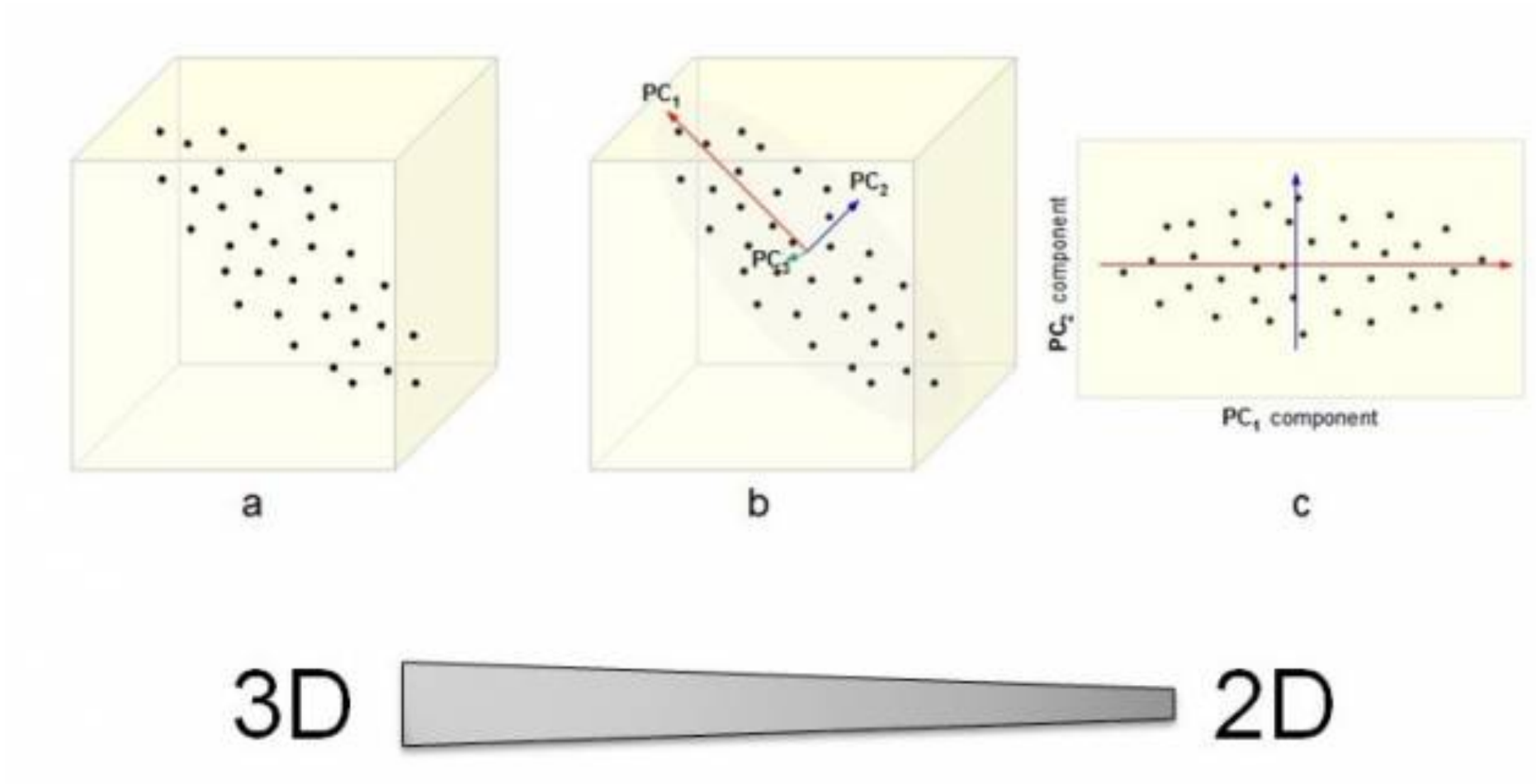


Image from https://www.davidzeleny.net/anadat-r/doku.php/en:pca

Dimensionality Reduction

Principal Component Analysis - main limitation

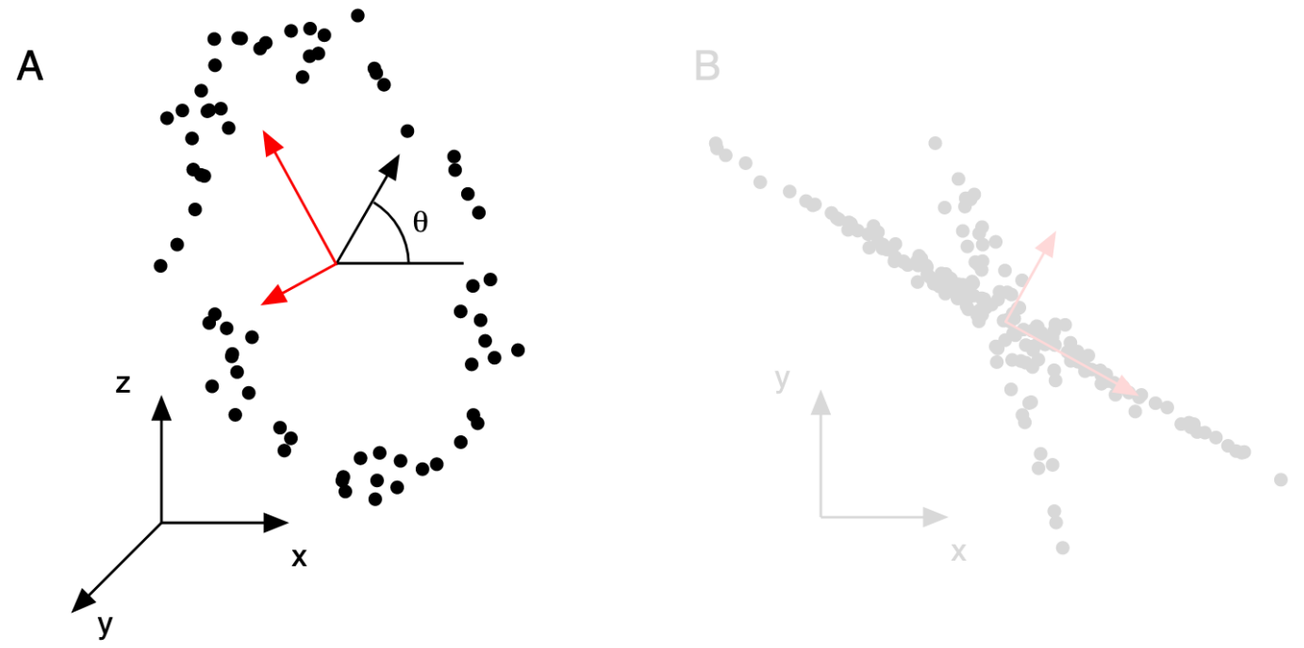
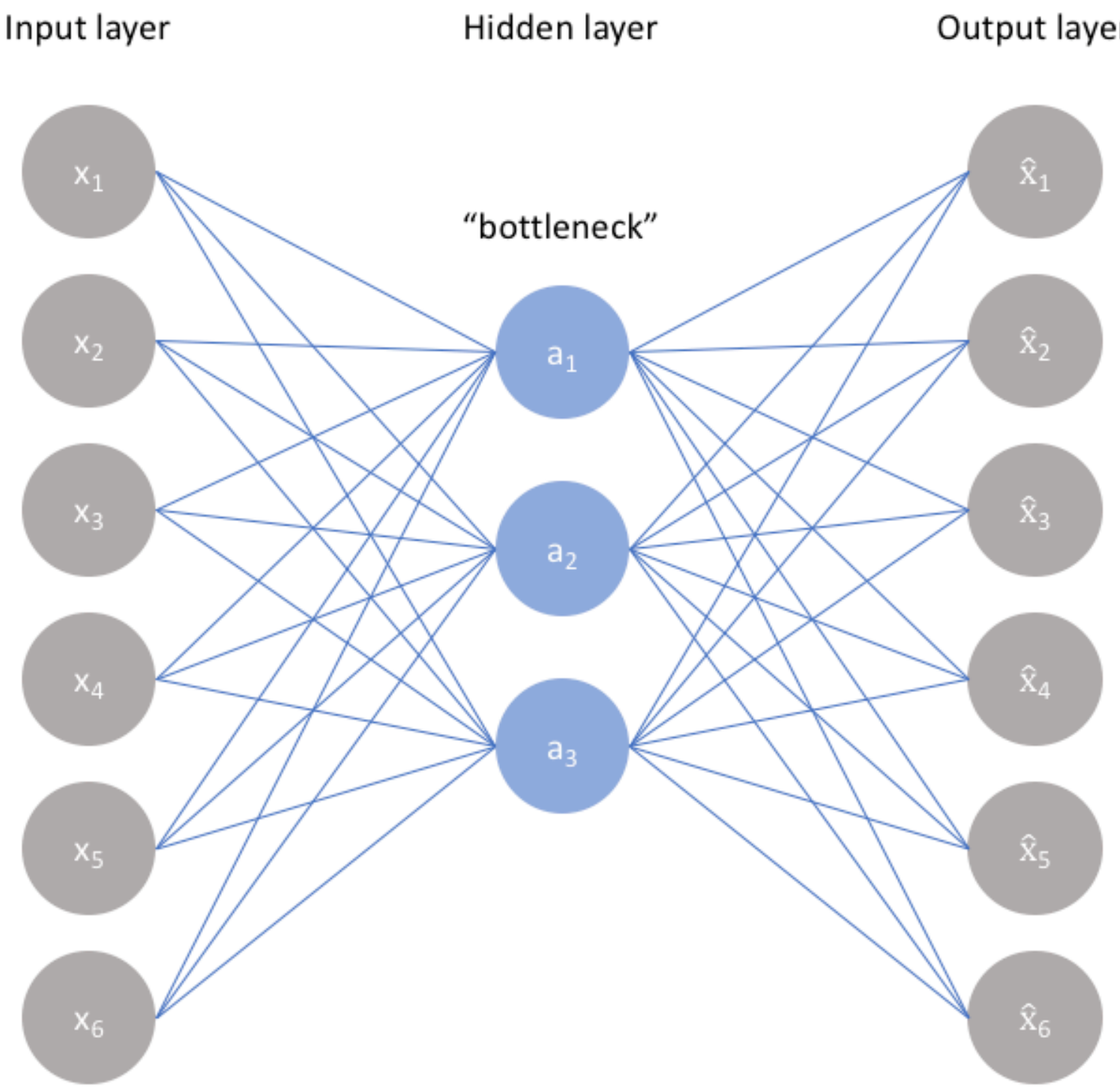


FIG. 6 Example of when PCA fails (red lines). (a) Tracking a person on a ferris wheel (black dots). All dynamics can be described by the phase of the wheel θ. a non-linear combination of the naive basis. (b) In this example data set, non-Gaussian distributed data and non-orthogonal axes causes PCA to fail. The axes with the largest variance do not correspond to the appropriate answer.

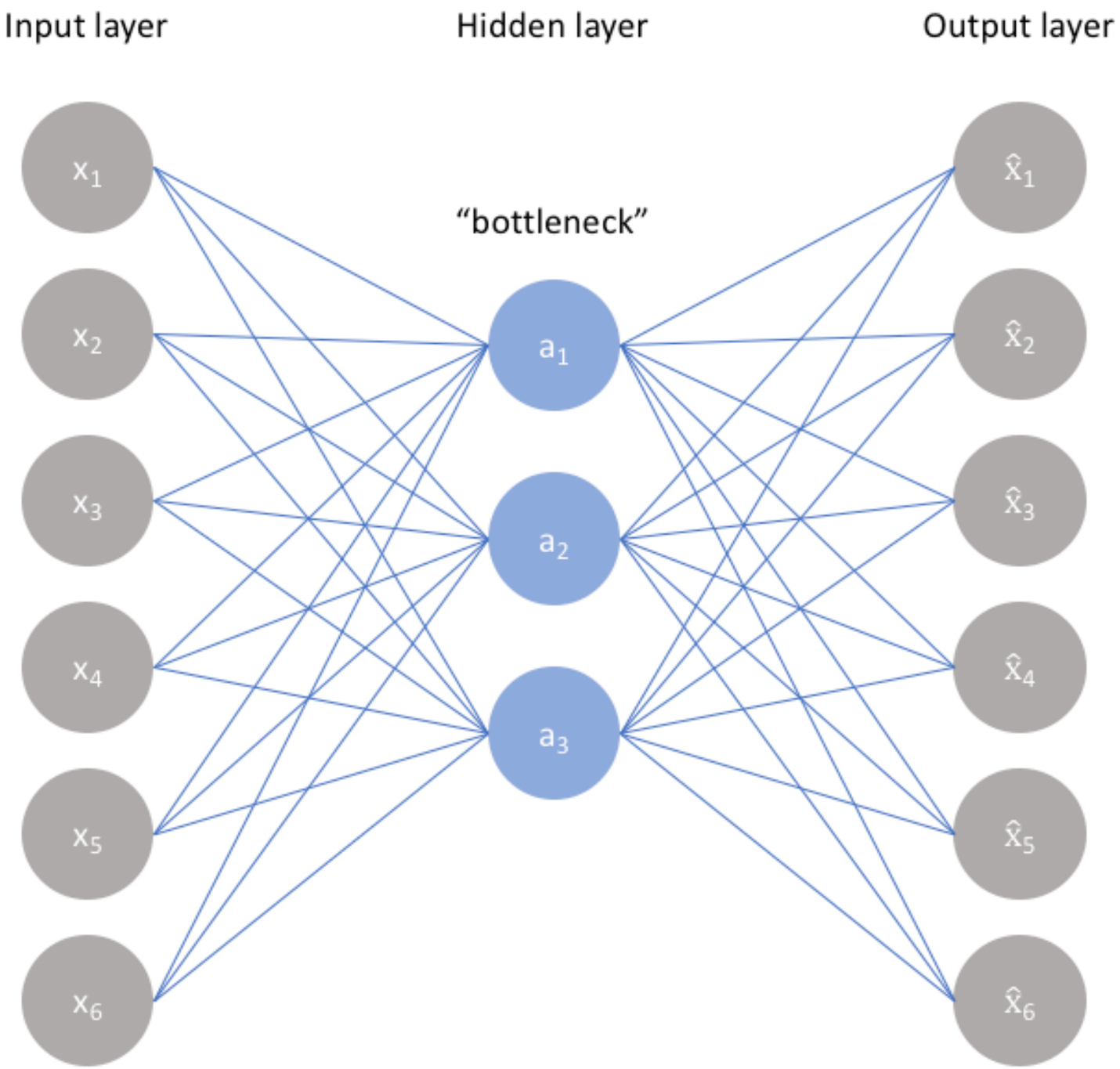
Figure from Shlens, Jonathon. "A tutorial on principal component analysis." arXiv preprint arXiv:1404.1100 (2014).

Autoencoder: basic architecture



https://www.v7labs.com/blog/autoencoders-guide#autoencoders-intro

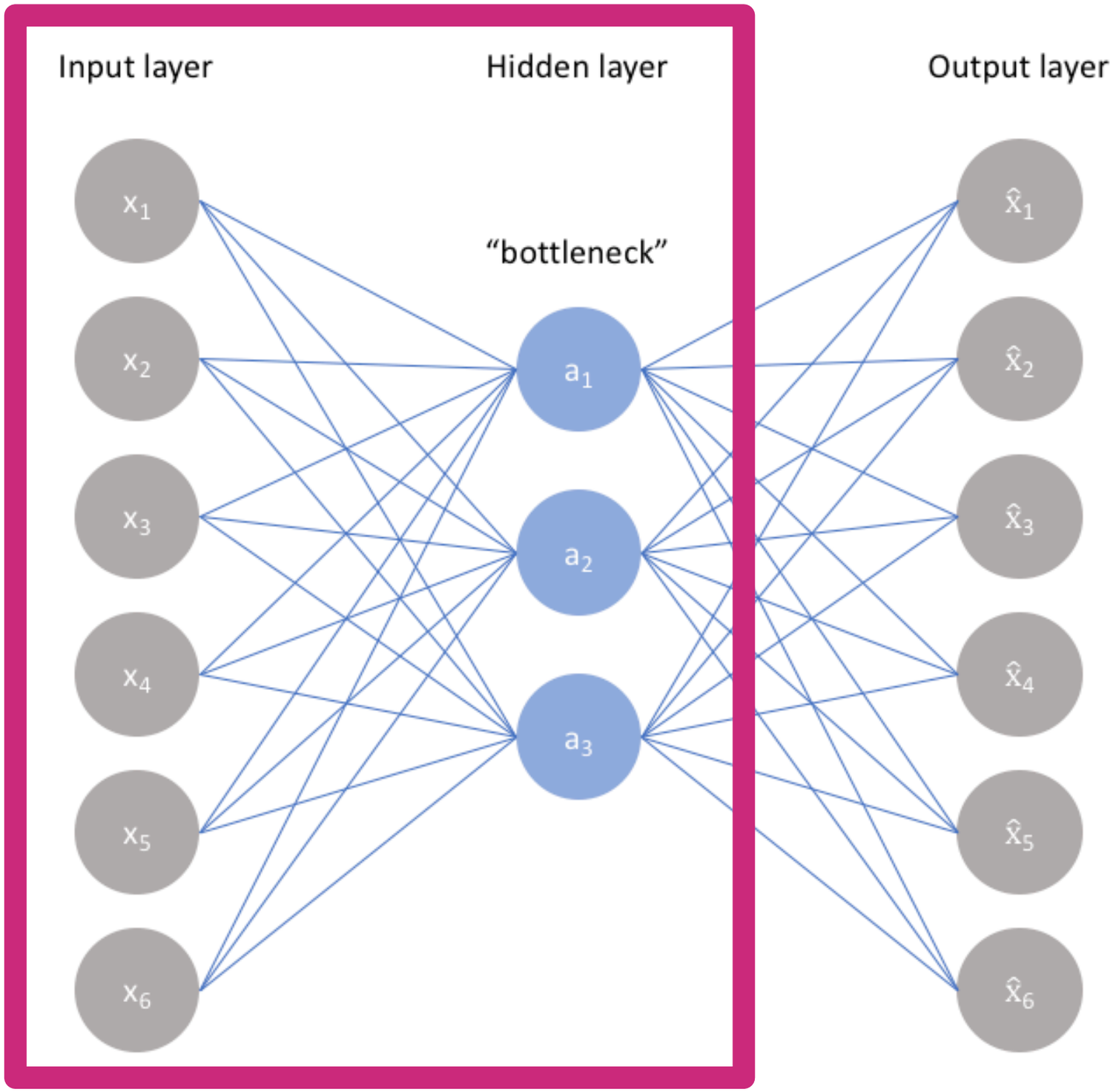
Autoencoder: basic architecture



3 main parts:

https://www.v7labs.com/blog/autoencoders-guide#autoencoders-intro

Autoencoder: basic architecture

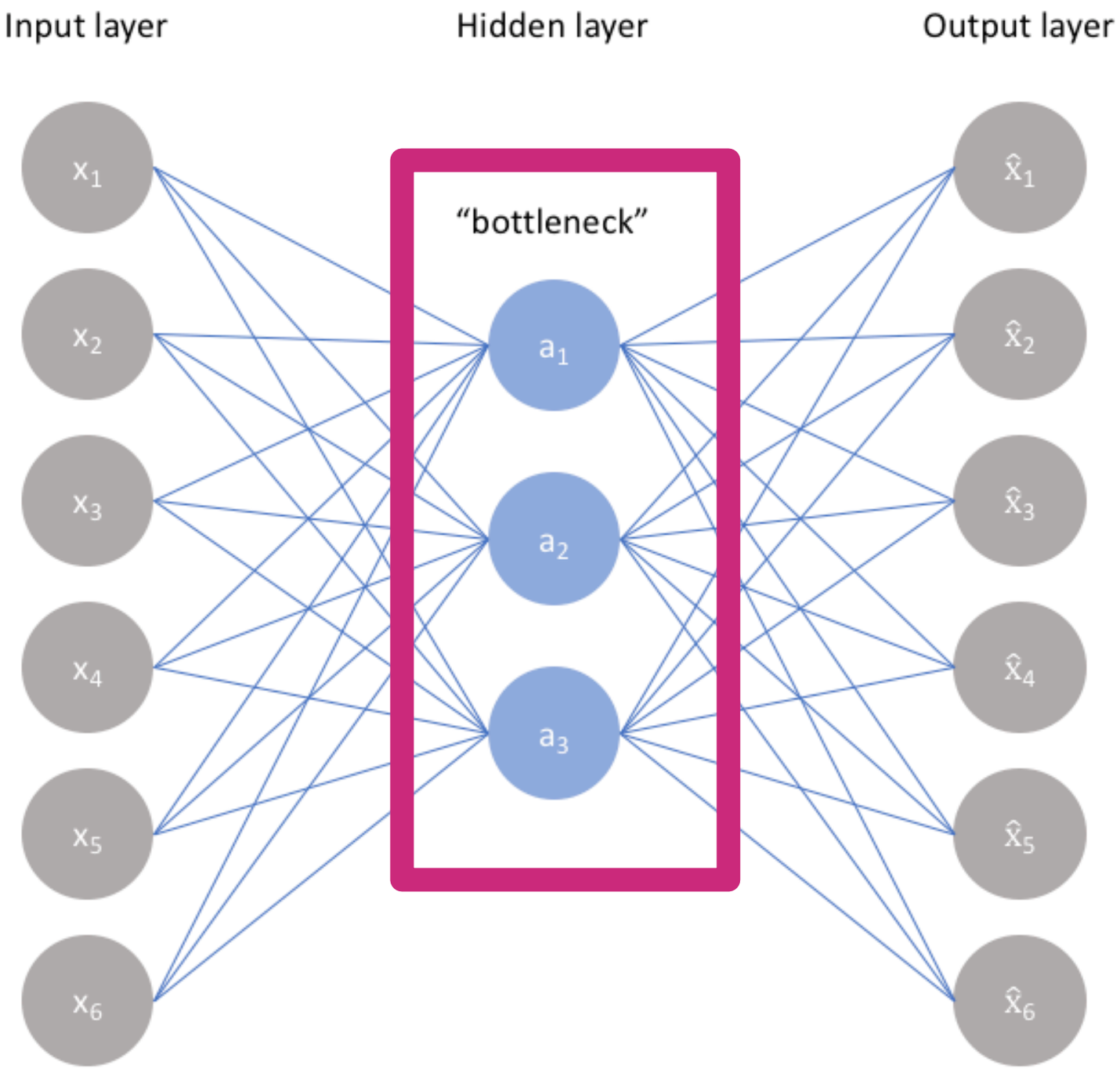


3 main parts:

Encoder

https://www.v7labs.com/blog/autoencoders-guide#autoencoders-intro

Autoencoder: basic architecture



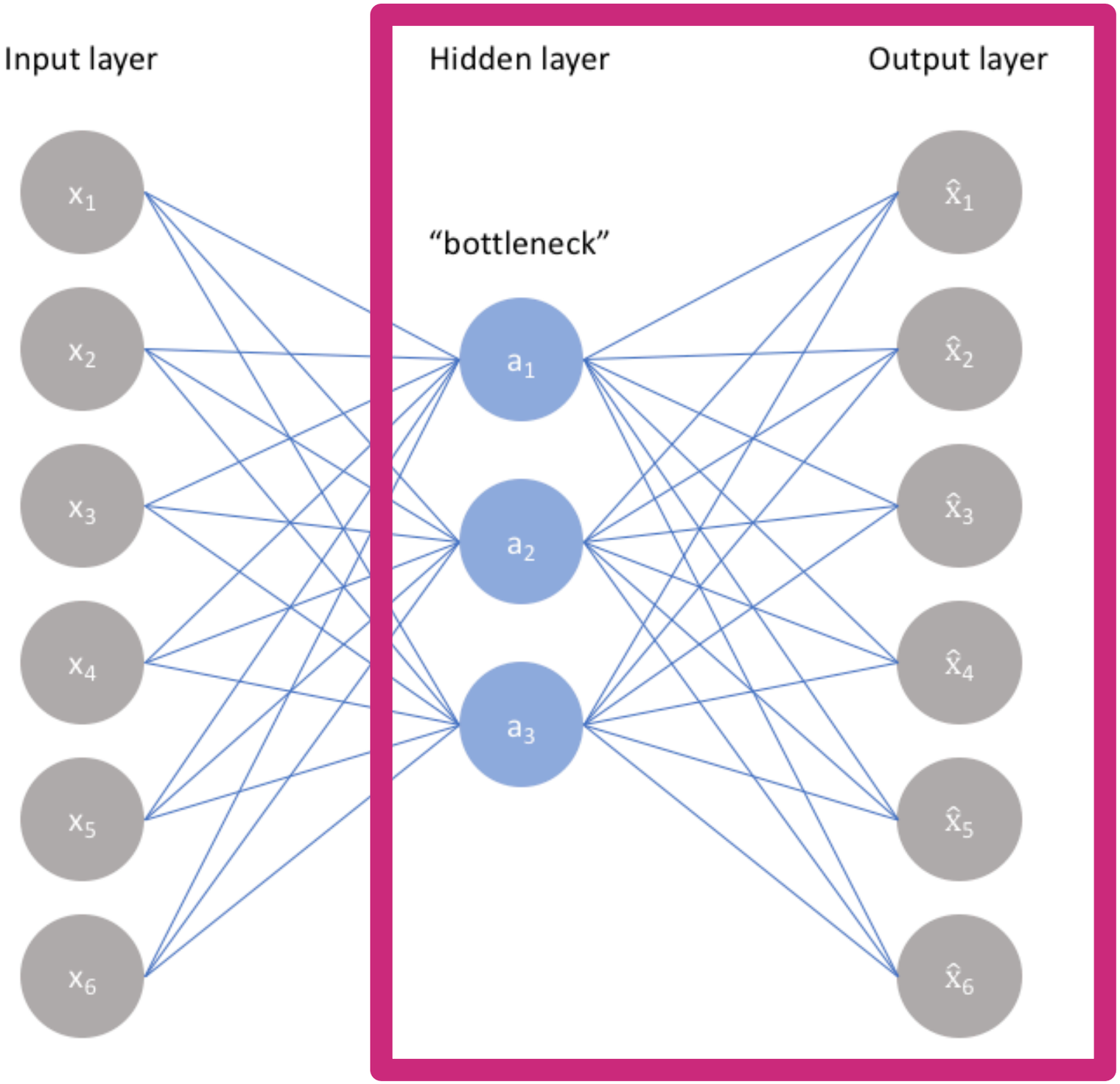
3 main parts:

Encoder

Bottleneck (code)

https://www.v7labs.com/blog/autoencoders-guide#autoencoders-intro

Autoencoder: basic architecture



3 main parts:

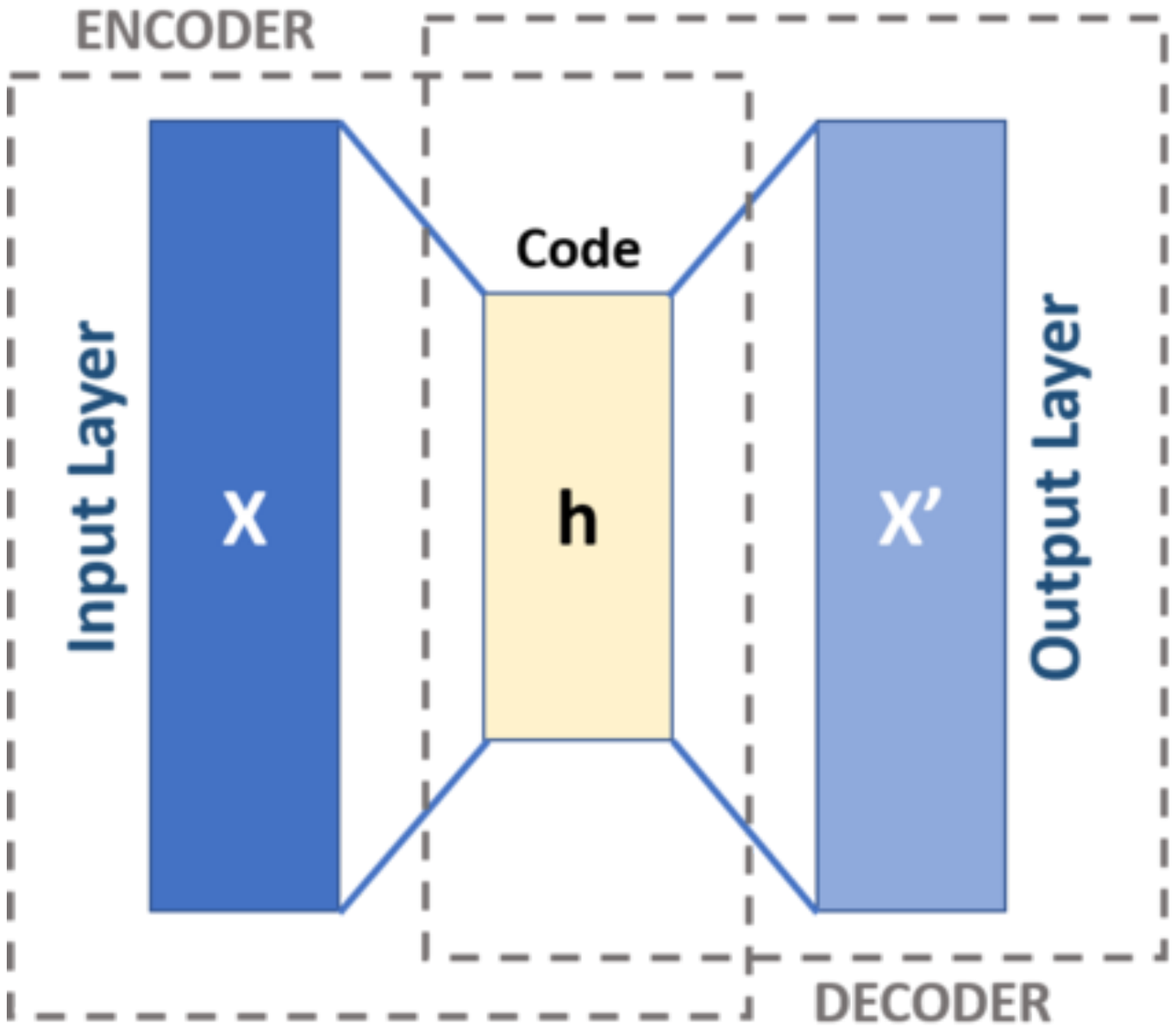
Encoder

Bottleneck (code)

Decoder

https://www.v7labs.com/blog/autoencoders-guide#autoencoders-intro

Autoencoder: basic architecture



3 main parts:

Encoder

Bottleneck (code)

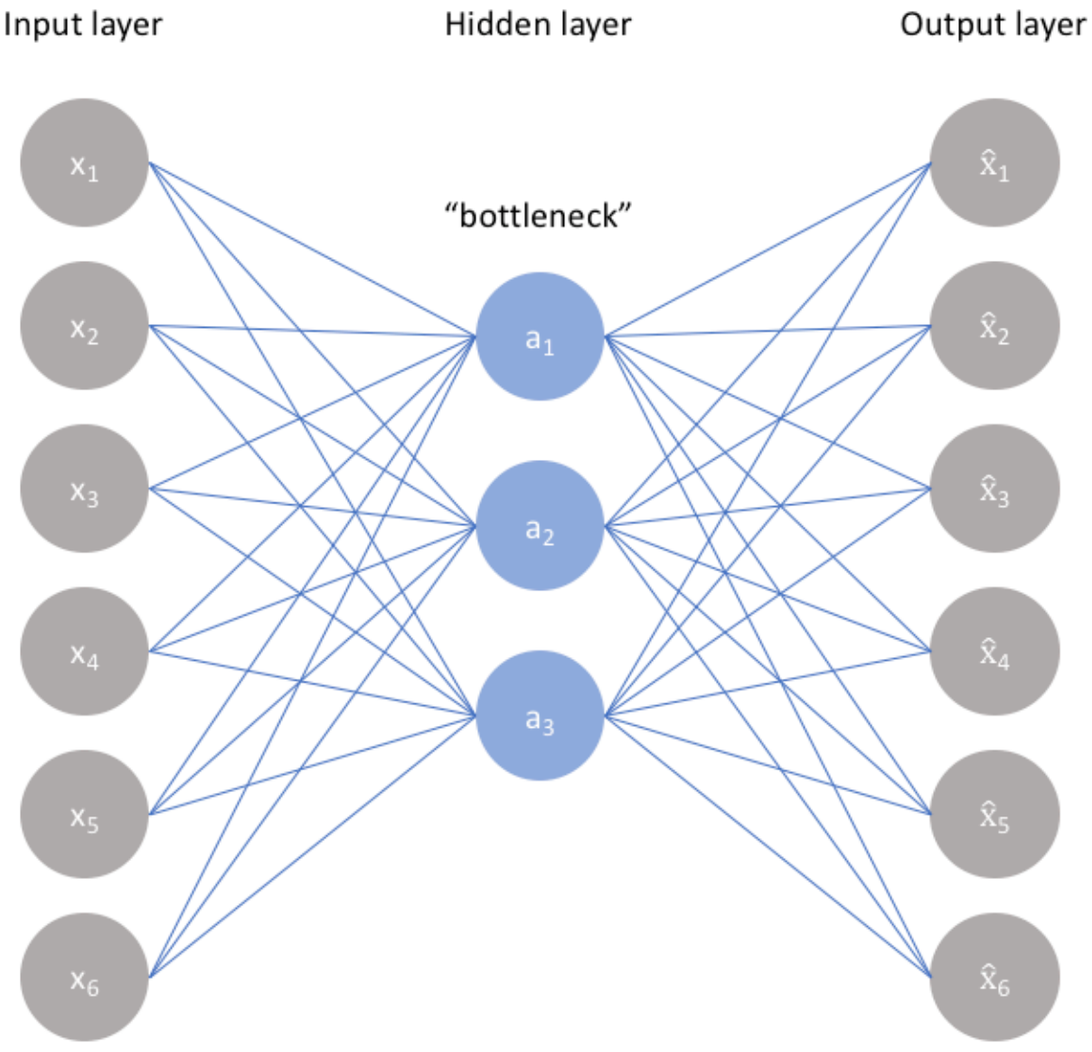
Decoder

Image from https://en.wikipedia.org/wiki/Autoencoder#Basic_architecture

How to train an Autoencoder?

4 choices:

- Code size
- Number of layers
- Number of nodes per layer
- Reconstruction loss

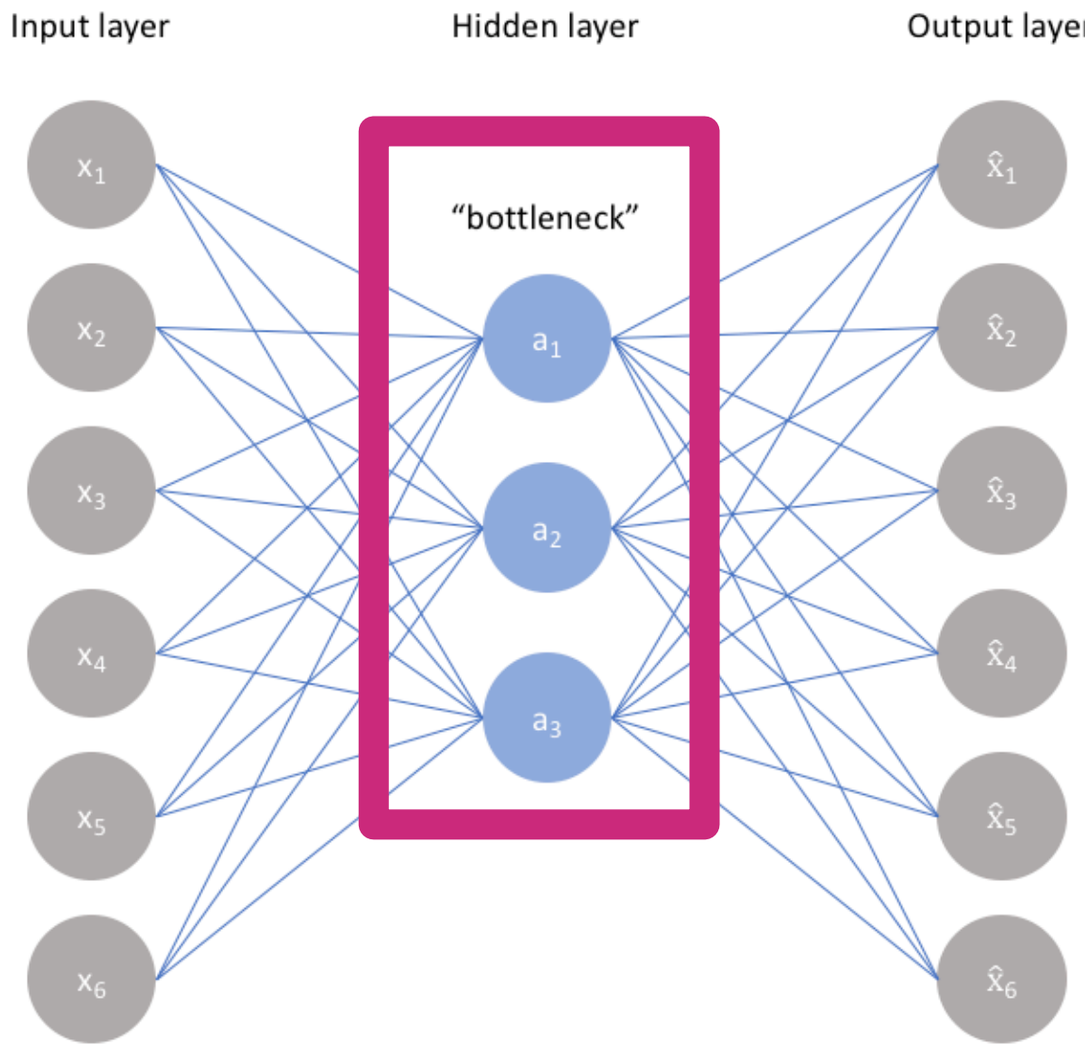


-

How to train an Autoencoder?

4 choices:

- Code size
- Number of layers
- Number of nodes per layer
- Reconstruction loss



Metaphor:
(big-small) table to sort objects

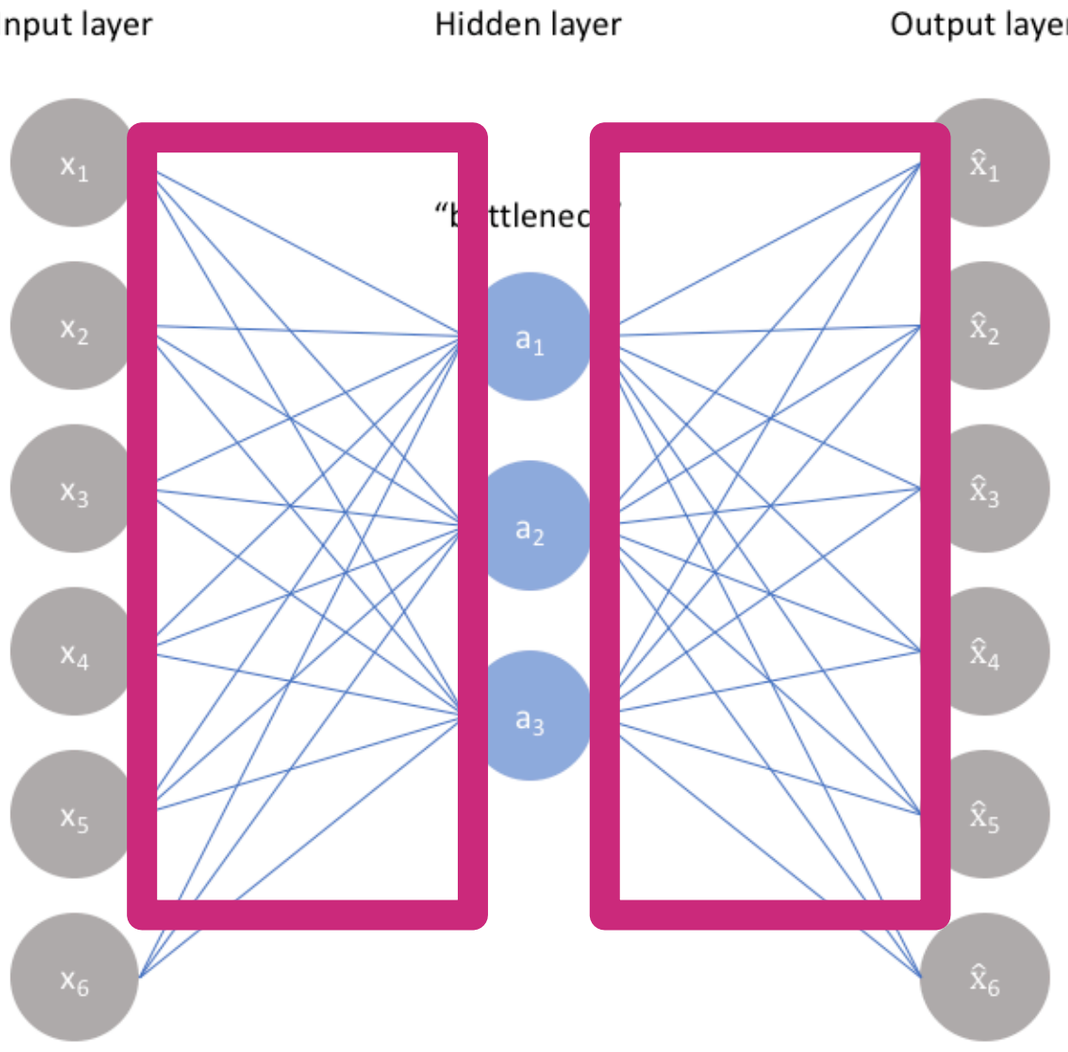
Too big: no "grouping" needed - potential overfit - lack of interest

Too small: loss of information - objects piled up - no way to recover

How to train an Autoencoder?

4 choices:

- Code size
- Number of layers
- Number of nodes per layer
- Reconstruction loss



Same choice as before:

Go deep vs too many parameters

How to train an Autoencoder?

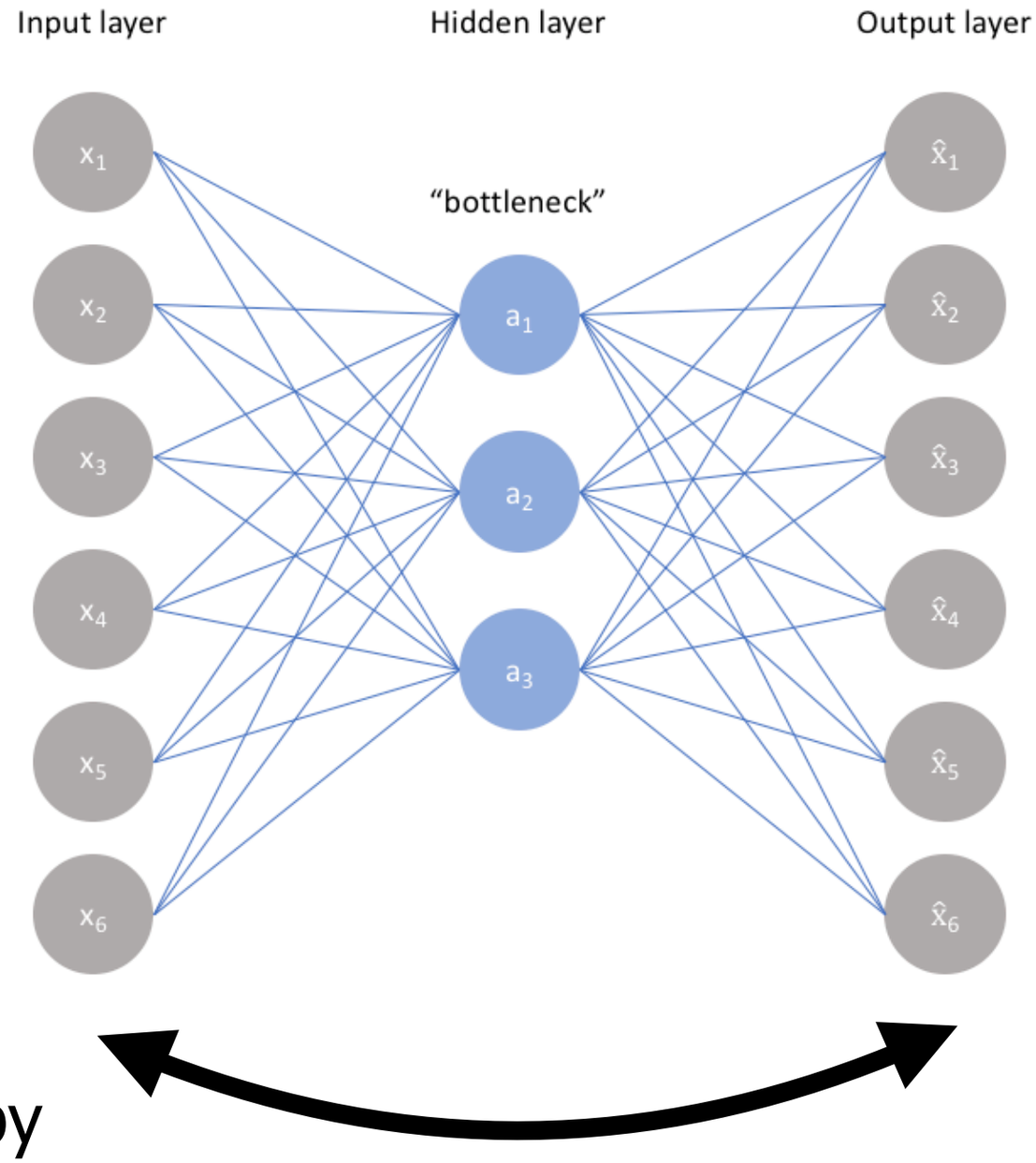
4 choices:

- Code size
- Number of layers
- Number of nodes per layer
- Reconstruction loss

Highly dependent on the type of input and output.

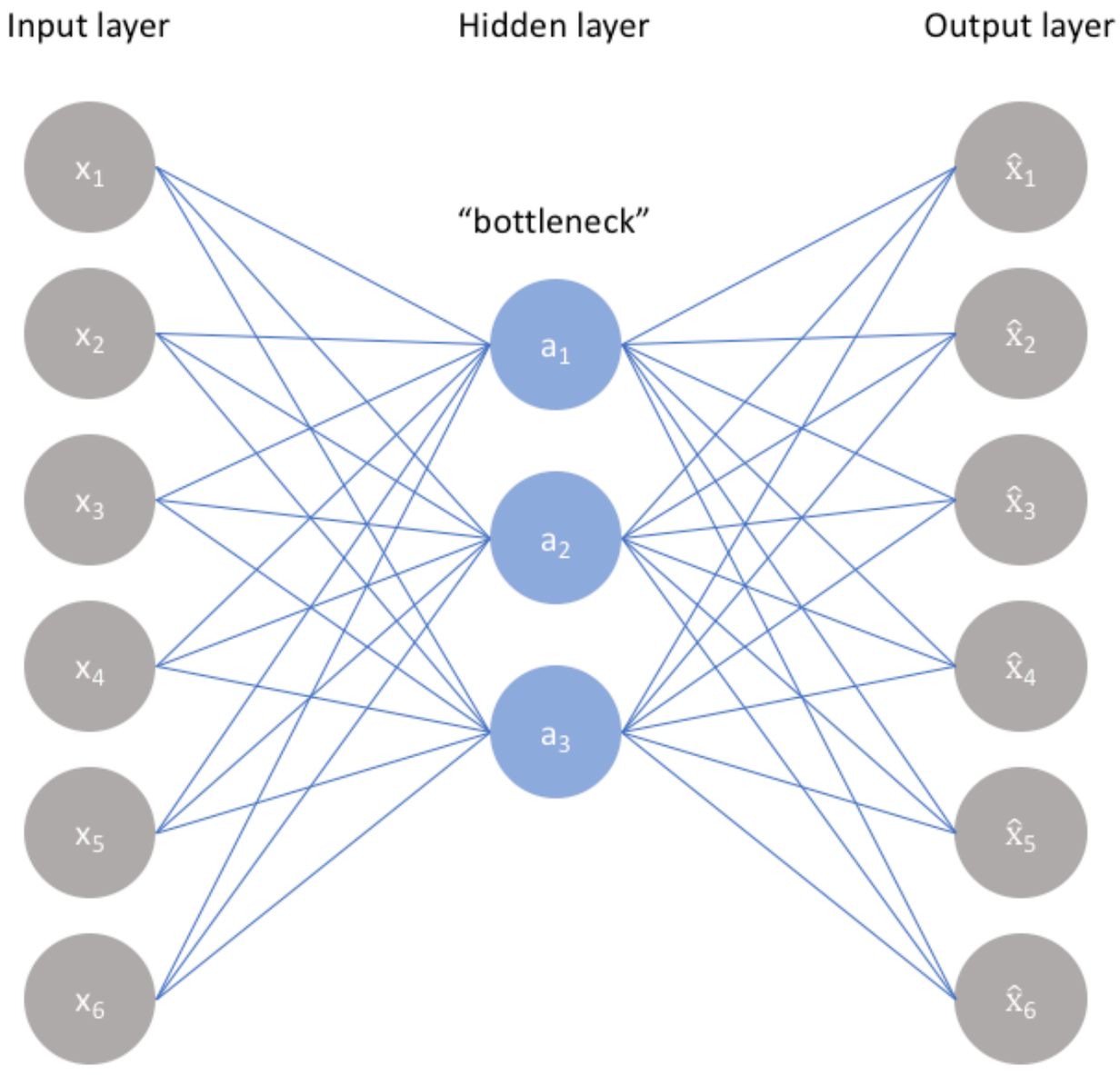
Exemples:

- Images: MSE or L1 loss
- Binary images: Binary cross entropy
- Motion - Displacements ?
- …



Good enough?

Autoencoder: other architectures

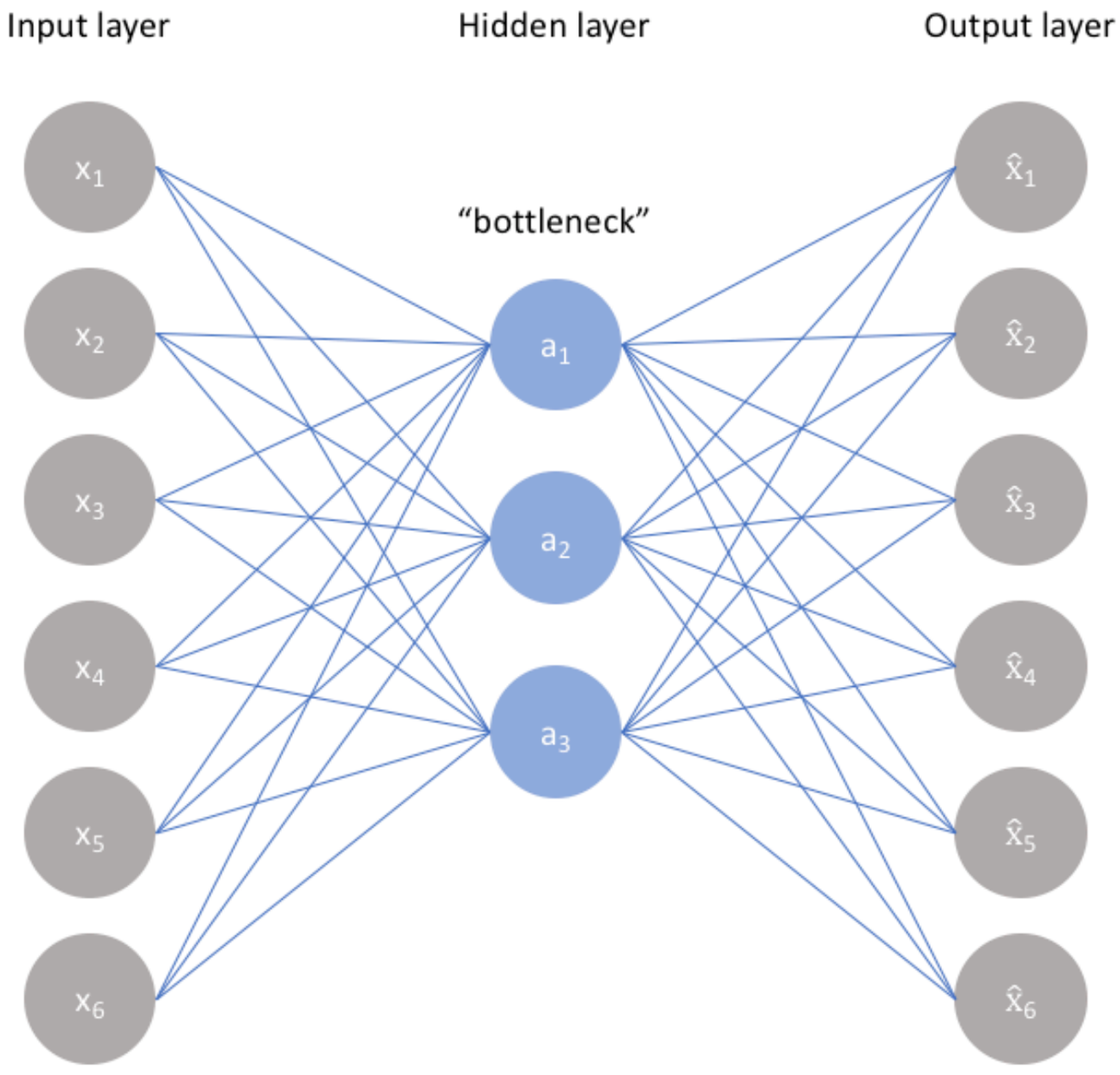


-

- Undercomplete
- Sparse
- Contractive
- Denoising
- Variational

https://www.v7labs.com/blog/autoencoders-guide#autoencoders-intro

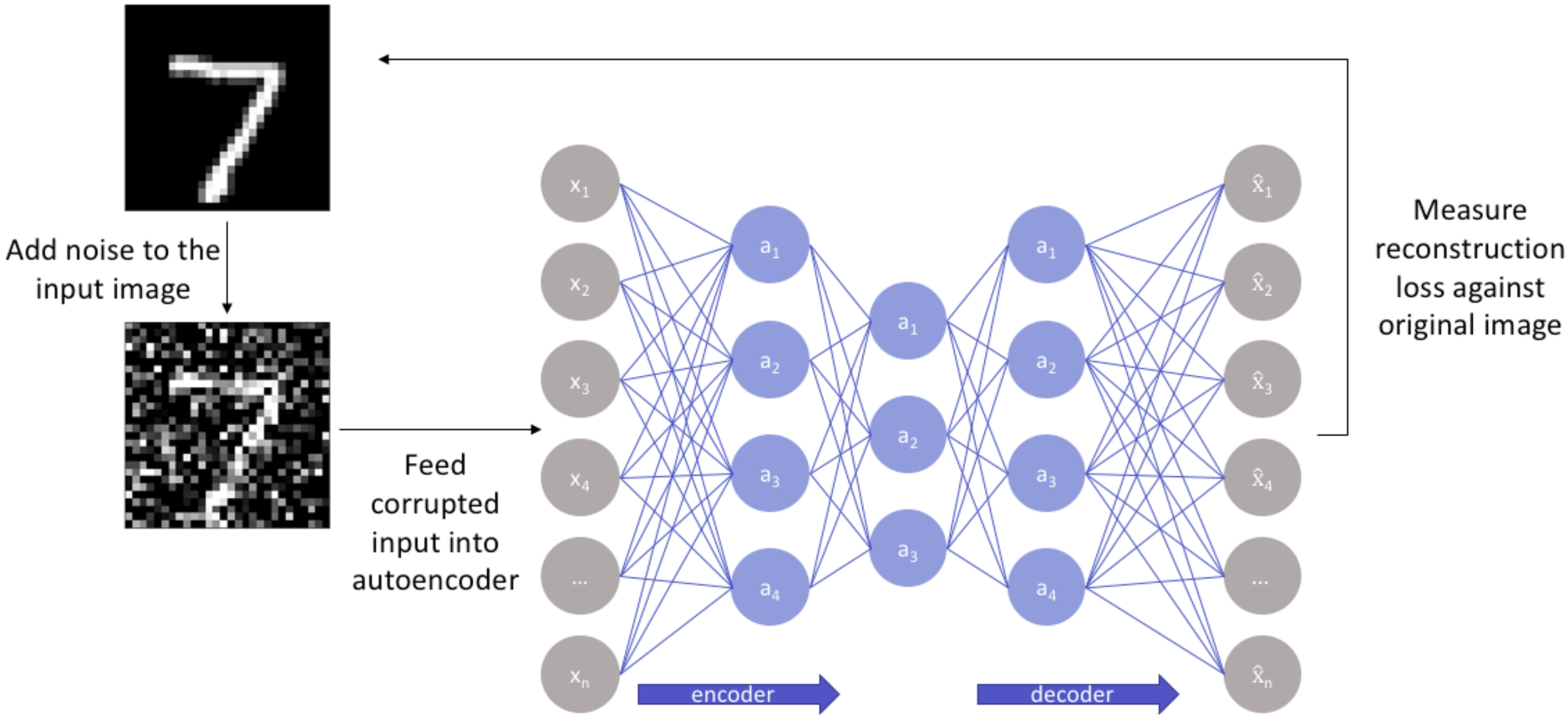
Autoencoder: other architectures



- Undercomplete
- Sparse
- Contractive
- Denoising
- Variational

https://www.v7labs.com/blog/autoencoders-guide#autoencoders-intro

Denoising Autoencoder



Data augmentation

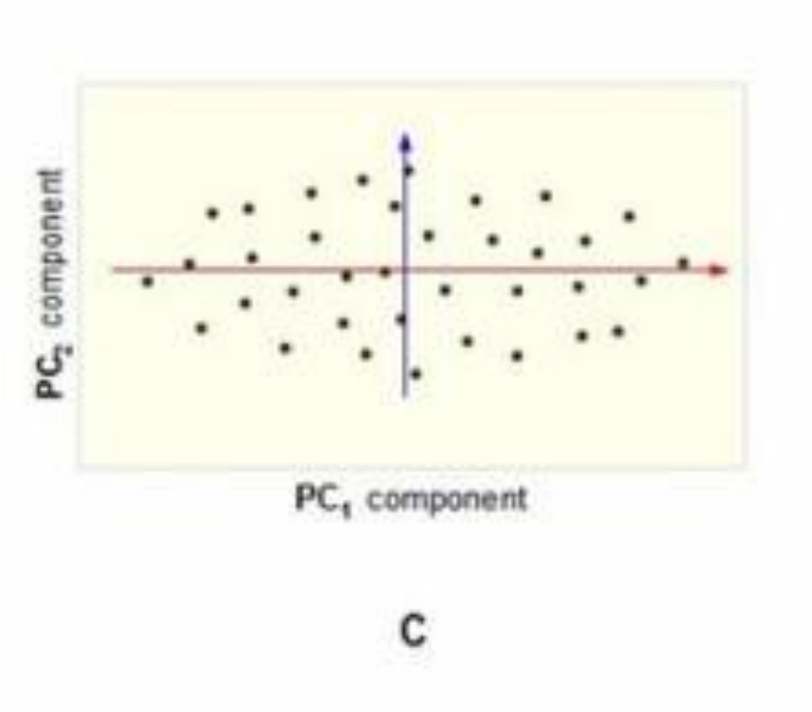
https://www.v7labs.com/blog/autoencoders-guide#autoencoders-intro

Variational Autoencoder

Motivation: can we interpolate in the latent space?

- For PCA: yes!

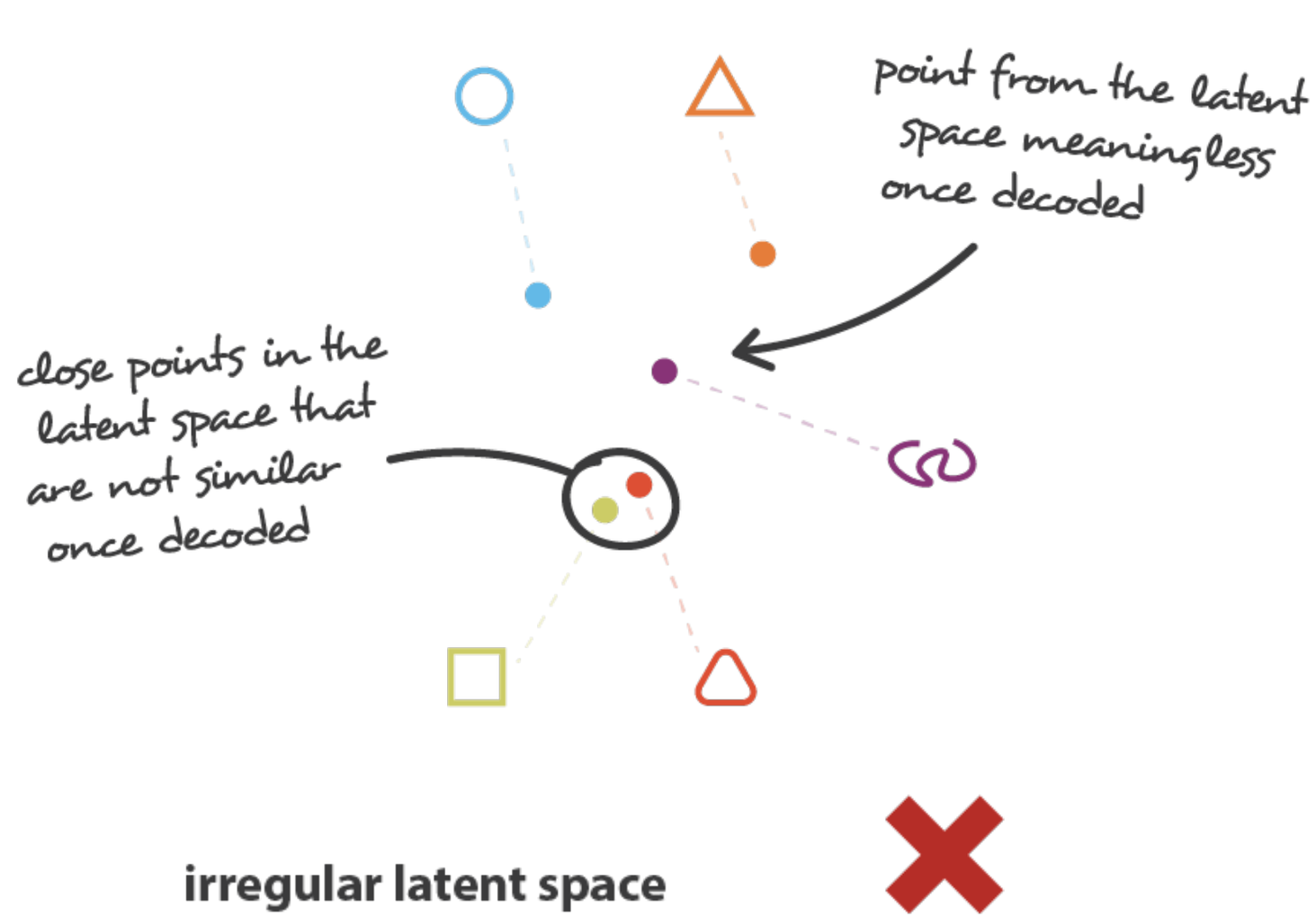
  We can generate new data that is "meaningful"

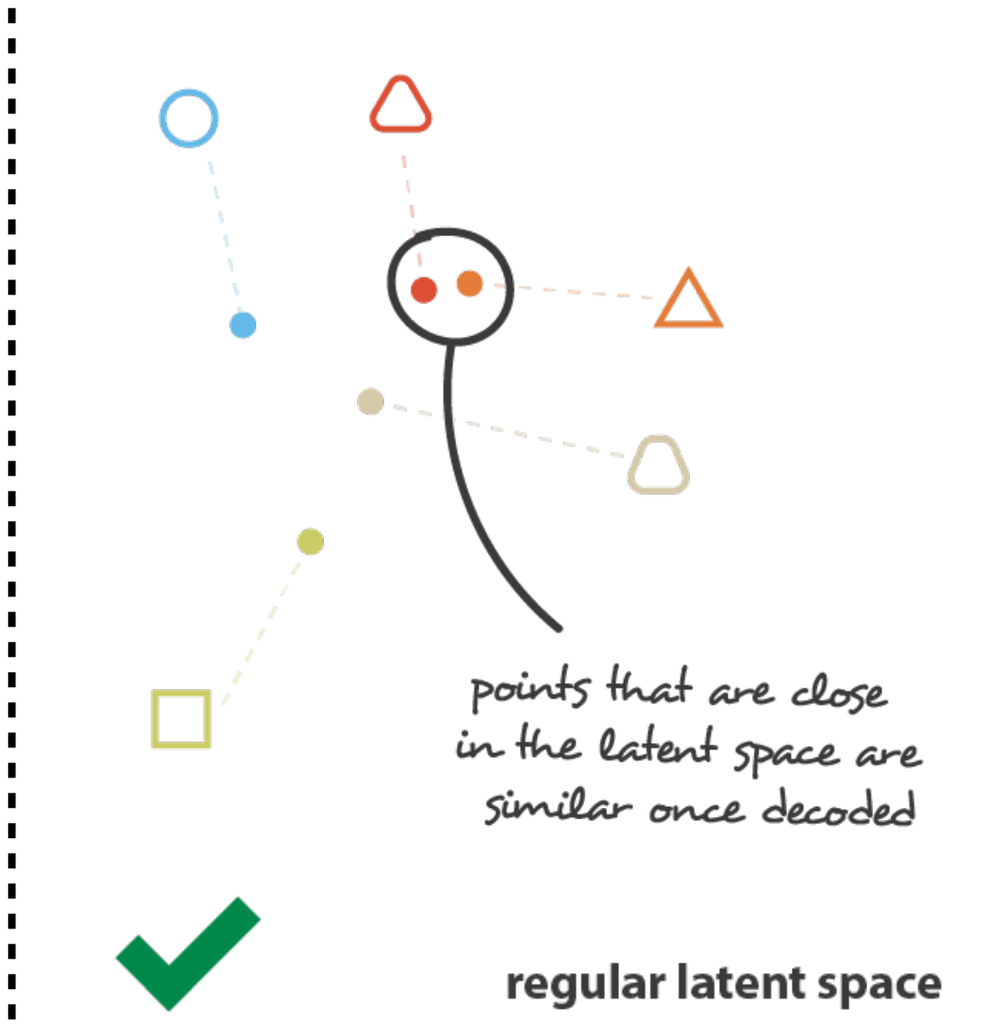


- For AE: Not sure… nobody asked!

Variational Autoencoder

Motivation: can we interpolate in the latent space?





https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder

Motivation: can we interpolate in the latent space?

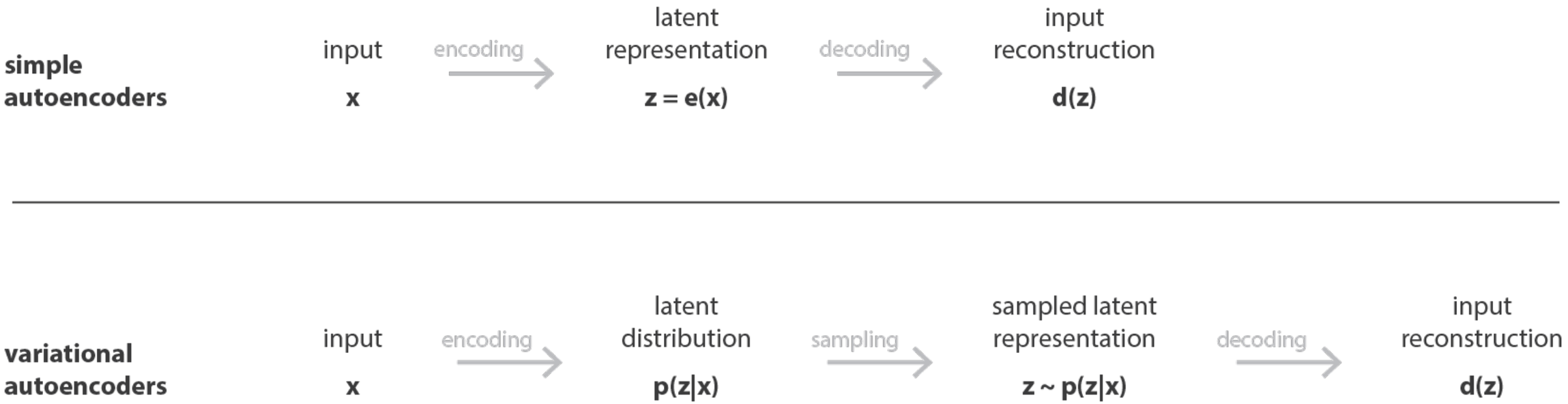
A **variational auto** encoder can be defined as an auto encoder whose training is **regularized** to avoid overfitting and ensure that the latent space has good properties that enable generative process.

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder

Instead to encode an input x as a single point $e(x) = z$

it is encoded as a distribution $e(x) = p(z|x)$ over the latent space.

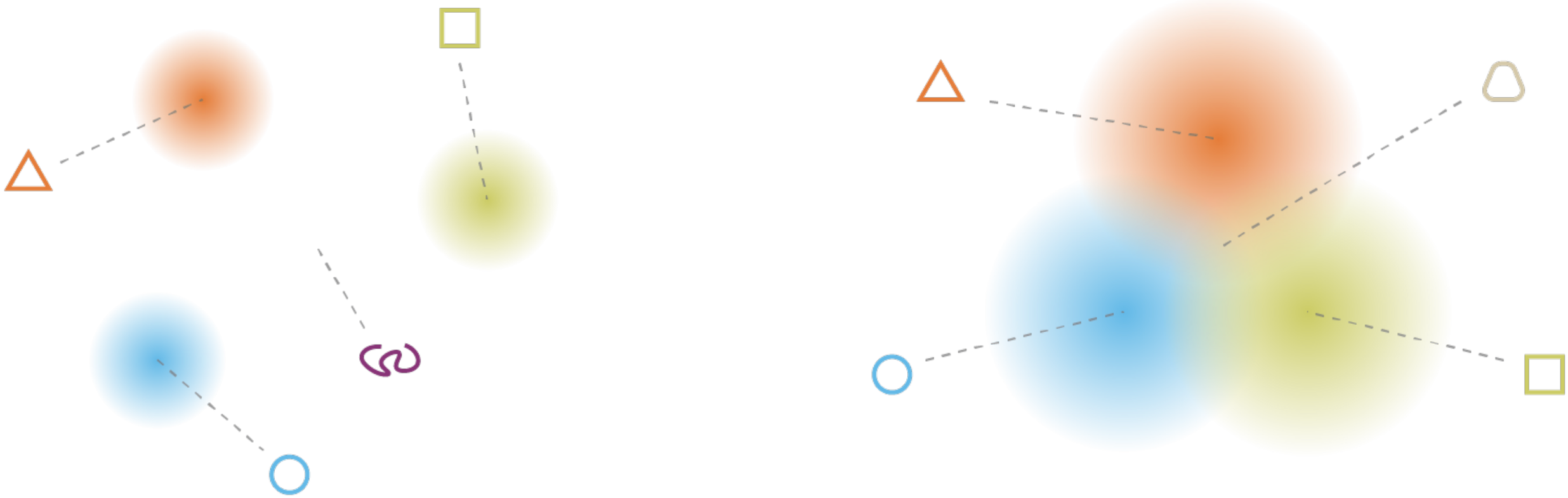


$p(z|x)$ is usually chosen as a normal distribution $\mathcal{N}(\mu, \sigma)$

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder

How to regularize?



what can happen without regularisation

what we want to obtain with regularisation

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

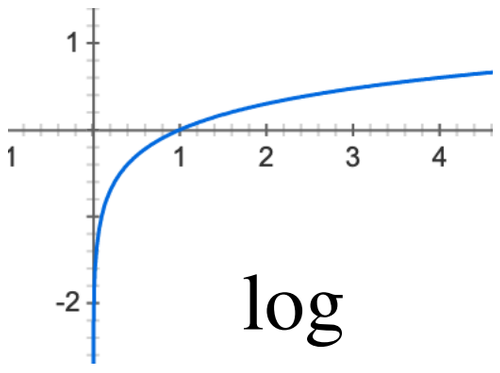
Variational Autoencoder

How to regularize?

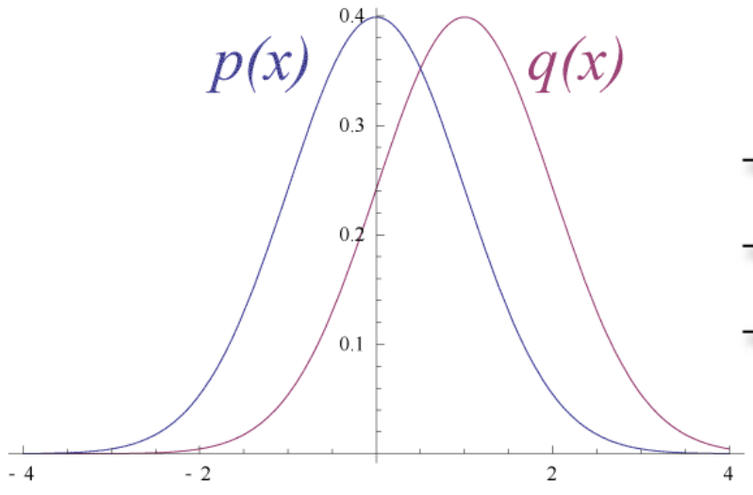
Kullback-Liebler divergence (D_{KL}, KL-loss, or relative entropy):

Measures how one probability distribution P is different from a second Q:

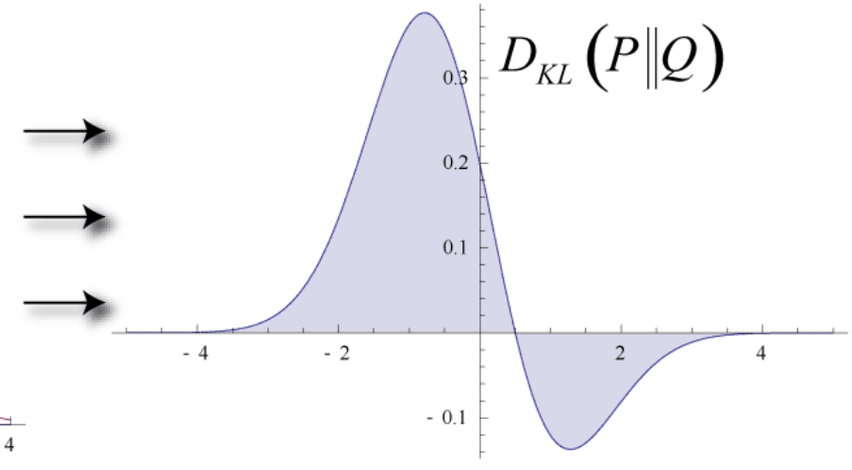
$$D_{KL}(P||Q) = \int_{-\infty}^{+\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$



log



Original Gaussian PDF's



KL Area to be Integrated

Variational Autoencoder

How to regularize?

Kullback-Liebler divergence (D_{KL}, KL-loss, or relative entropy):

Measures how one probability distribution P is different from a second Q:

Closed form for $\mathcal{N}_0, \mathcal{N}_1$:

$$D_{\mathrm{KL}}\left(\mathcal{N}_0 \parallel \mathcal{N}_1\right) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + \left(\mu_1 - \mu_0\right)^{\mathsf{T}}\Sigma_1^{-1}\left(\mu_1 - \mu_0\right) - k + \ln\left(\frac{\det\Sigma_1}{\det\Sigma_0}\right)\right)$$

And the derivative?

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder

Kullback-Liebler divergence from an optimization perspective:

$$
\begin{aligned}
\mathbf{KL}(q_\theta || p) &= \sum_d q(d) \log\left(\frac{q(d)}{p(d)}\right) \\
&= \sum_d q(d) \left(\log q(d) - \log p(d)\right) \\
&= \underbrace{\sum_d q(d) \log q(d)}_{-\text{entropy}} - \underbrace{\sum_d q(d) \log p(d)}_{\text{cross-entropy}}
\end{aligned}
$$

https://timvieira.github.io/blog/post/2014/10/06/kl-divergence-as-an-objective-function/

Variational Autoencoder

Kullback-Liebler divergence from an optimization perspective:

$$\underset{\theta}{\operatorname{argmin}}\, \mathbf{KL}(q_\theta||p) = \underset{\theta}{\operatorname{argmin}} \sum_d q_\theta(d) \log q_\theta(d) - \sum_d q_\theta(d) \log \bar{p}(d)$$

$$\nabla \left[\sum_d q_\theta(d) \log q_\theta(d) - \sum_d q_\theta(d) \log \bar{p}(d) \right]$$

$$= \sum_d \nabla \left[q_\theta(d) \log q_\theta(d) \right] - \sum_d \nabla \left[q_\theta(d) \right] \log \bar{p}(d)$$

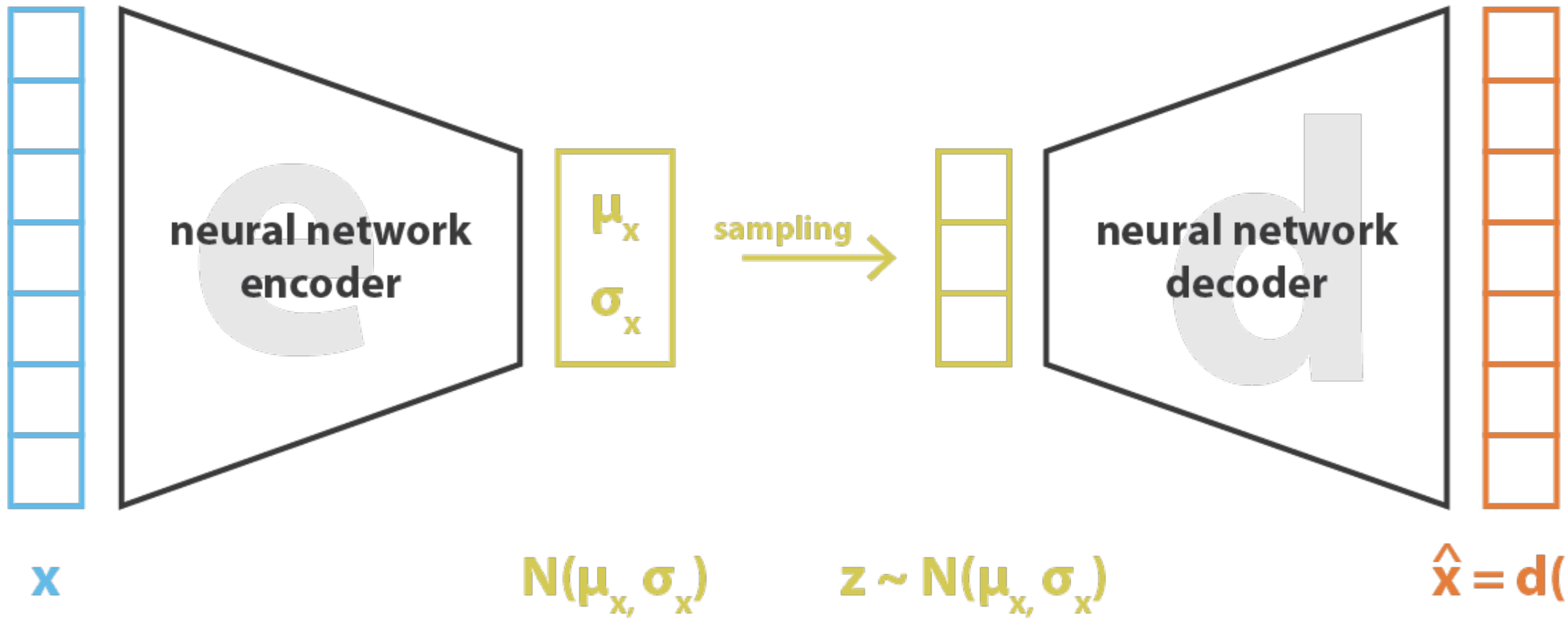
$$= \sum_d \nabla \left[q_\theta(d) \right] (1 + \log q_\theta(d)) - \sum_d \nabla \left[q_\theta(d) \right] \log \bar{p}(d)$$

$$= \sum_d \nabla \left[q_\theta(d) \right] (1 + \log q_\theta(d) - \log \bar{p}(d))$$

$$= \sum_d \nabla \left[q_\theta(d) \right] (\log q_\theta(d) - \log \bar{p}(d))$$

https://timvieira.github.io/blog/post/2014/10/06/kl-divergence-as-an-objective-function/

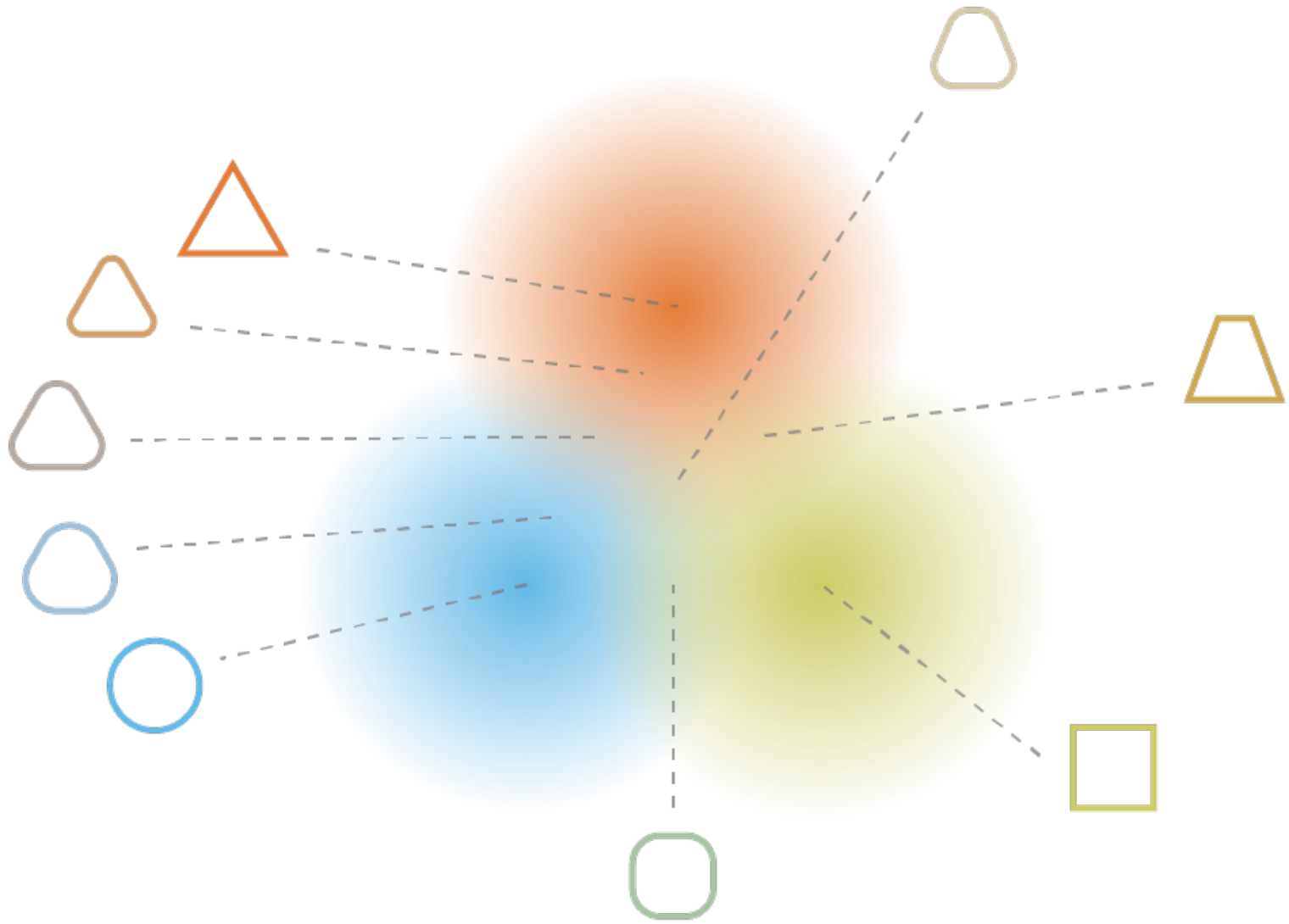
Variational Autoencoder



$$\text{loss} = \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder



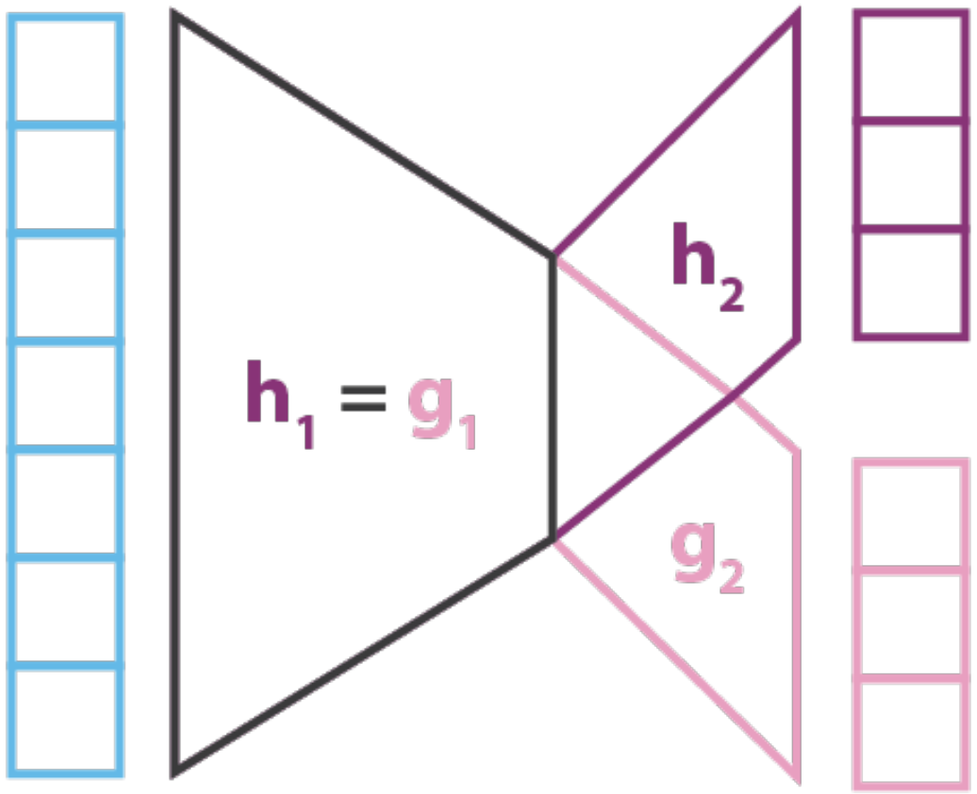
$$e(x_1) = \mathcal{N}(\mu_1, \sigma_1)$$
$$e(x_2) = \mathcal{N}(\mu_2, \sigma_2)$$

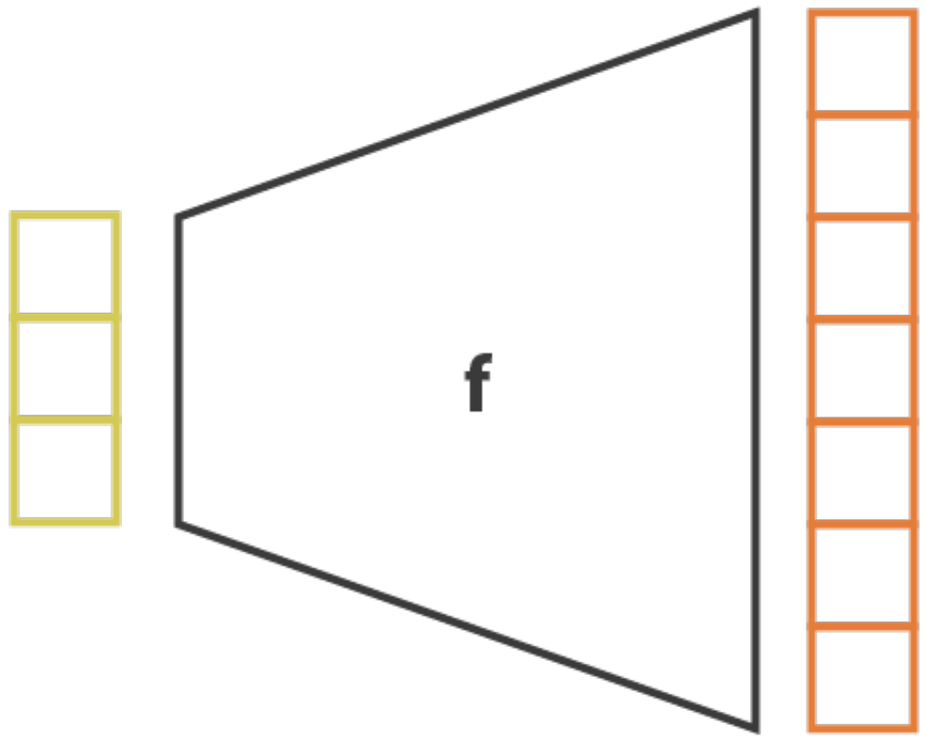
$$z \sim \mathcal{N}(\mu_1, \sigma_1)$$
and also
$$z \sim \mathcal{N}(\mu_2, \sigma_2)$$

$$d(z) \approx \frac{(x_2 + x_1)}{2}$$

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder





x

$\mu_x = g(x) = g_2(g_1(x))$

$\sigma_x = h(x) = h_2(h_1(x))$

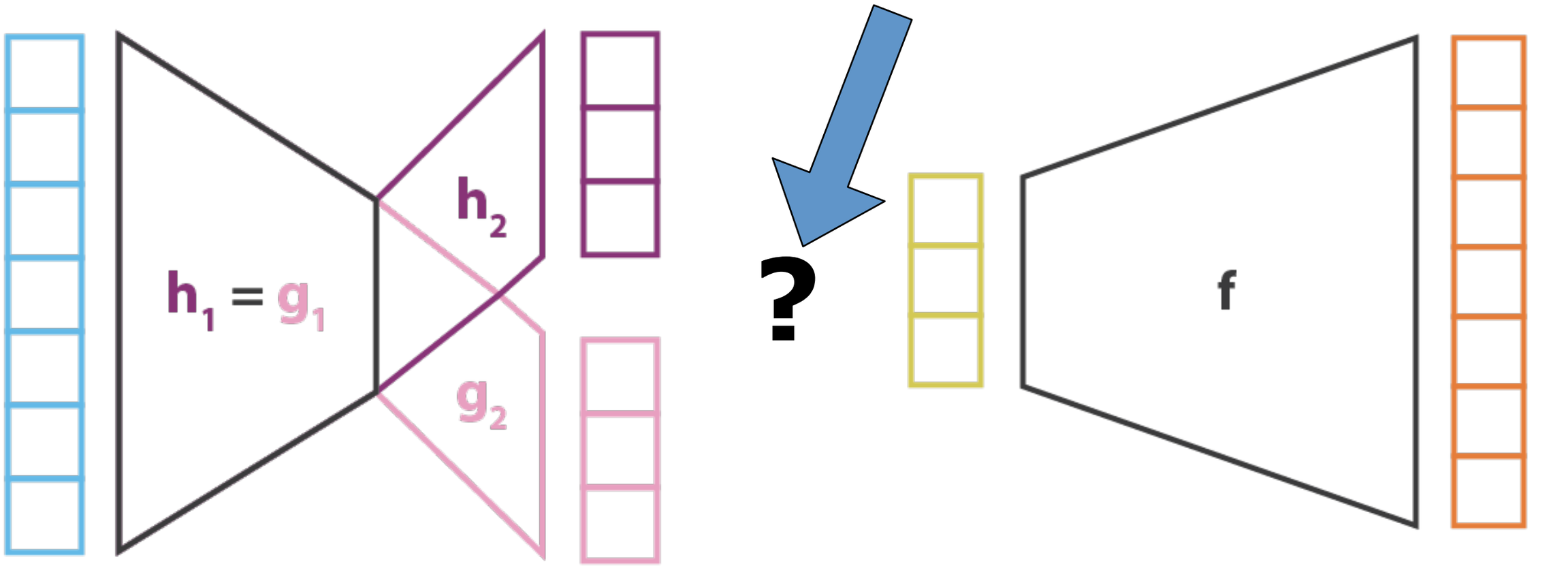
z

$\hat{x} = f(z)$

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder

How to sample to allow back propagation?

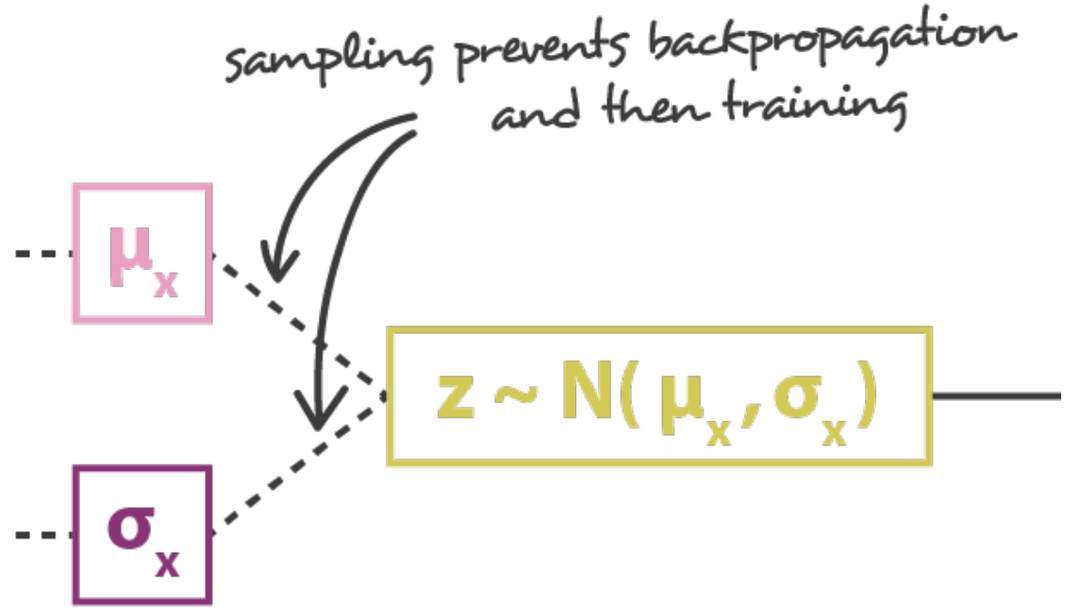


https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

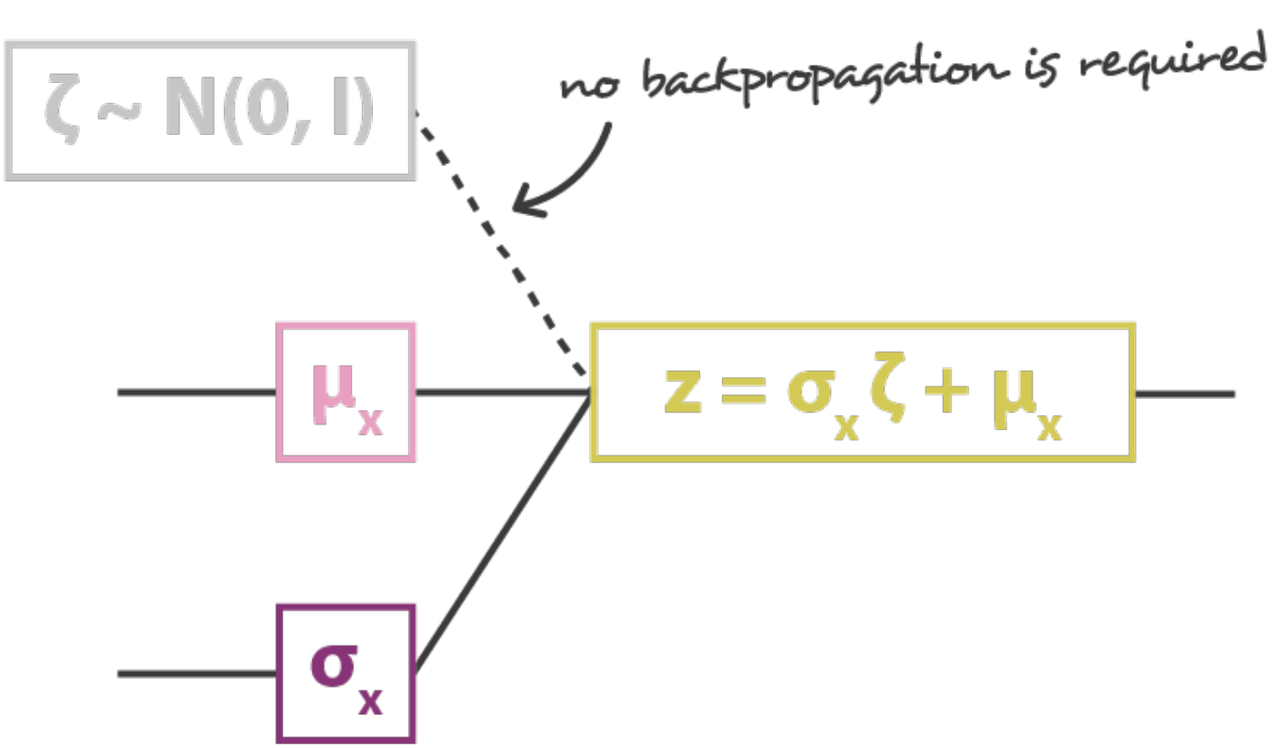
Variational Autoencoder

Reparametrization trick

—— no problem for backpropagation - - - - - backpropagation is not possible due to sampling



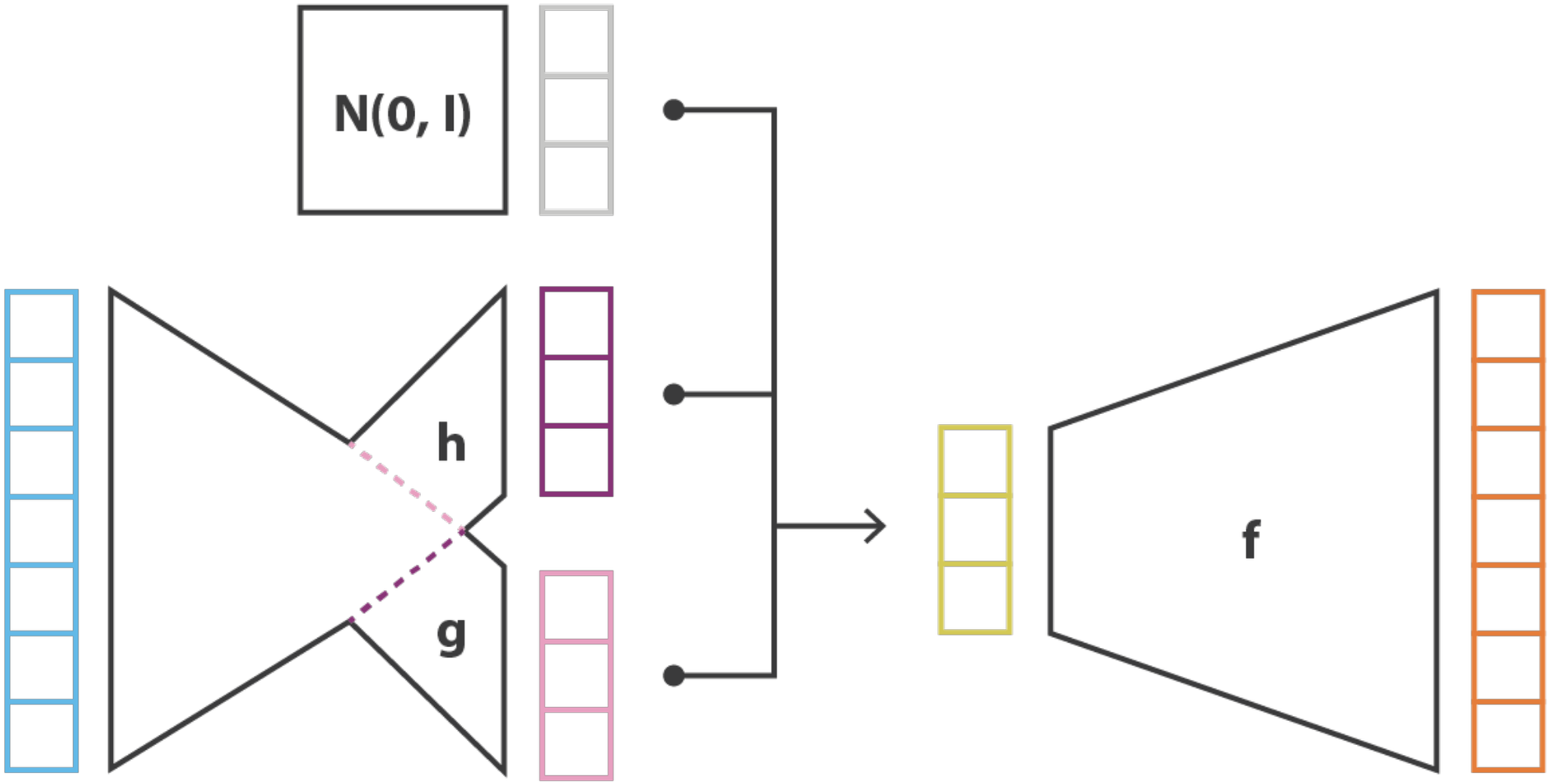
sampling without reparametrisation trick



sampling with reparametrisation trick

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder



x

$\mu_x = g(x)$

$\sigma_x = h(x)$

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

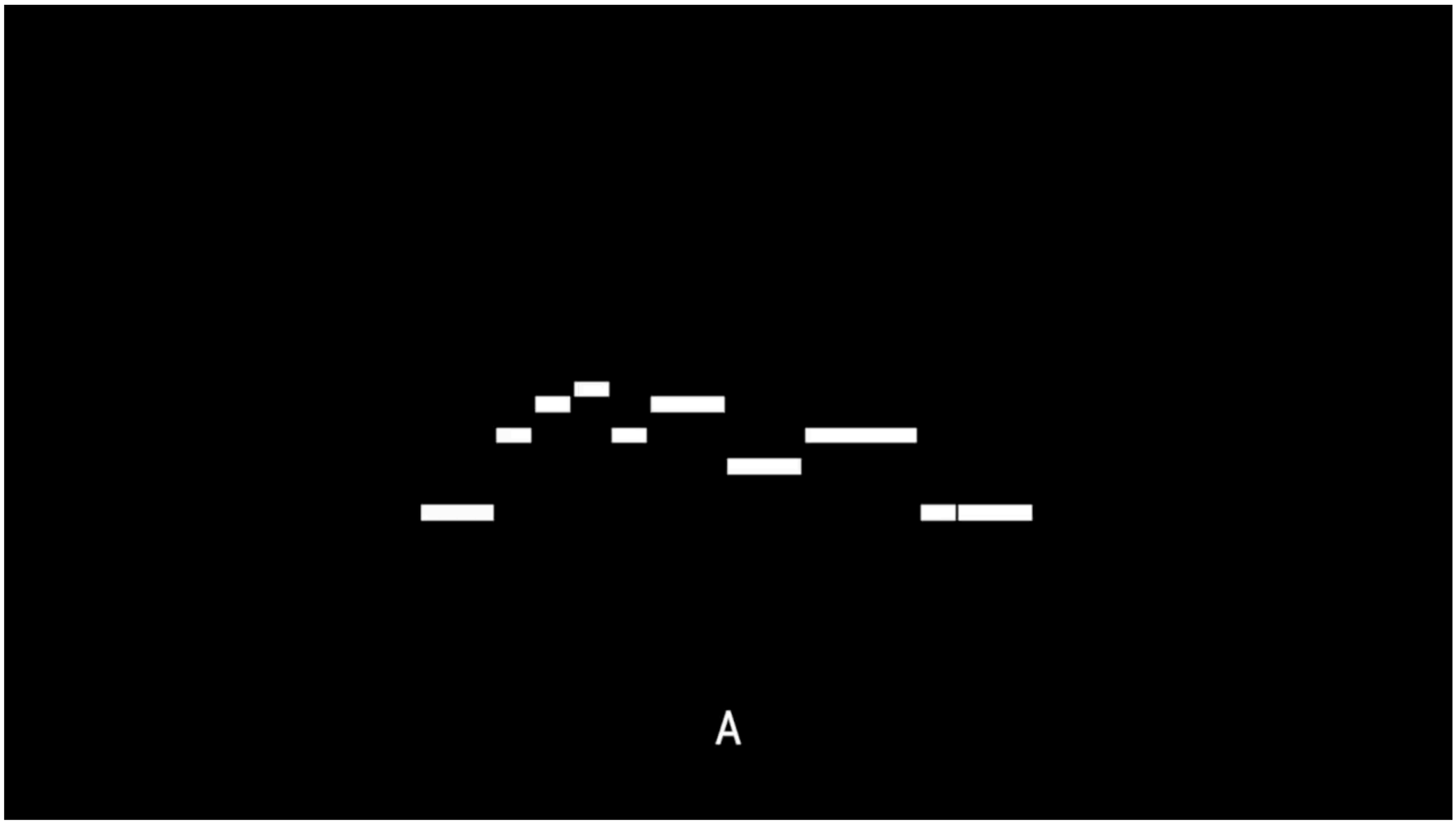
$z = \sigma_x \zeta + \mu_x$

$\hat{x} = f(z)$

$$\text{loss} = C \,||\, x - \hat{x} \,||^2 + KL[\, N(\mu_x, \sigma_x), N(0, I) \,] = C \,||\, x - f(z) \,||^2 + KL[\, N(g(x), h(x)), N(0, I) \,]$$

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder

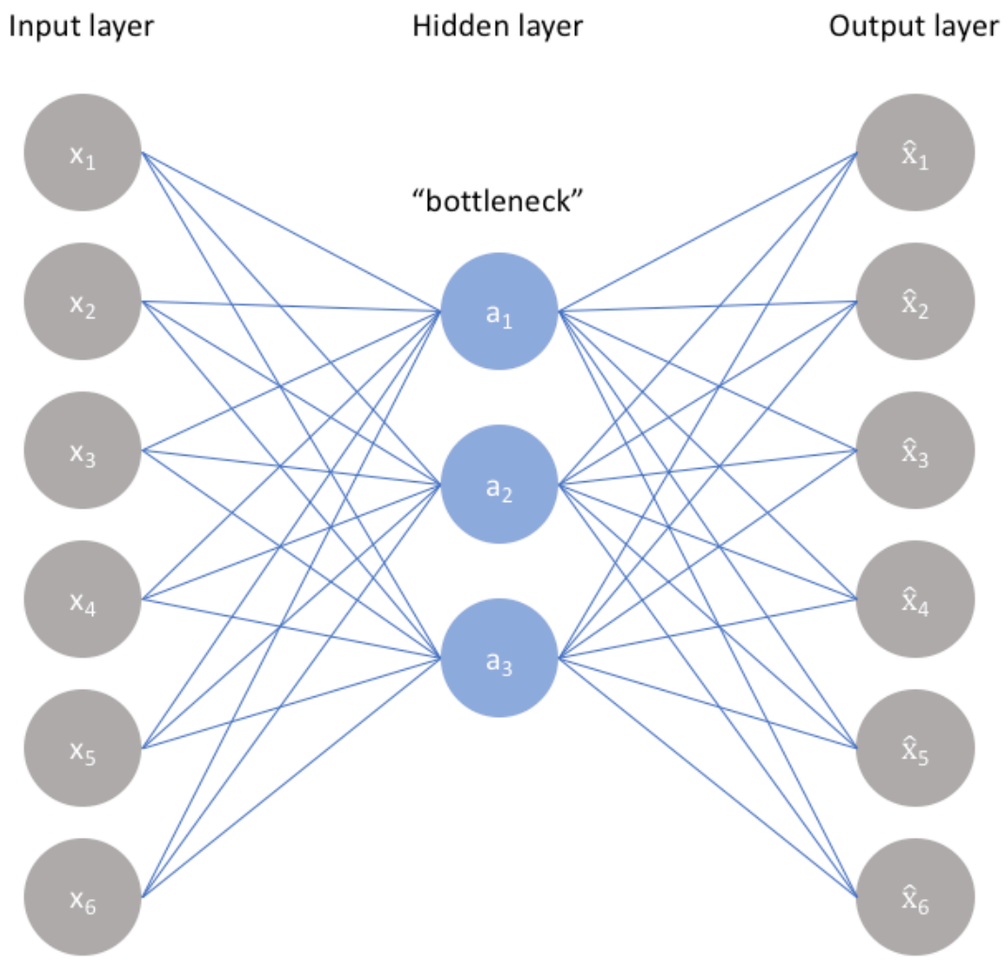


Roberts, Adam, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. "A hierarchical latent vector model for learning long-term structure in music." In *International conference on machine learning*, pp. 4364-4373. PMLR, 2018.

https://magenta.tensorflow.org/music-vae

Autoencoders summary

- Unsupervised learning
- Learns data representations (embeddings)
- Applications:
  - Denoising
  - Compression
  - Anomaly Detection
  - Generation
- VAE and DAE most adopted



https://www.v7labs.com/blog/autoencoders-guide#autoencoders-intro

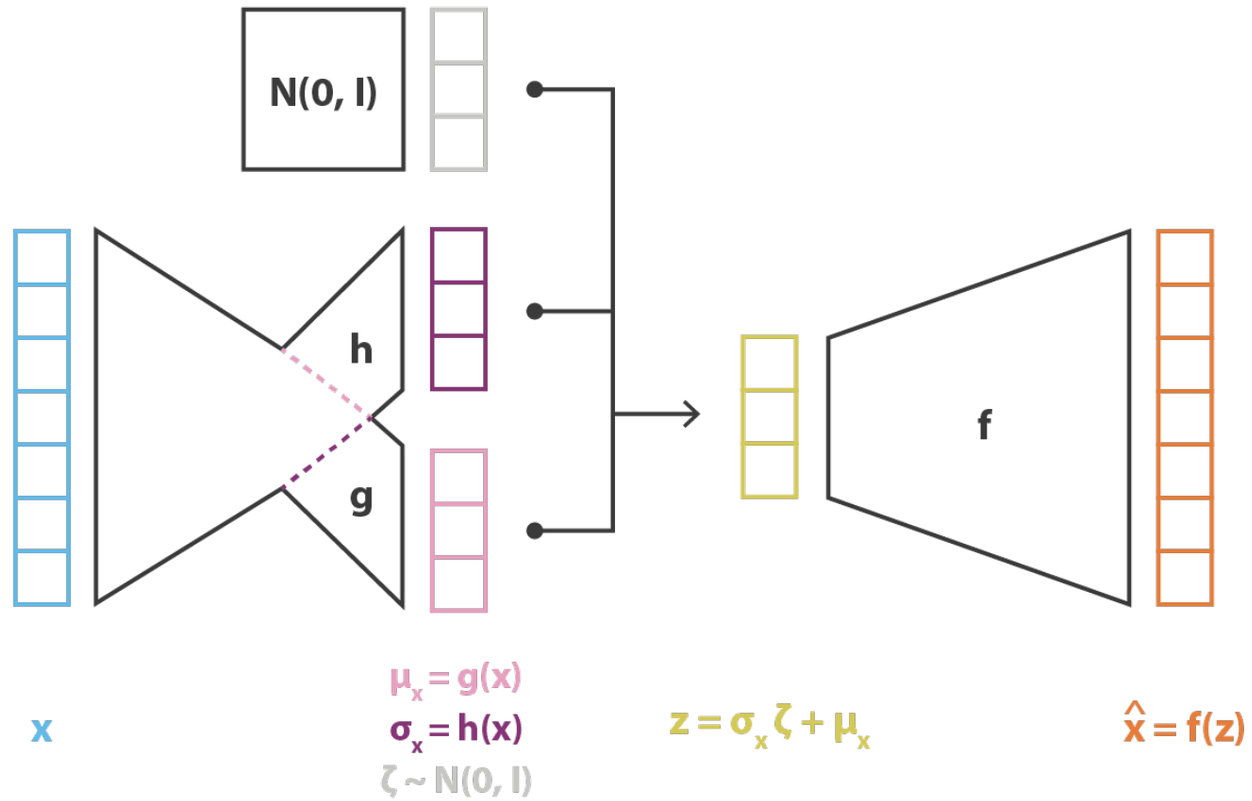
Variational Autoencoders summary

Pros:

- Elegant
- Theoretically pleasing (ELBO)
- Good results

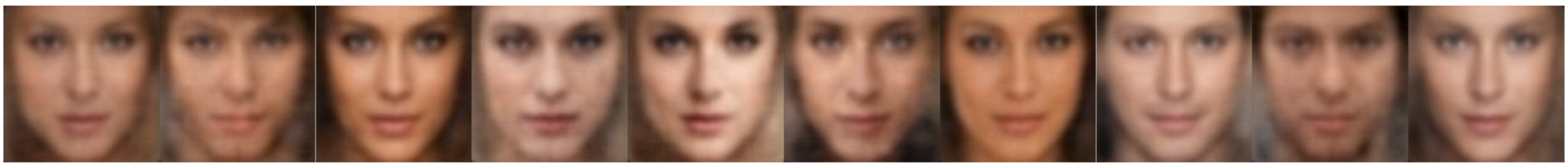
Cons:

- Blurry results

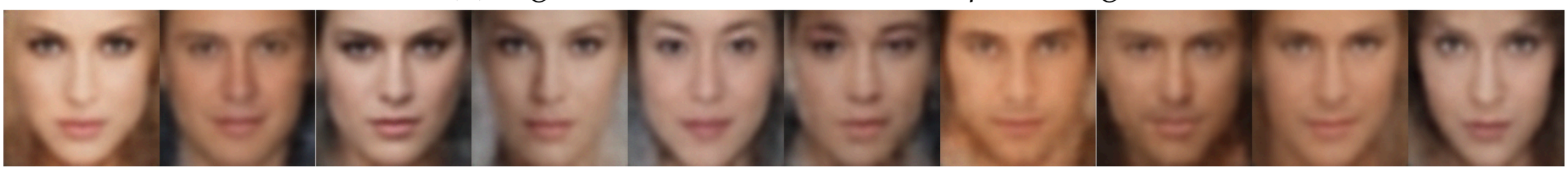


https://www.v7labs.com/blog/autoencoders-guide#autoencoders-intro

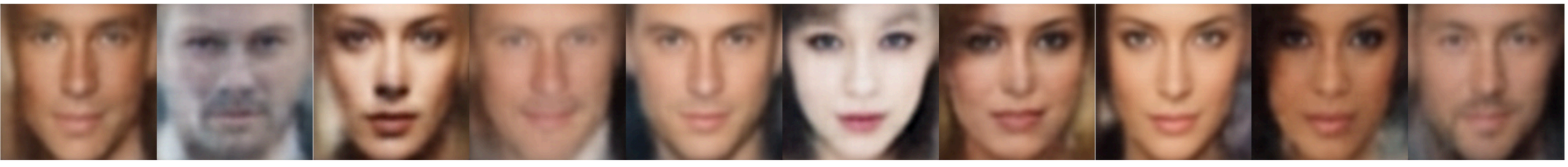
Variational Autoencoder



(b) BigConvNet with Incremental β training



(c) ResNet with $\beta = 2$ training



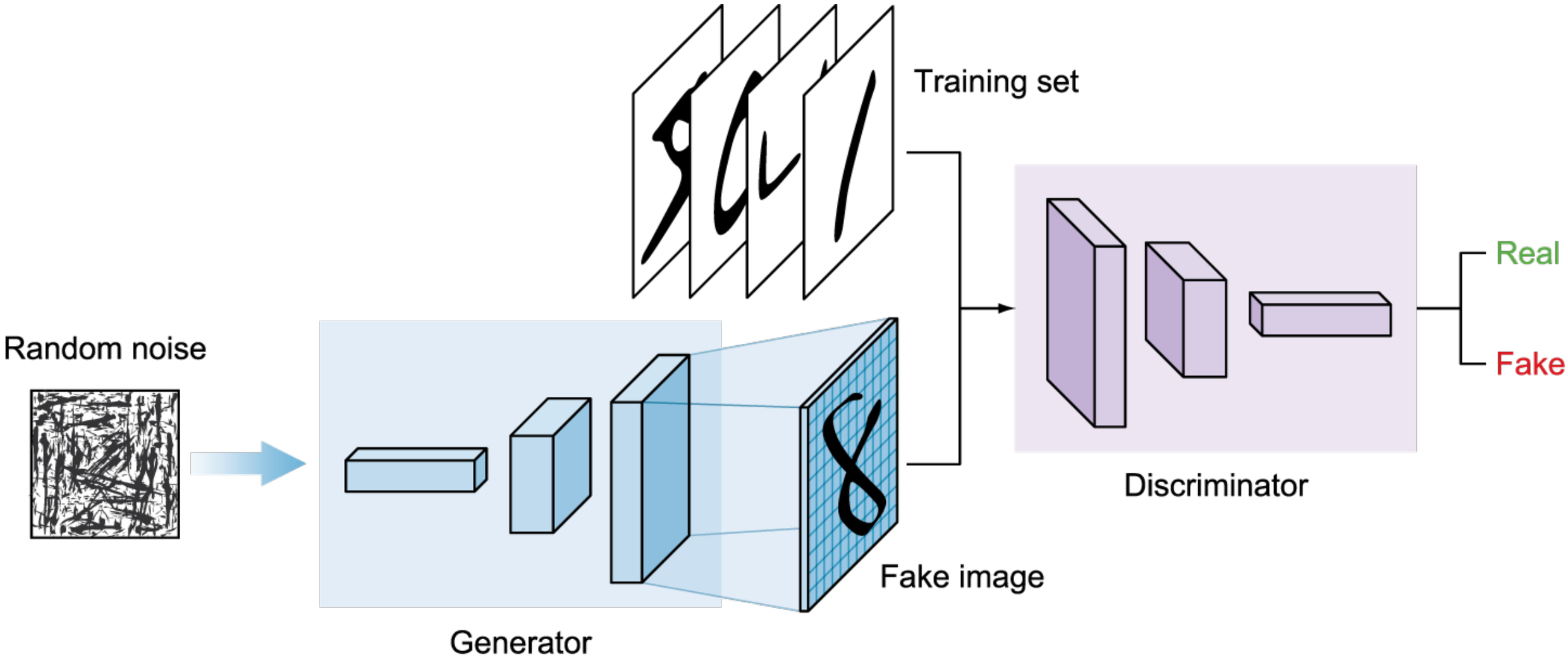
(e) VGG with $\beta = 2$ training

Acosta Hernández, Jorge. "Neutral face generator using variational autoencoders." (2020).

Generative Adversarial Nets (2014)

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." *Advances in neural information processing systems* 27 (2014).

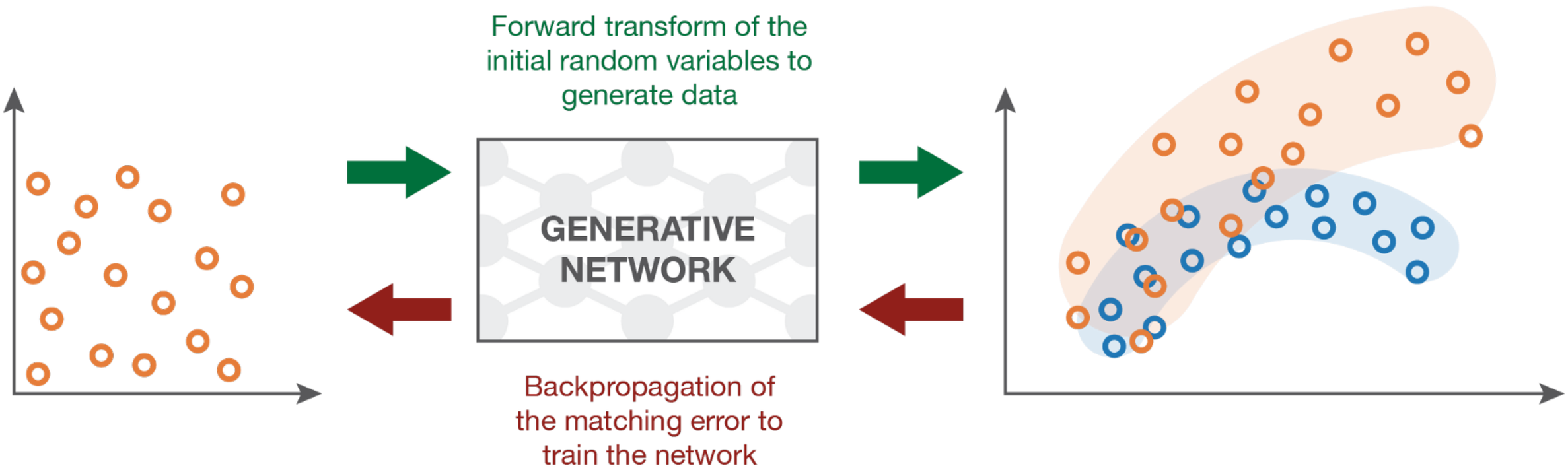
Generative Adversarial Networks



Ref to be updated

Generative Adversarial Networks

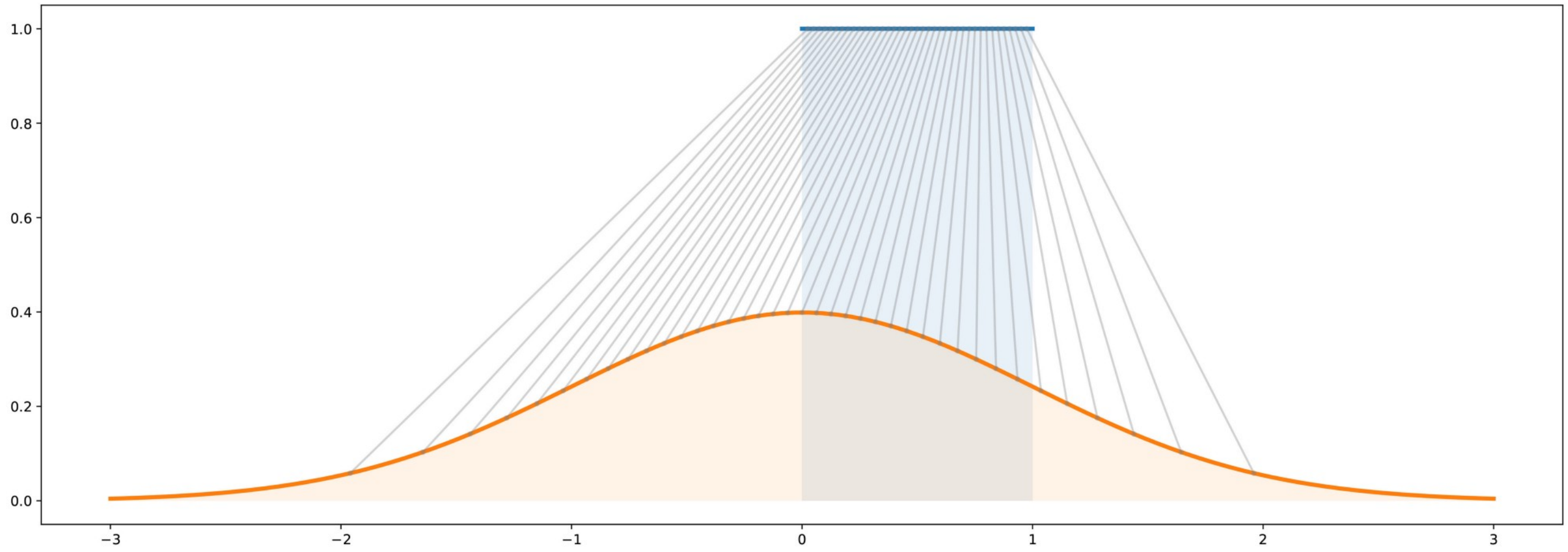
The game of distribution matching



https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Generating random variables

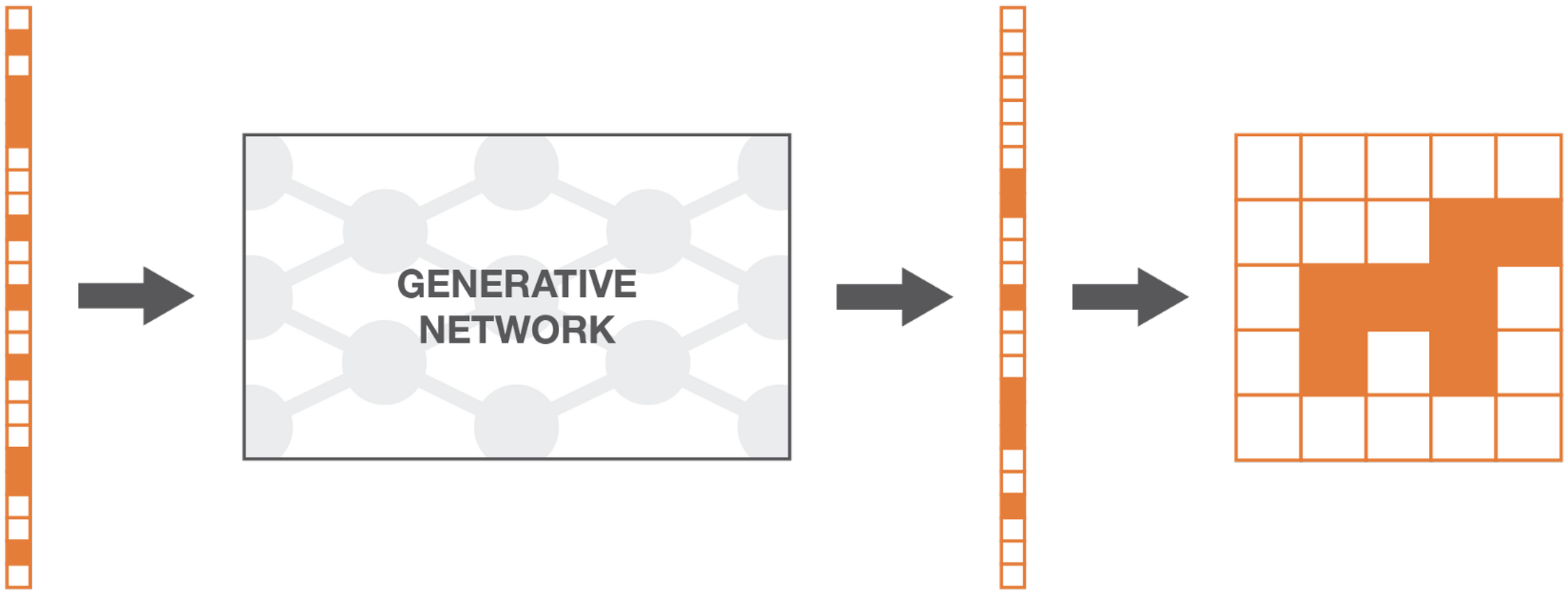


The **inverse transform method** is a way to generate a random variable that follows a given distribution by making a **uniform random variable** go through a **well designed "transform function".**

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Generating random variables



Input random variable (drawn from a simple distribution, for example uniform).

The generative network transforms the simple random variable into a more complex one.

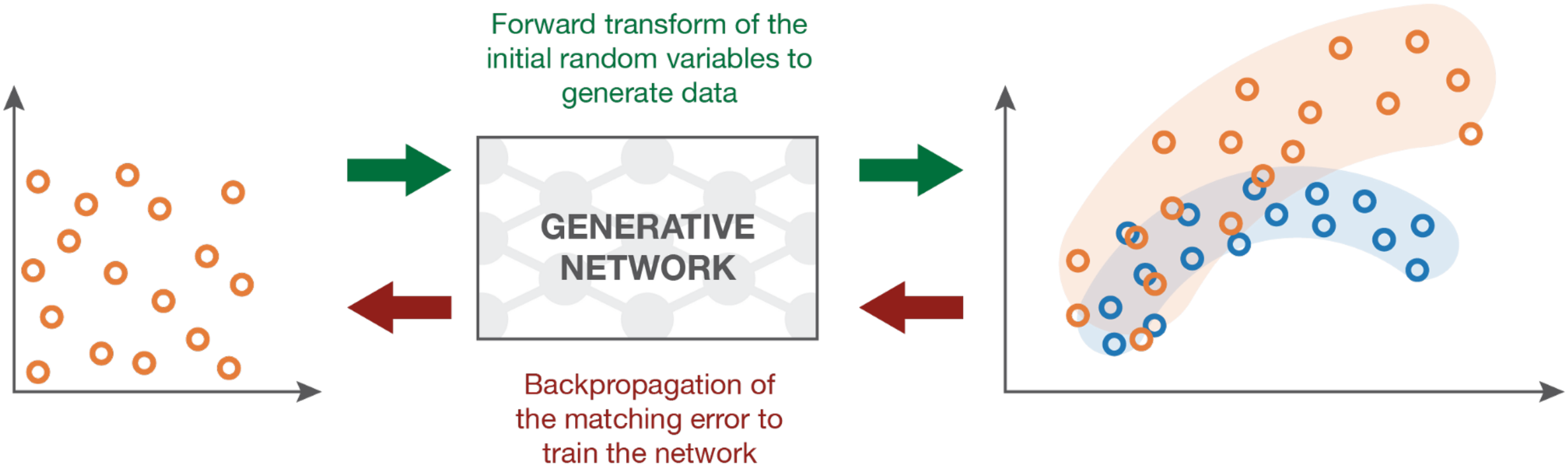
Output random variable (should follow the targeted distribution, after training the generative network).

The output of the generative network once reshaped.

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Comparing the generated distribution to the actual distribution



Input random variables (drawn from a uniform).

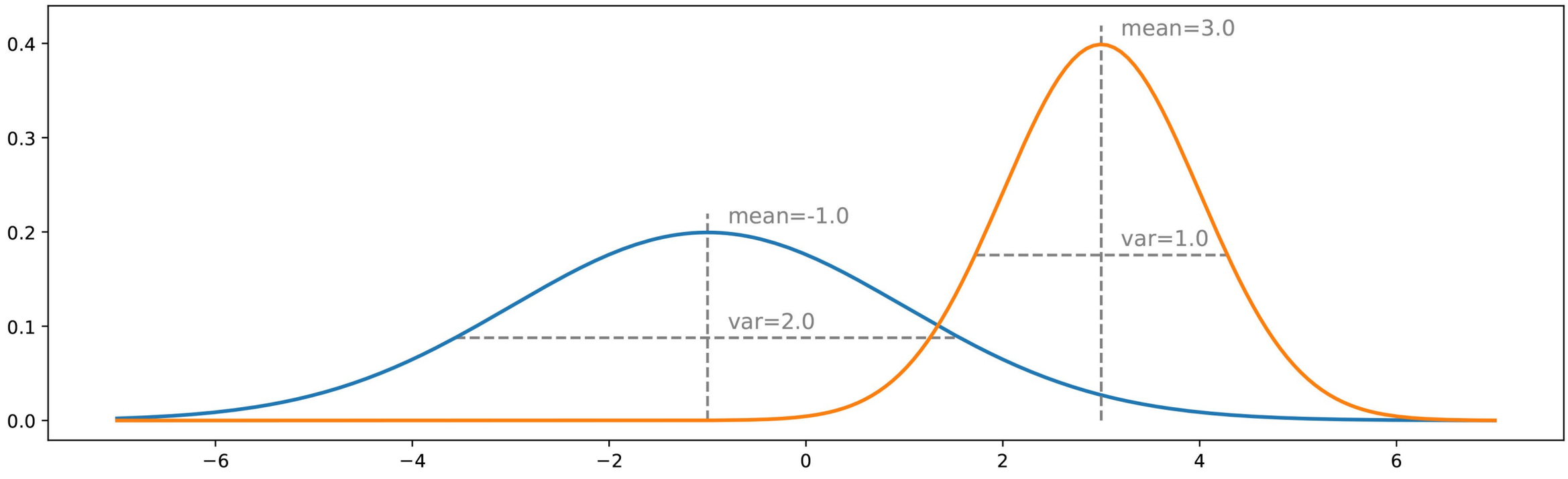
Generative network to be trained.

The generated distribution is compared to the true distribution and the "matching error" is backpropagated to train the network.

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Comparing the generated distribution to the actual distribution

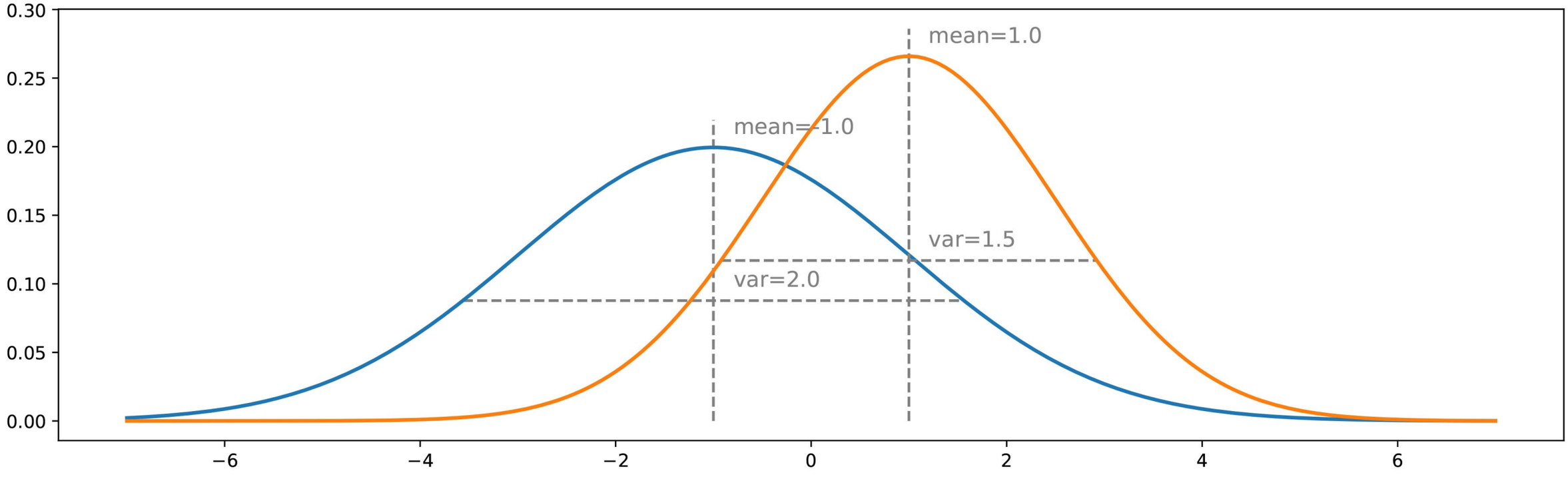


Hypothesis: known distribution

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Comparing the generated distribution to the actual distribution

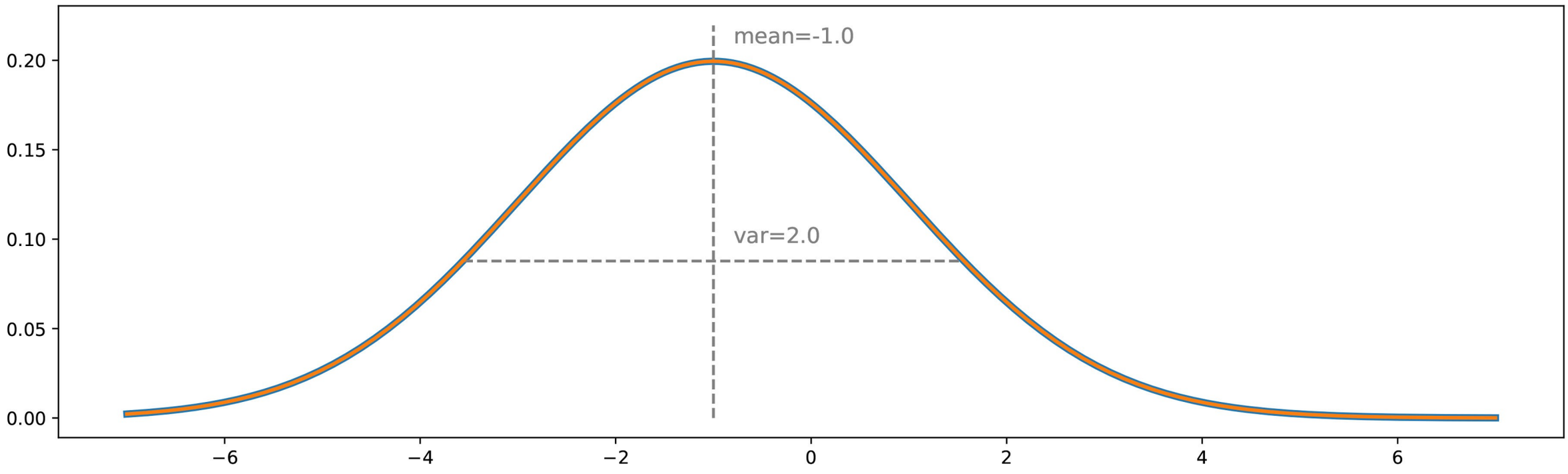


Hypothesis: known distribution

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Comparing the generated distribution to the actual distribution



Hypothesis: known distribution

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Comparing the generated distribution to the actual distribution

Ex: what is the distribution of faces in images?

Count possible images of 32x32x32 ?

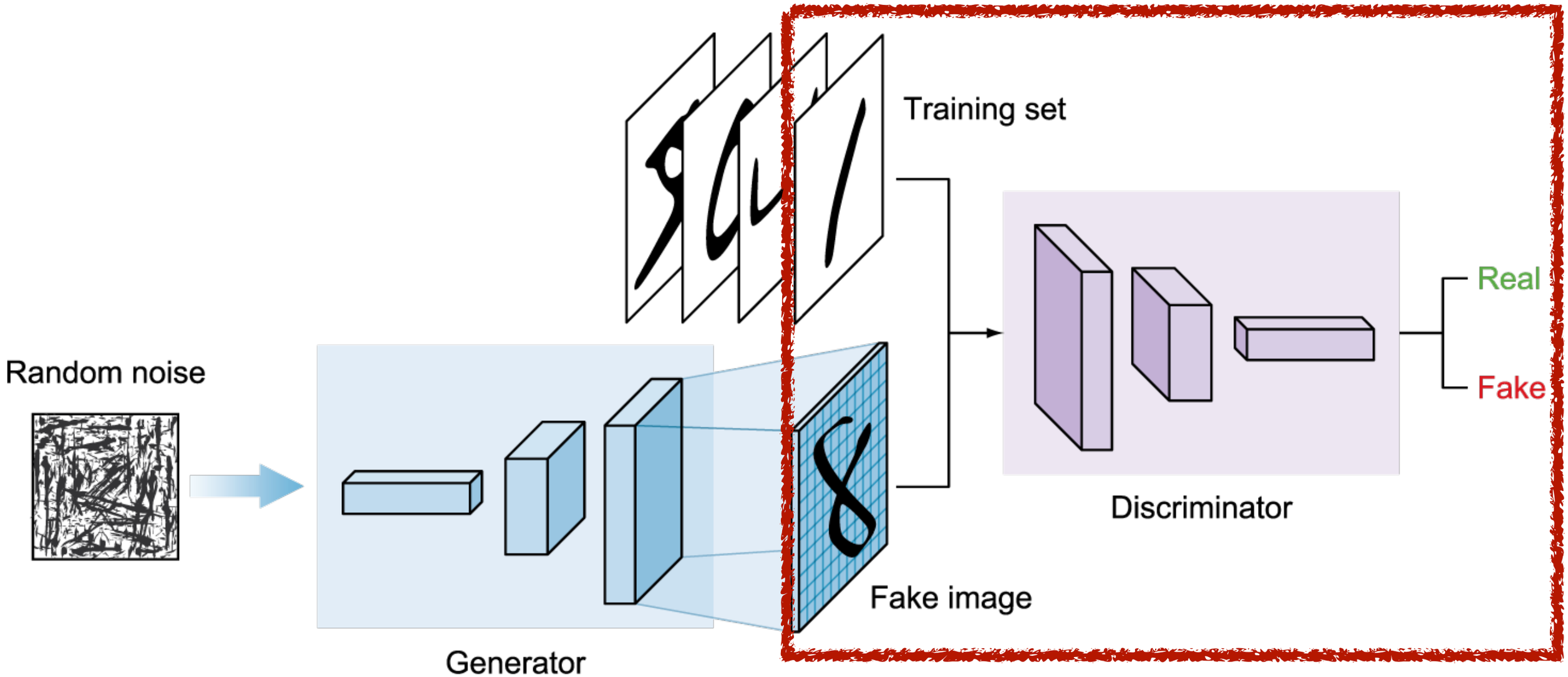
$$18 \cdot 10^{1540}$$

How many are faces ?

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Learning the actual distribution



https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

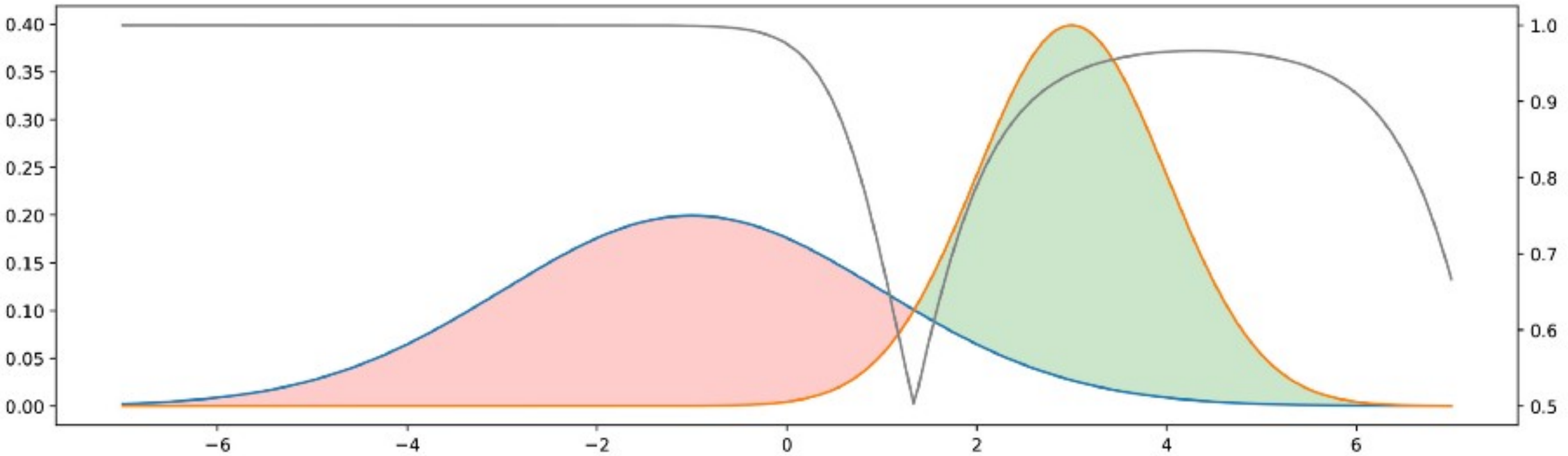
Goal: fool the discriminator

Hypothesis: we have an oracle that can discriminate

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Goal: fool the discriminator



Blue: true distribution

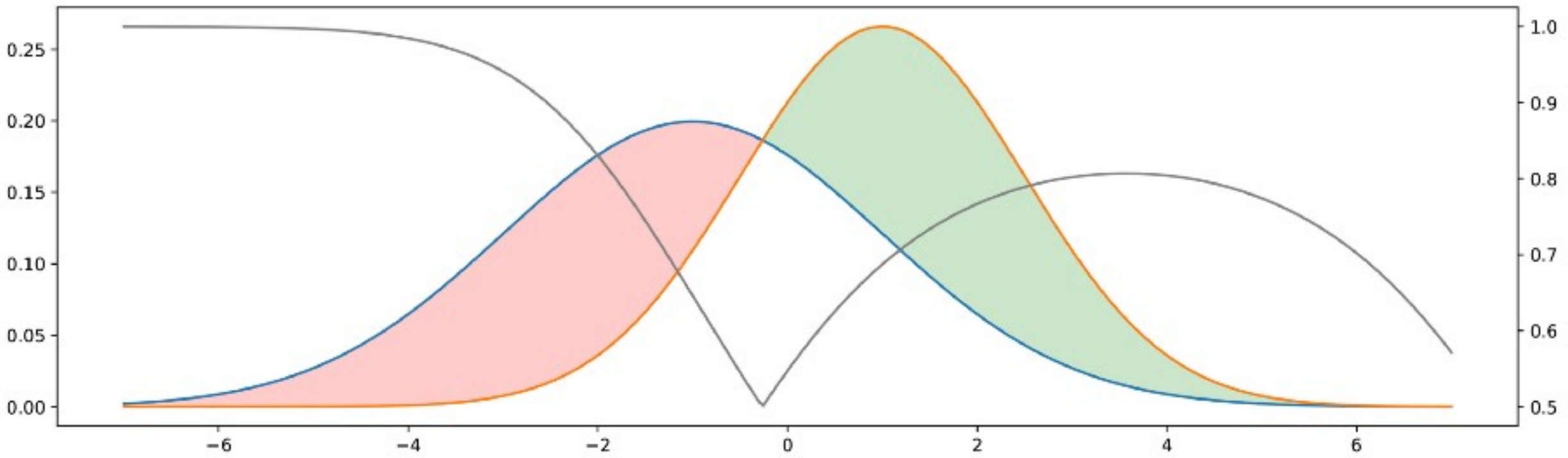
Orange: generated distribution

Gray: probability of discriminator to be true, if it choses the class with higher density in each point (axis on the right!)

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Goal: fool the discriminator



Blue: true distribution

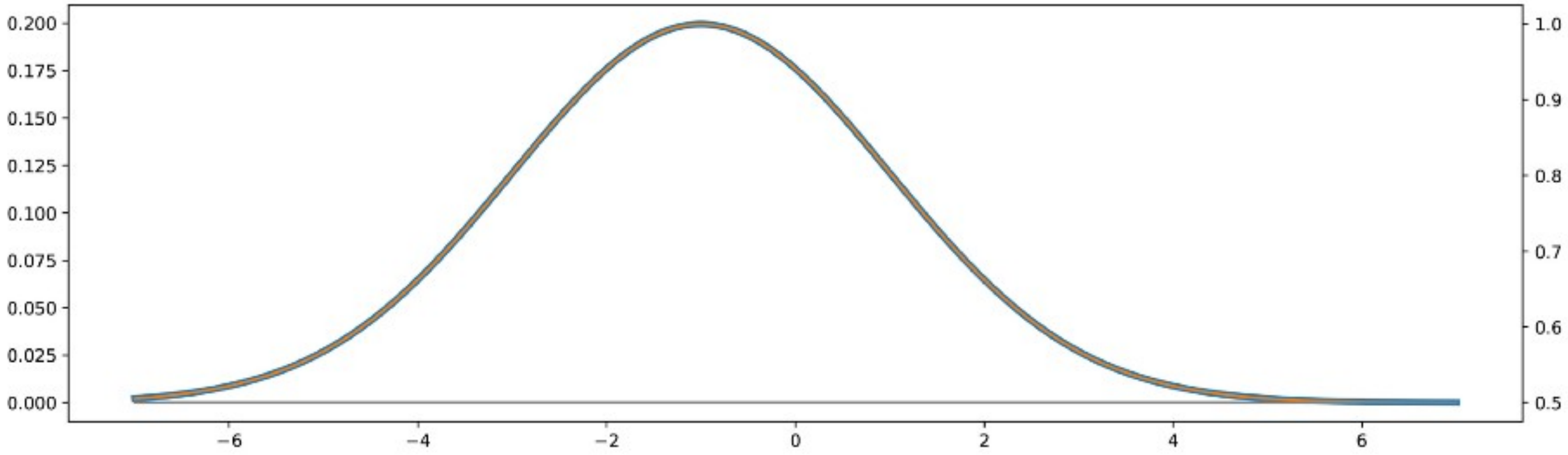
Orange: generated distribution

Gray: probability of discriminator to be true, if it choses the class with higher density in each point (axis on the right!)

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Goal: fool the discriminator



Blue: true distribution

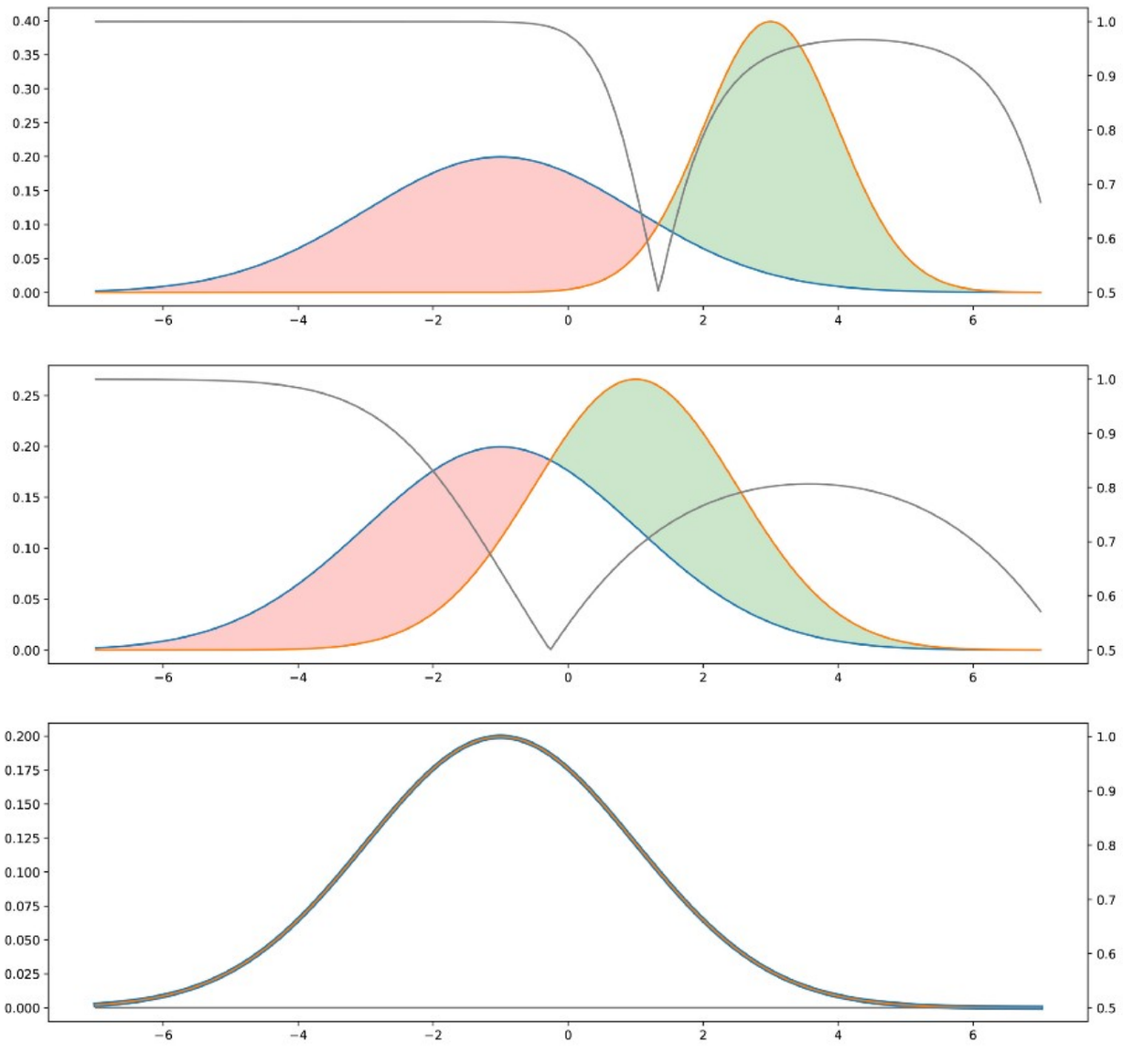
Orange: generated distribution

Gray: probability of discriminator to be true, if it choses the class with higher density in each point (axis on the right!)

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Goal: fool the discriminator



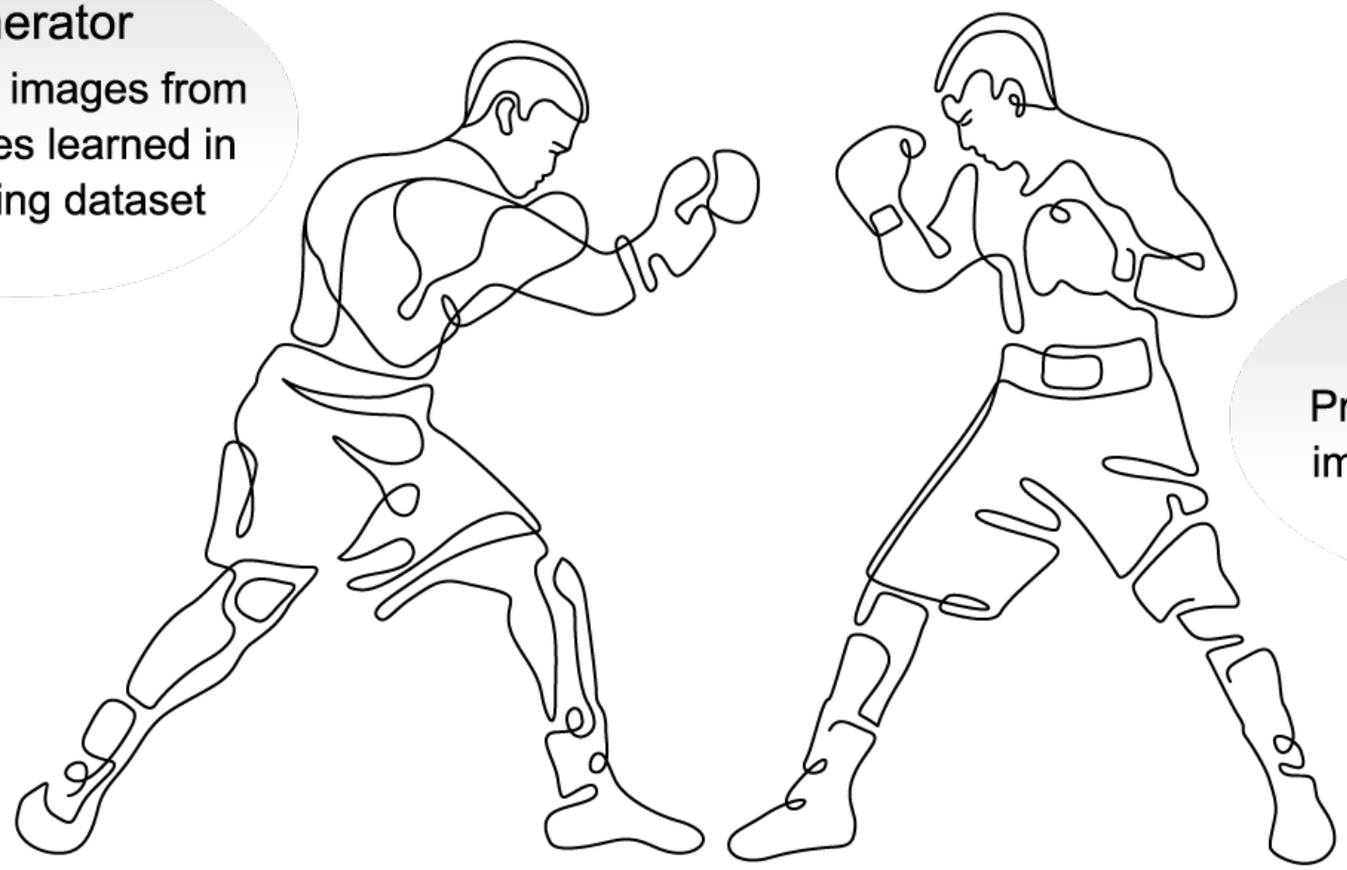
https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Goal: learn the generator and the oracle

Generator

Generates images from the features learned in the training dataset



Discriminator

Predicts whether the image is real or fake

Ref to be updated

Generative Adversarial Networks

Goal: learn the generator and the oracle

Generator $G(z) = x_g$
with $z \sim p_z$ (uniform)

Discriminator $D(x) \in \mathbb{B}$ with

$$x = \begin{cases} x_t \sim p_t \\ x_g \sim p_g = G(p_z) \end{cases}$$

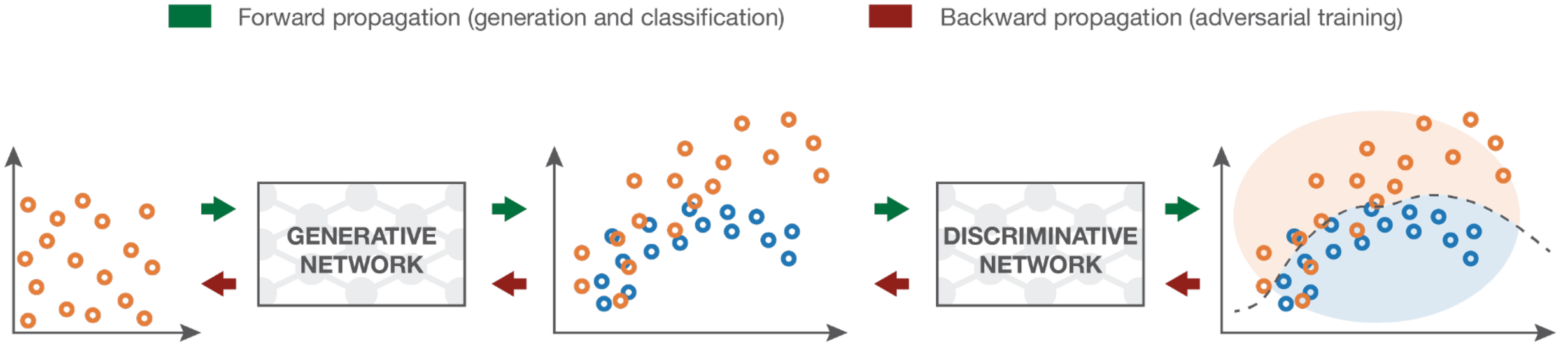
$$\begin{aligned} E(G, D) &= \frac{1}{2}\mathbb{E}_{x \sim p_t}[1 - D(x)] + \frac{1}{2}\mathbb{E}_{z \sim p_z}[D(G(z))] \\ &= \frac{1}{2}\left(\mathbb{E}_{x \sim p_t}[1 - D(x)] + \mathbb{E}_{x \sim p_g}[D(x)]\right) \end{aligned}$$

$$\max_G \left(\min_D E(G, D) \right)$$

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

The game of distribution matching



Input random variables.

The generative network is trained to **maximise** the final classification error.

The generated distribution and the true distribution are not compared directly.

The discriminative network is trained to **minimise** the final classification error.

The classification error is the basis metric for the training of both networks.

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks

Applications

- Generate Examples for Image Datasets
- Generate Photographs of Human Faces
- Generate Realistic Photographs
- Generate Cartoon Characters
- Image-to-Image Translation
- Text-to-Image Translation
- Semantic-Image-to-Photo Translation
- Face Frontal View Generation
- Generate New Human Poses
- Photos to Emojis
- Photograph Editing
- Face Aging
- Photo Blending
- Super Resolution
- Photo Inpainting
- Clothing Translation
- Video Prediction
- 3D Object Generation

https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

Generative Adversarial Networks

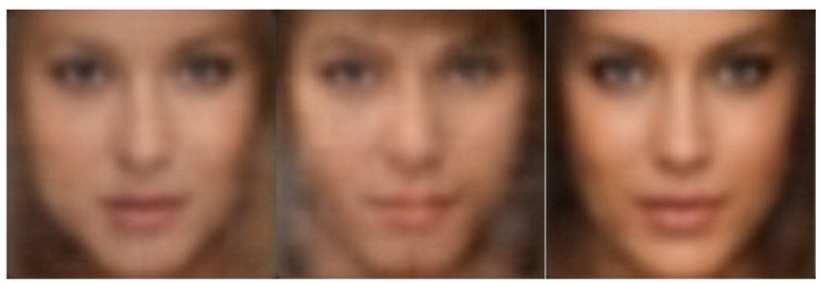
Comparing the generated distribution to the actual distribution

Ex: what is the distribution of faces in images?

Count possible images of 32x32x32 ?

$$18 \cdot 10^{1540}$$

How many are faces ?

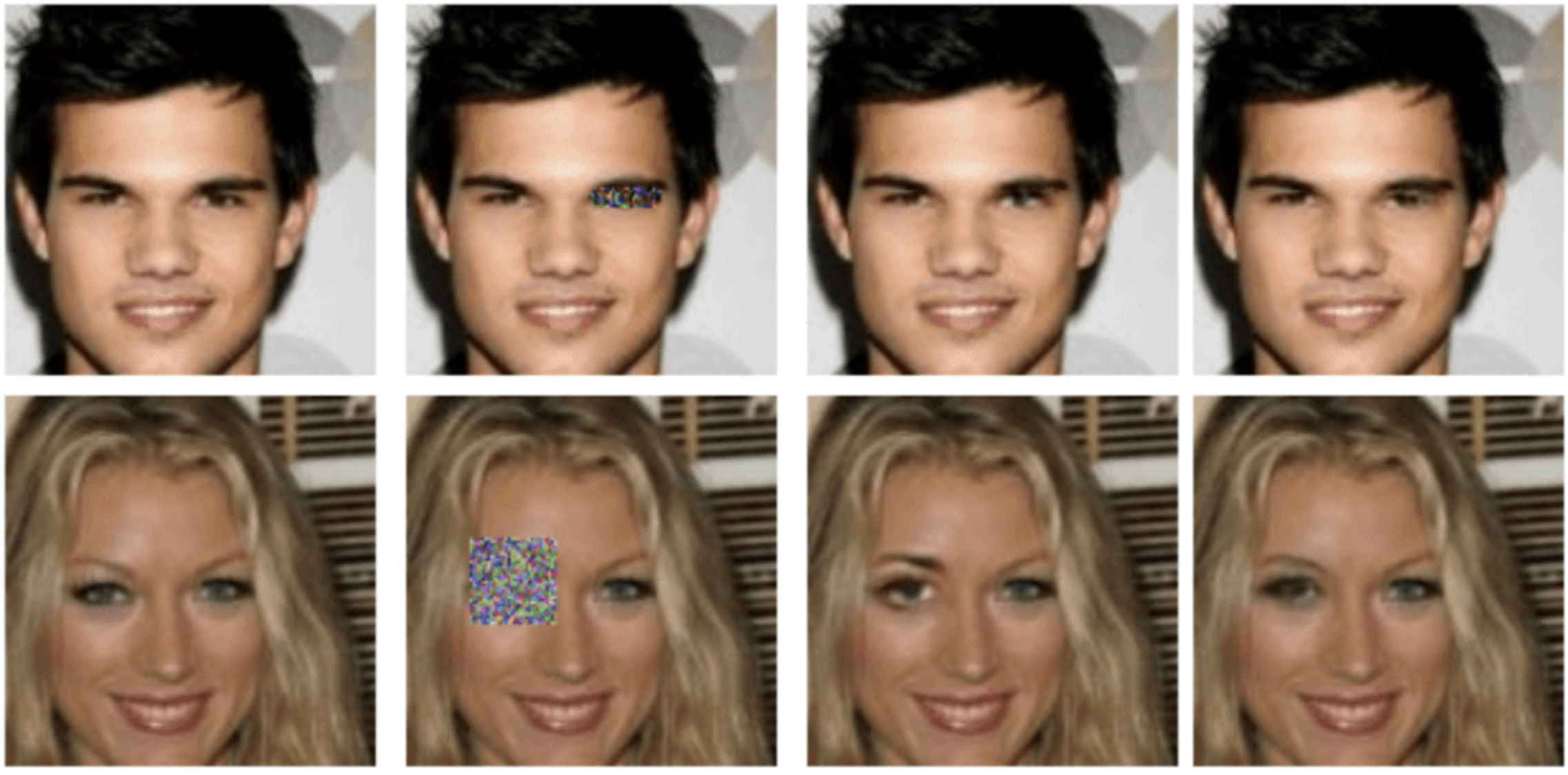


VAE generated

<u>Intuition VAE vs GAN:</u>
The discriminator learns that a **blurry face** is a **fake face.**

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Generative Adversarial Networks



Li, Yijun, Sifei Liu, Jimei Yang, and Ming-Hsuan Yang. "Generative face completion." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3911-3919. 2017.

Recurrent Neural Networks

Motivation:

- for some problems, understanding is build in a sequential way:
  - Examples:
    - Reading text
    - Temporal sequences

- Thoughts have **persistence**

- Previous networks are not able to capitalize previous knowledge:
  - Example: classify an action in a movie at every point

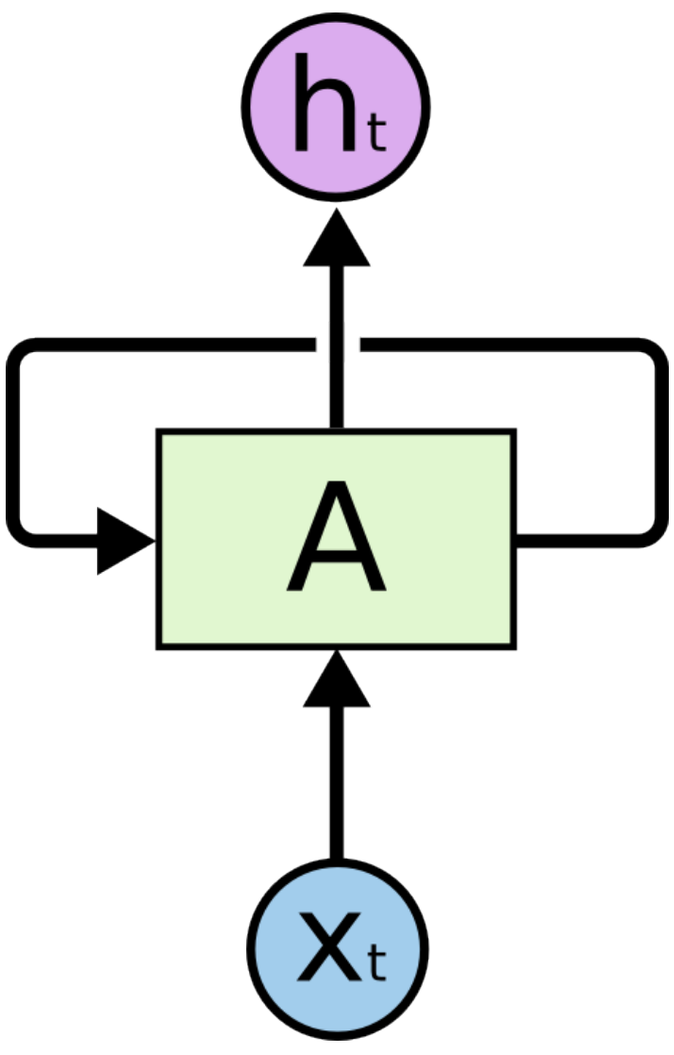
Understanding LSTM Networks

A big thanks to Christopher Olah

- https://colah.github.io/
- https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks

RNNS address this issue with loops



A network

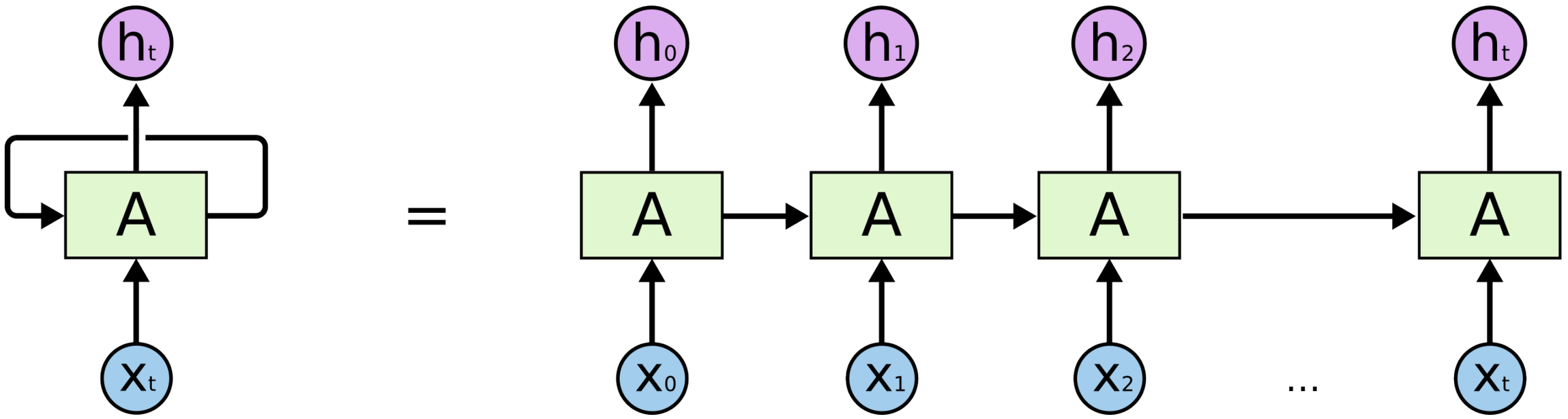
$\mathbf{x_t}$ input

h_t output

loop allows to pass information from one step to the next

Recurrent Neural Networks

RNNS unrolled



A network

$\mathbf{x_t}$ input

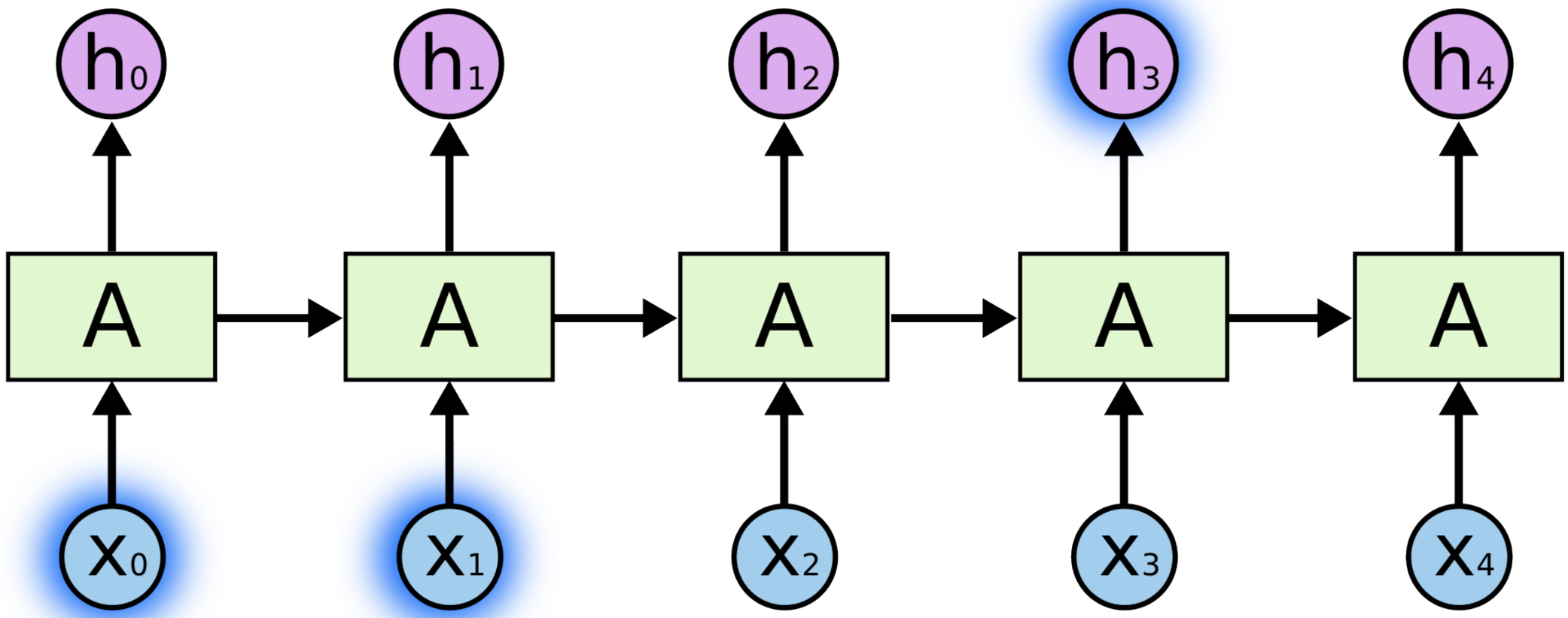
h_t output

RNNs are designed for sequences and lists.

Examples?

Recurrent Neural Networks

Short term dependencies OK - RNNs use past information

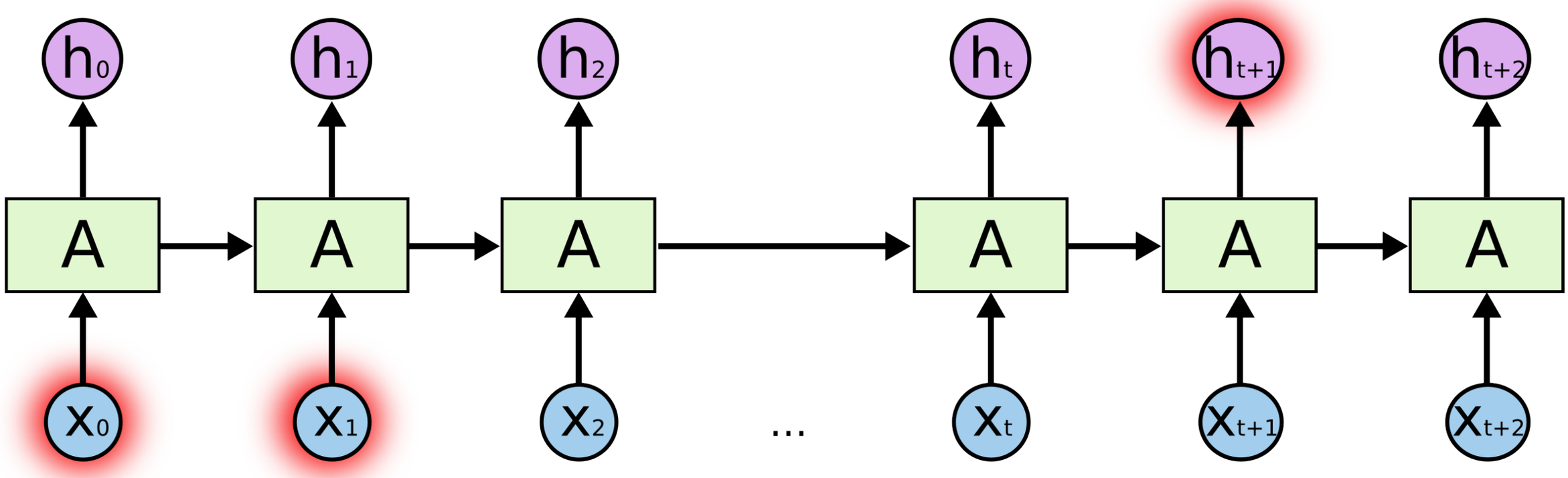


Example: previous video frames inform about present frame.

Recurrent Neural Networks

The problem of long-term dependencies:

as the gap grows, RNNs become unable to connect information



Theoretically they could, but not in practice (vanishing and exploding gradients)

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." *IEEE transactions on neural networks* 5, no. 2 (1994): 157-166.

LSTM: Long Short Term Memory

LONG SHORT-TERM MEMORY

Neural Computation 9(8):1735–1780, 1997

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

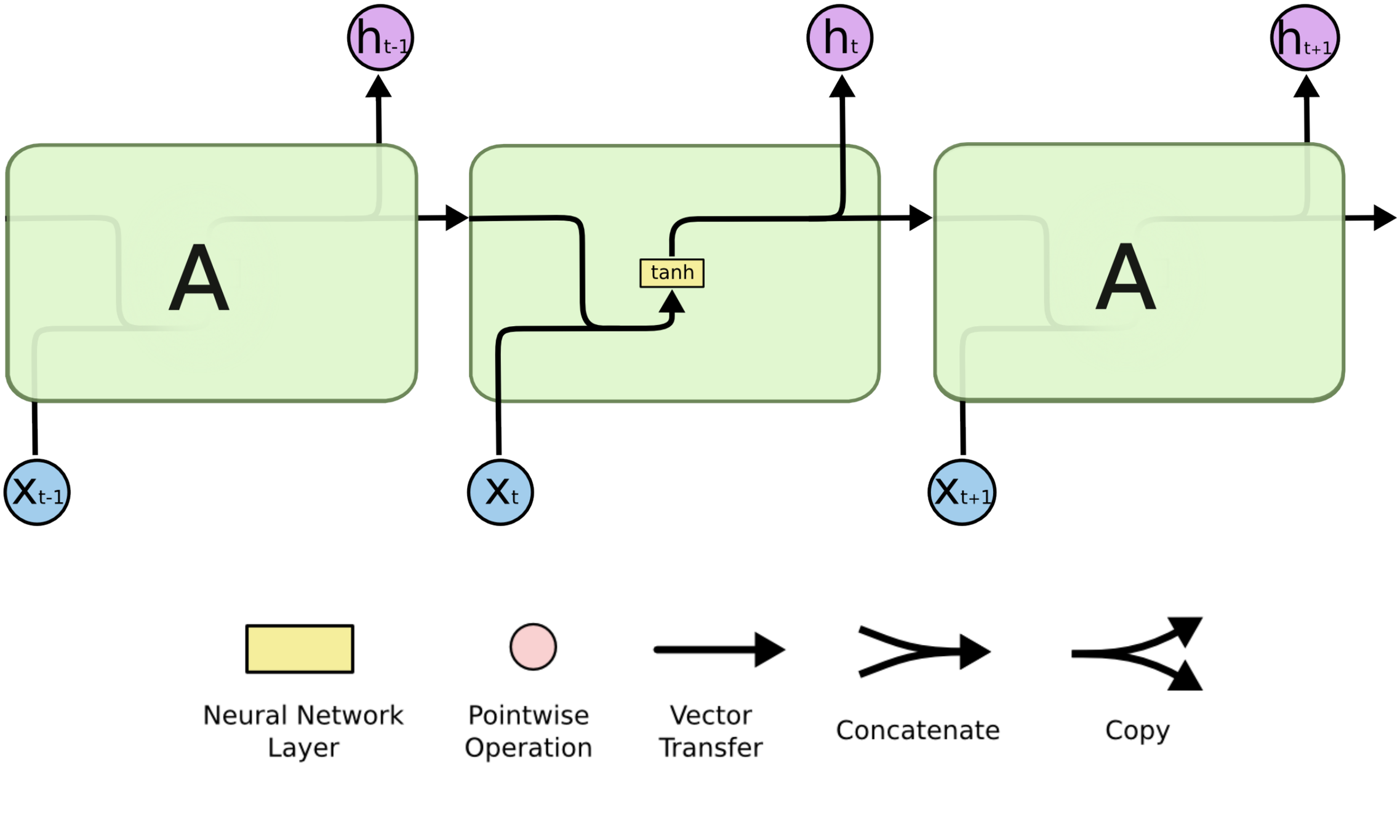
Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780.

Special kind of RNN, capable to learn long term dependencies.
By “default” they remember information for long time!

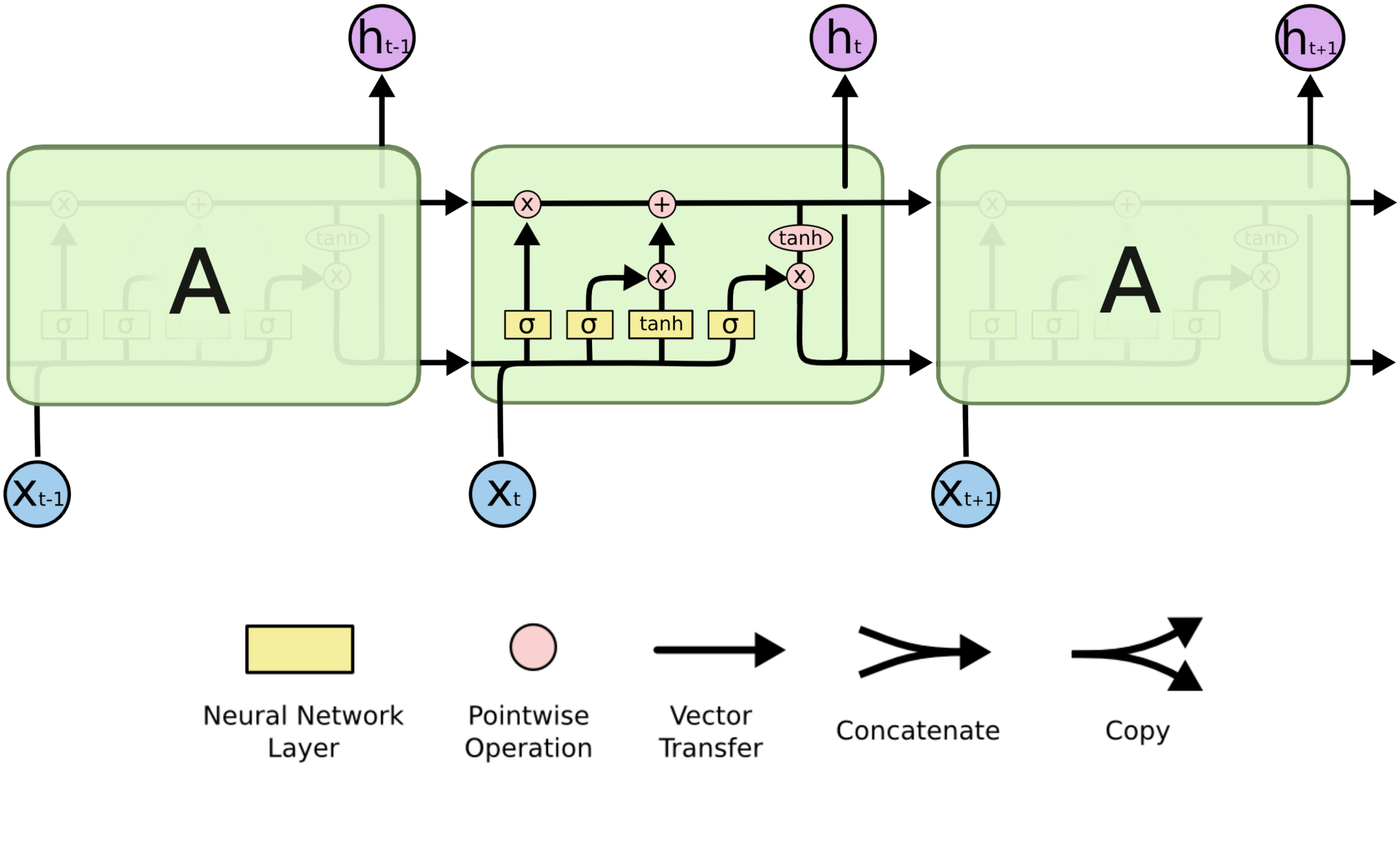
LSTM: Long Short Term Memory

Classical RNN architecture



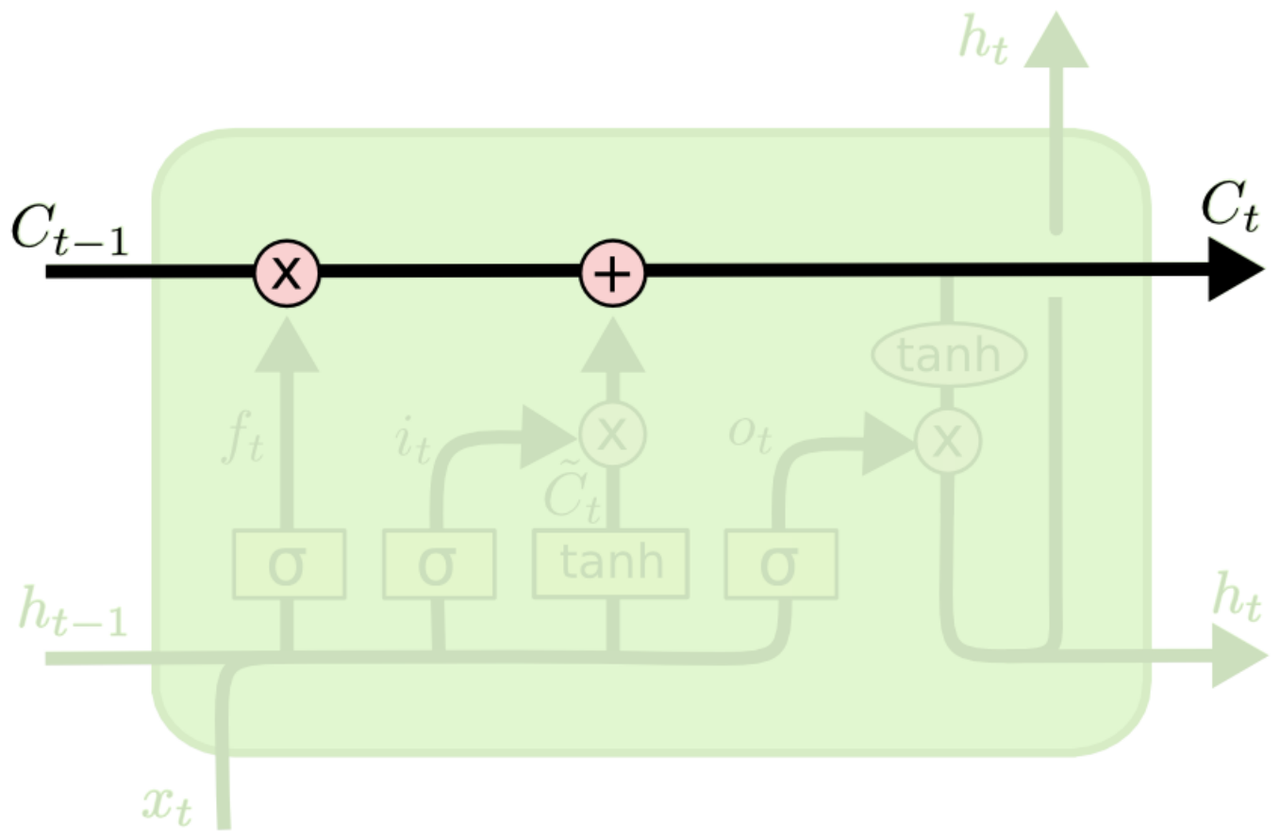
LSTM: Long Short Term Memory

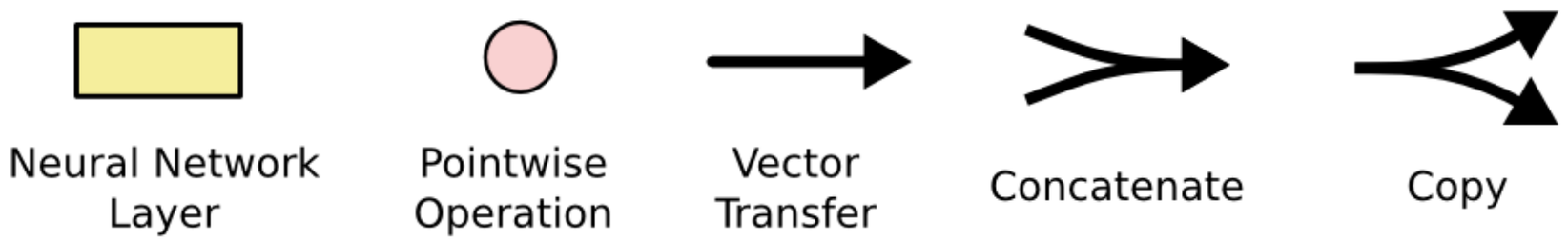
LSTM architecture



LSTM: Long Short Term Memory

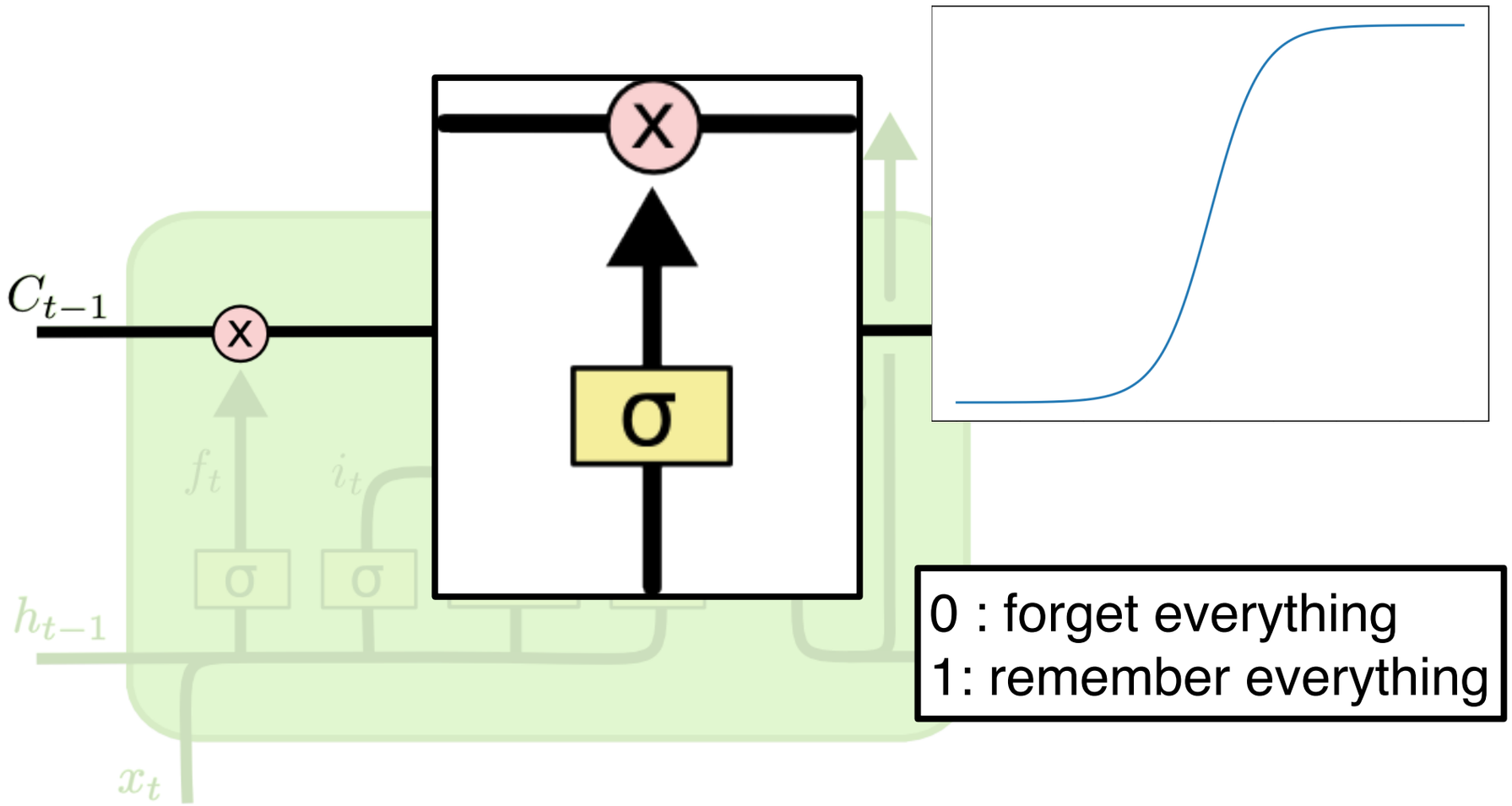
Core Idea

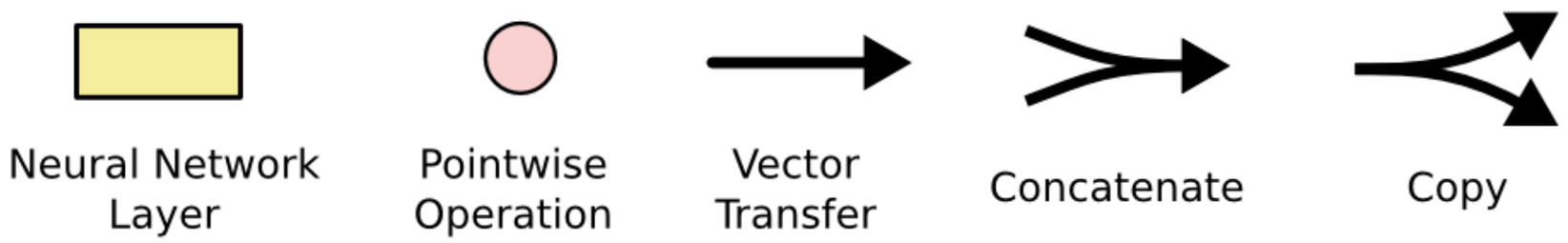




LSTM: Long Short Term Memory

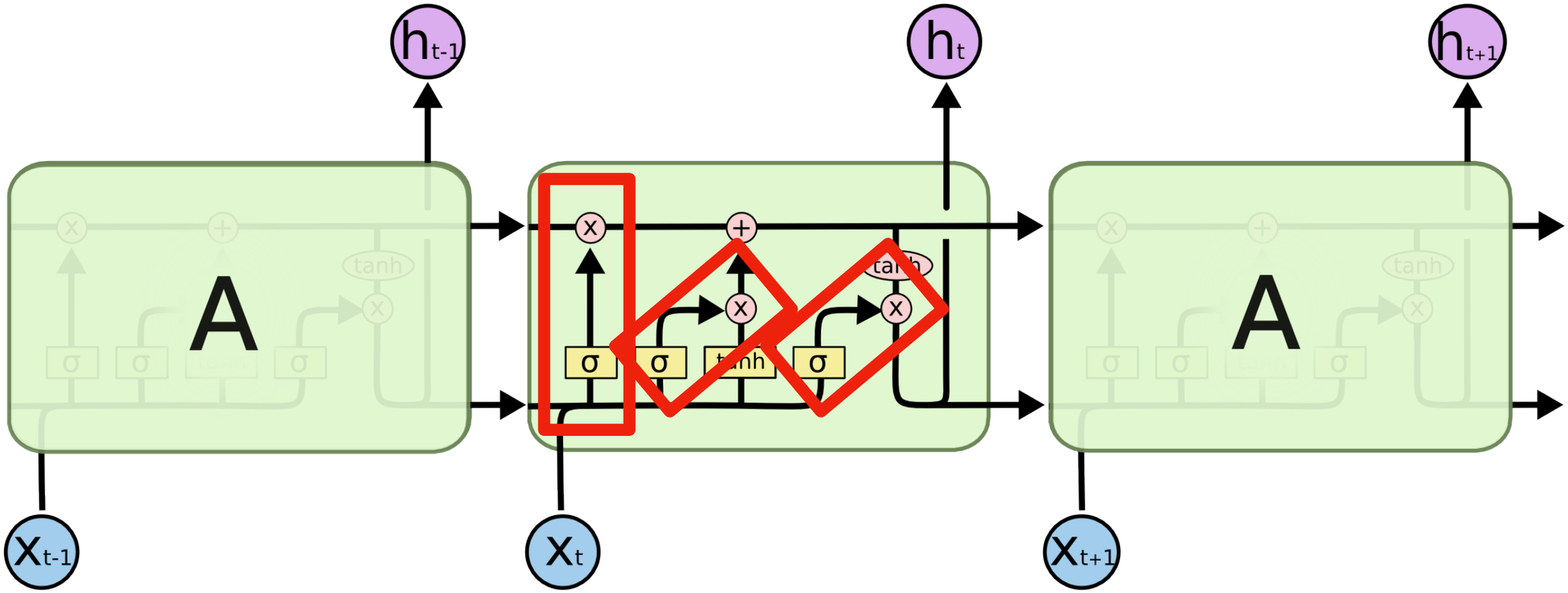
Core Idea: Gate

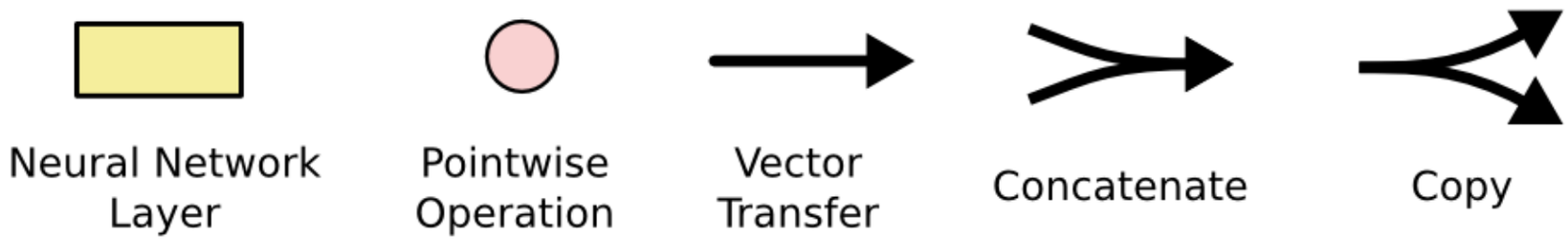




LSTM: Long Short Term Memory

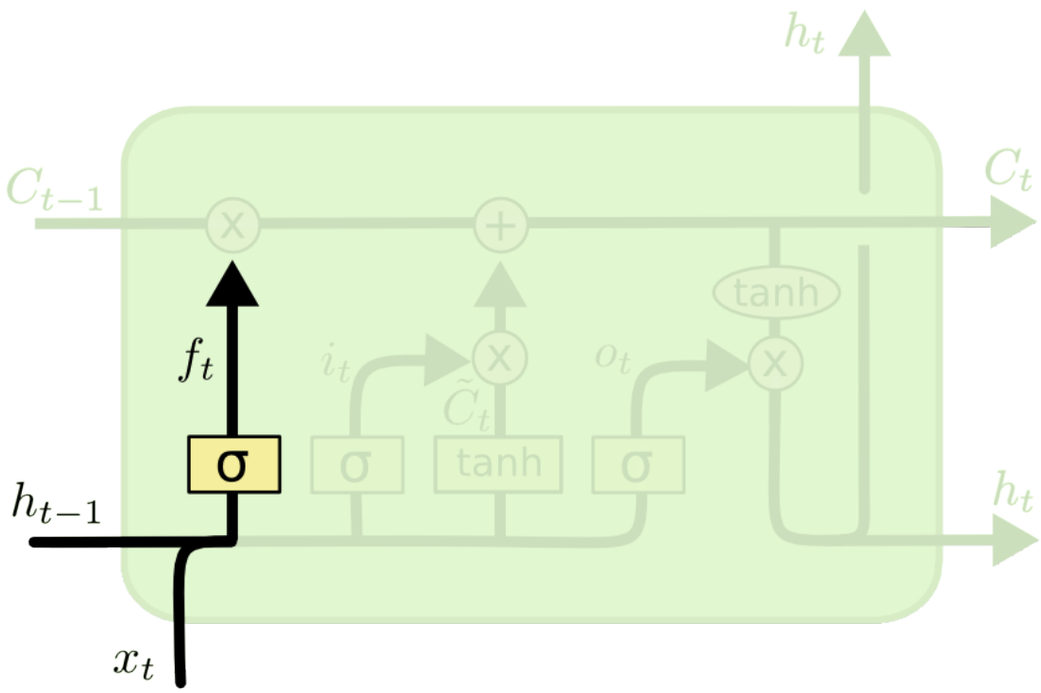
Core Idea: 3 Gates



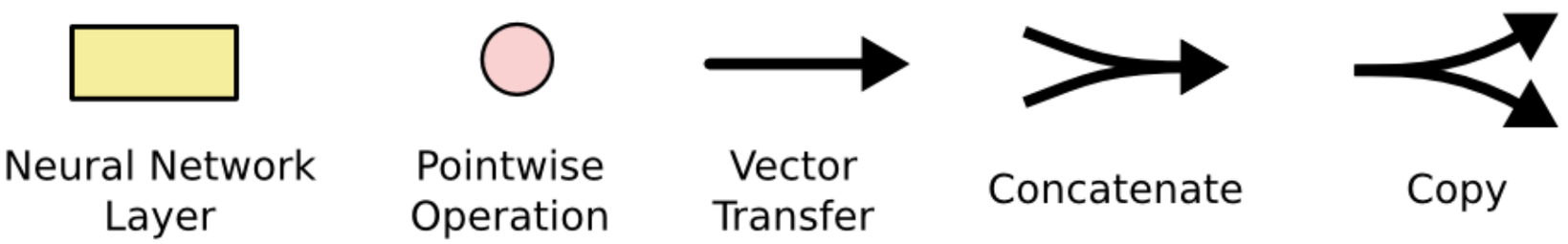


Step-by-step LSTM Walk Through

"Forget gate layer":
Which information are we <u>forgetting</u> from the cell state?



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

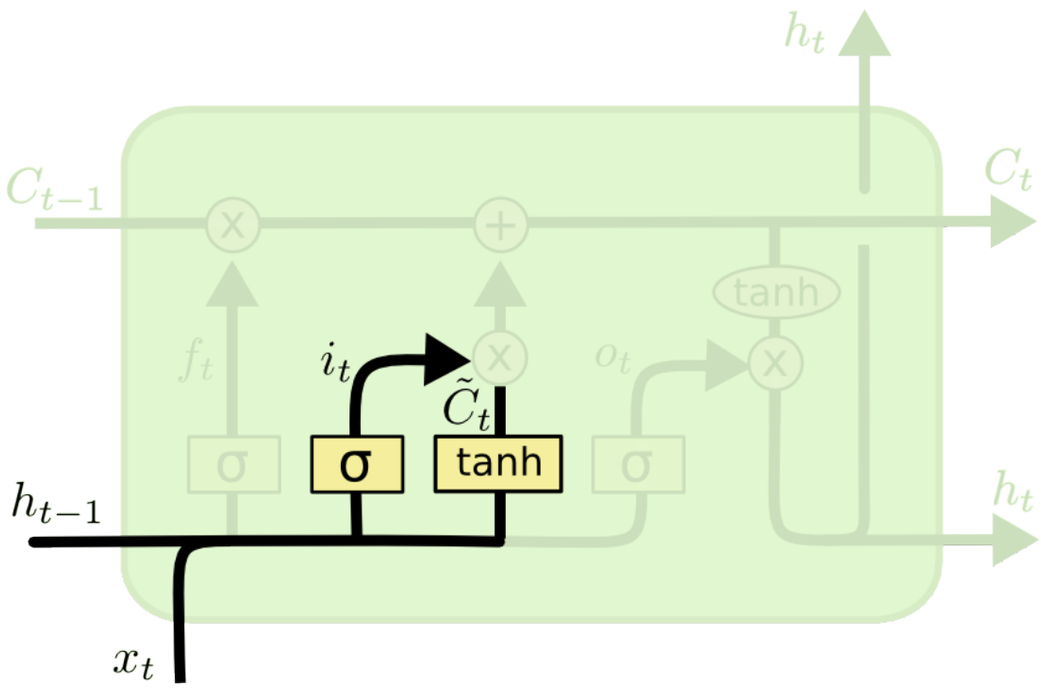


Step-by-step LSTM Walk Through

"Input gate layer":

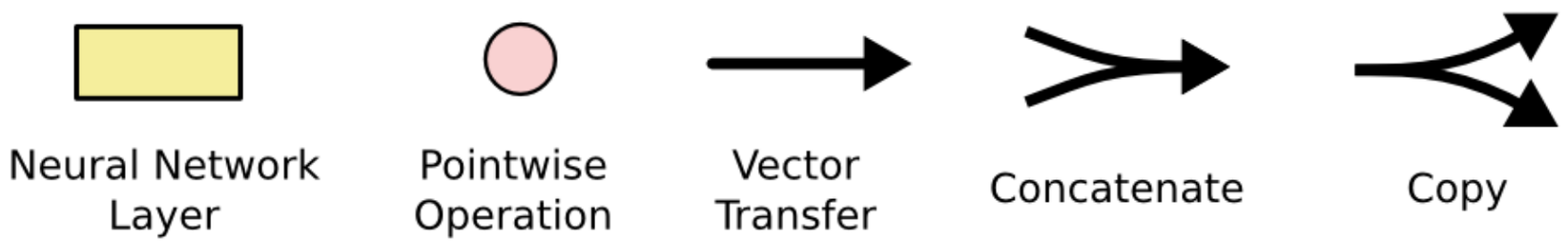
What <u>new information</u> are we storing in the cell state?

Two parts: i_t and $\tilde{C}_t$



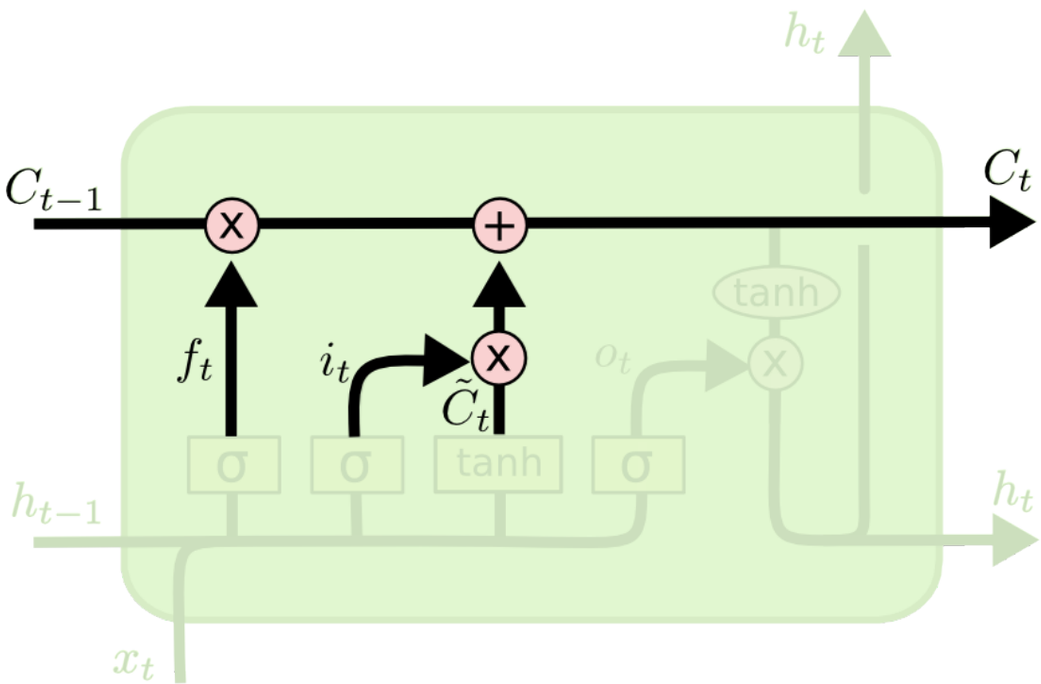
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

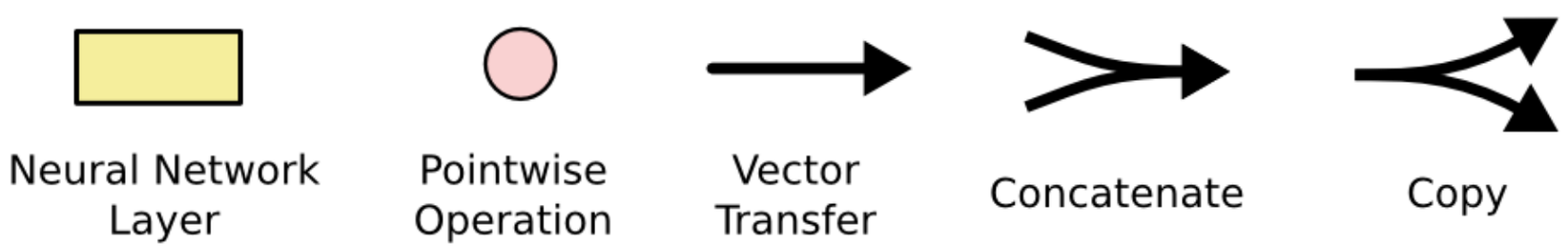


Step-by-step LSTM Walk Through

Update

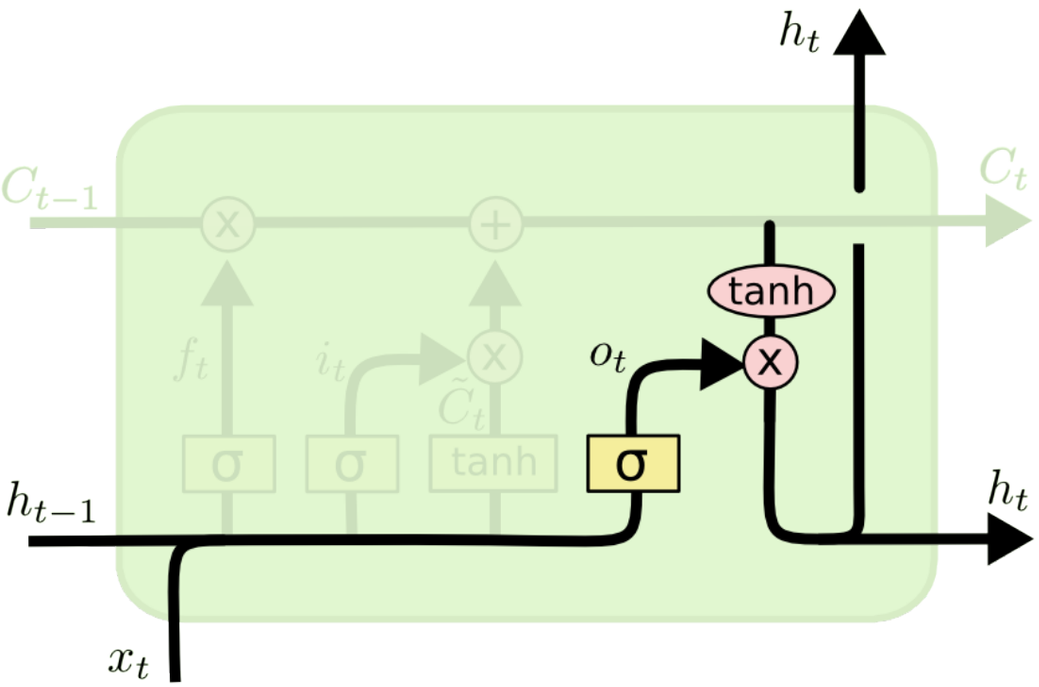


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



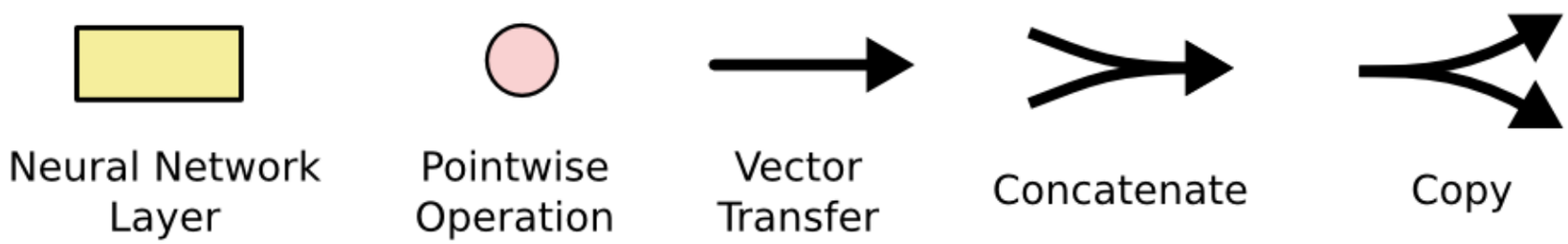
Step-by-step LSTM Walk Through

What about the output ? Output gate



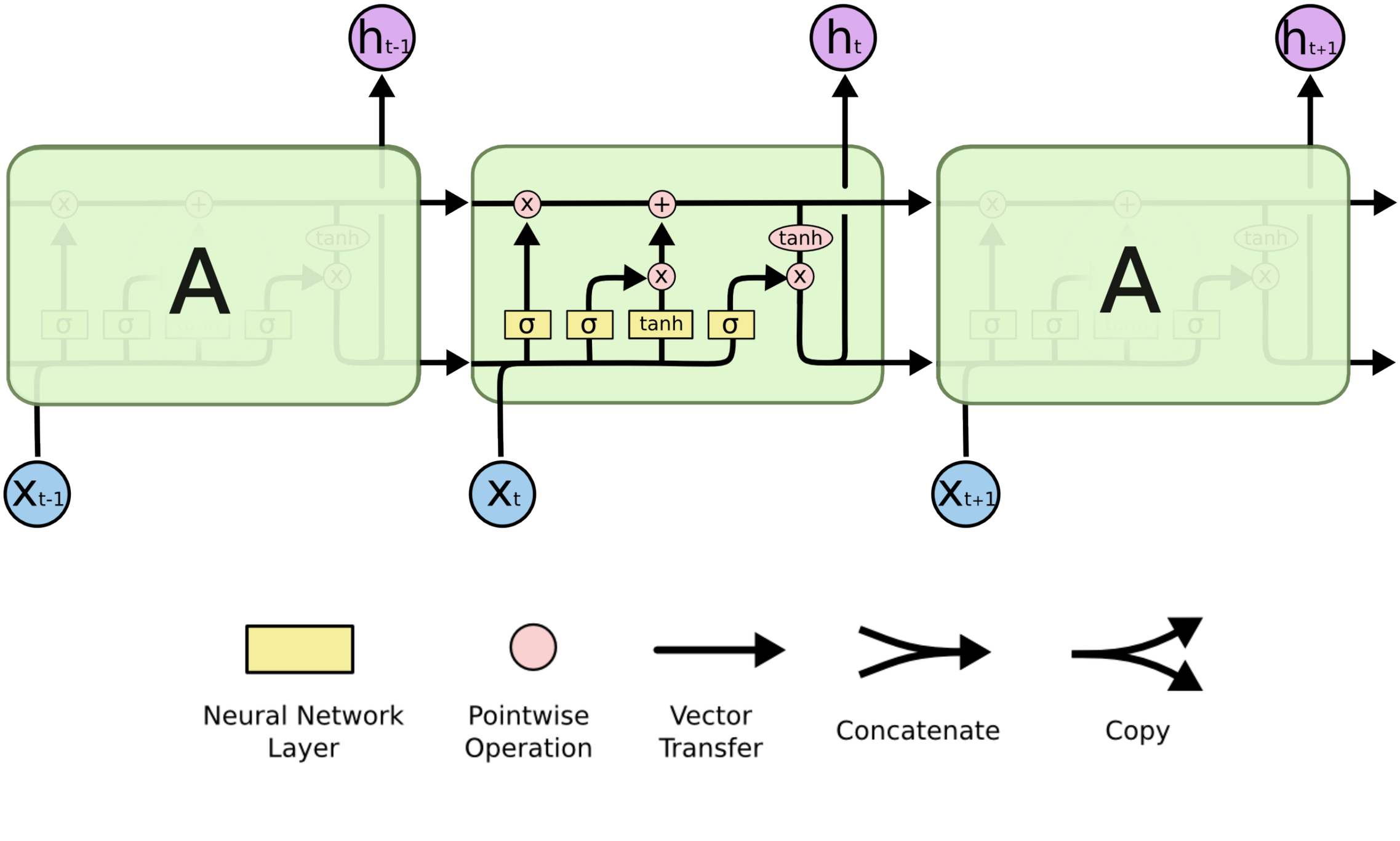
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$



LSTM: Long Short Term Memory

LSTM architecture

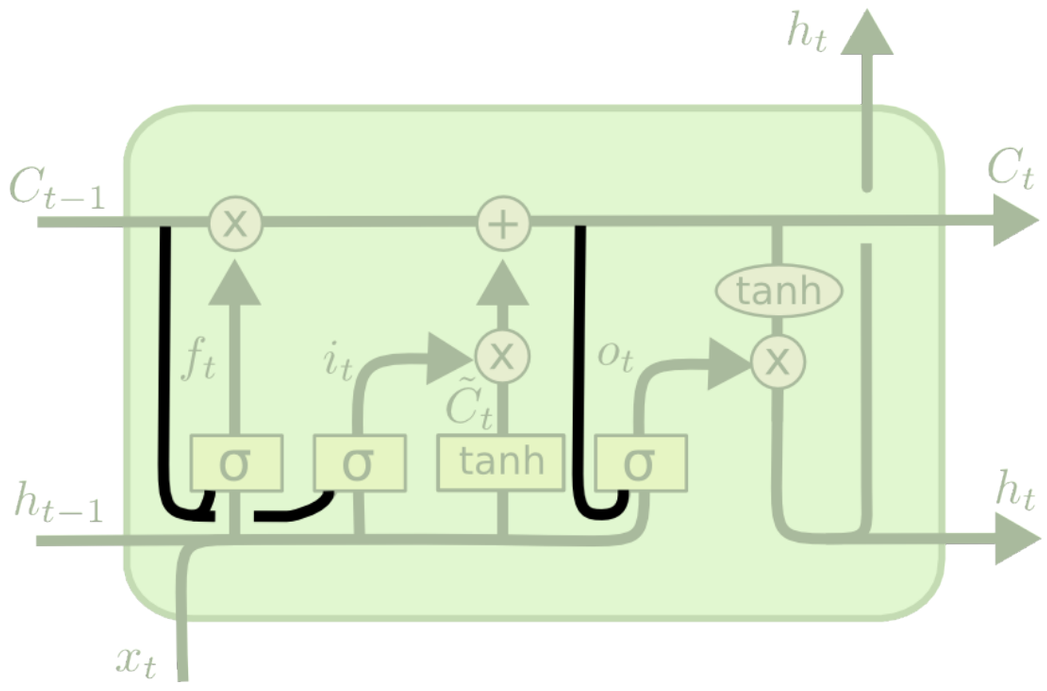


LSTM Variants

Peephole connections:

let gate layers look at the cell state

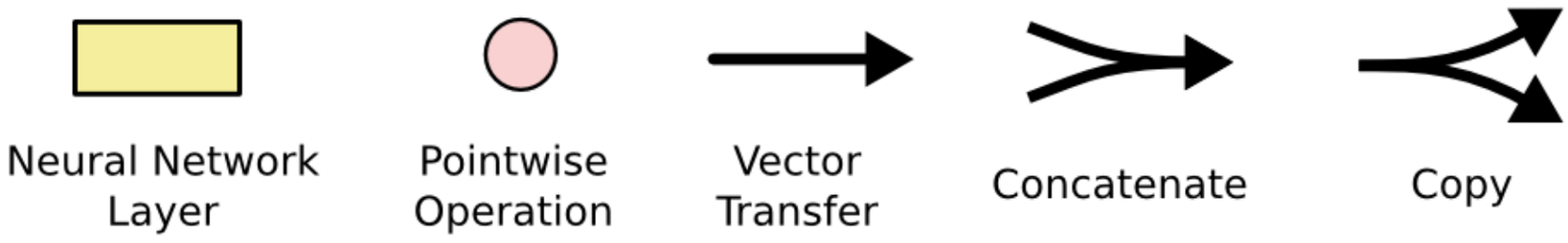
Gers, Felix A., and Jürgen Schmidhuber. "Recurrent nets that time and count." *International Joint Conference on Neural Networks. IJCNN 2000*. vol. 3, pp. 189-194. IEEE, 2000.



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$

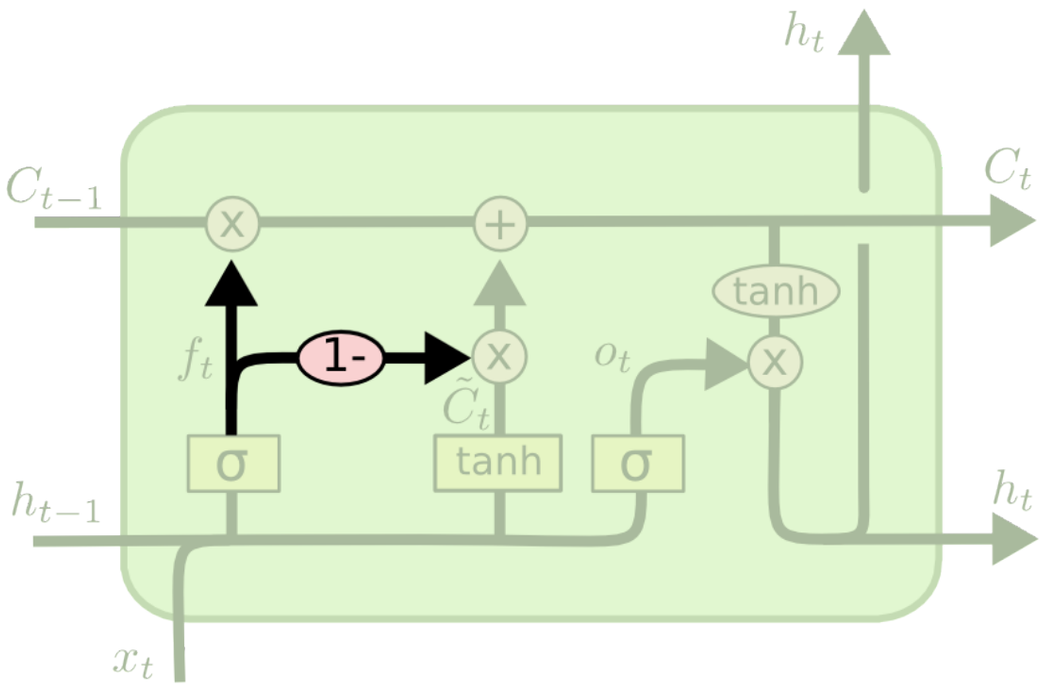
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$

$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

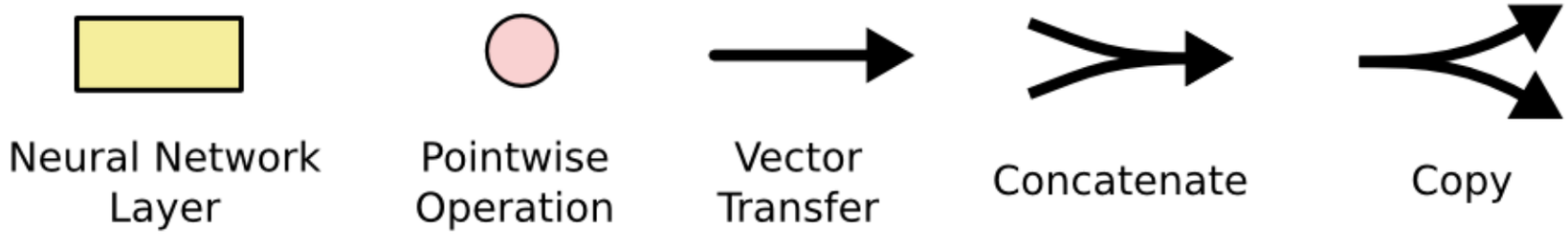


LSTM Variants

Grouping forget and keep gates



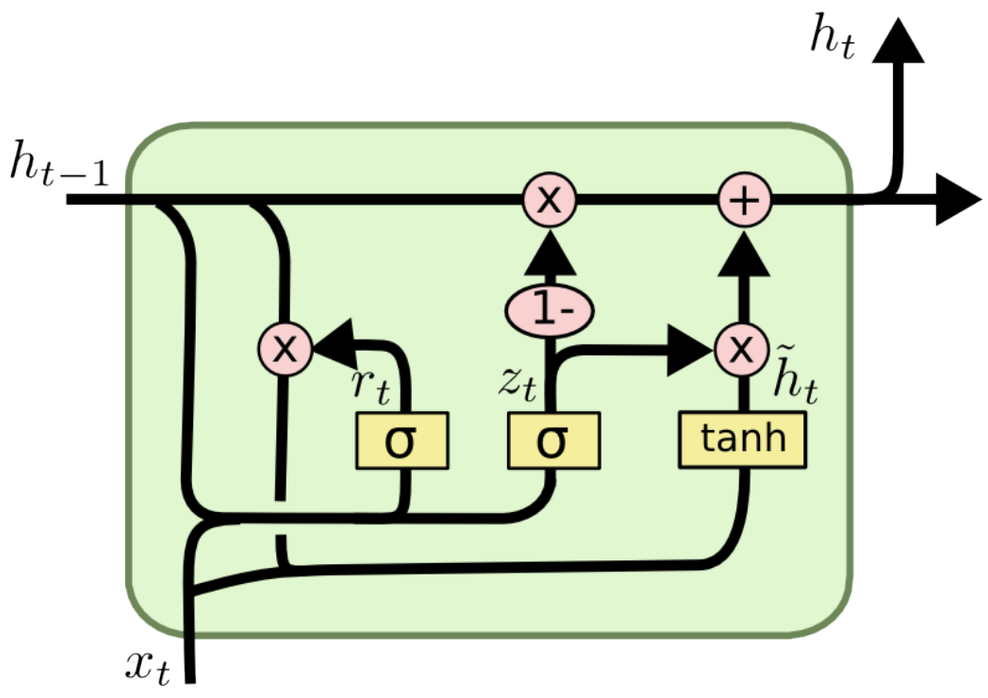
$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$



LSTM Variants

GRU: gated recurrent unit

Cho, Kyunghyun, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the properties of neural machine translation: Encoder-decoder approaches." arXiv preprint arXiv:1409.1259 (2014).

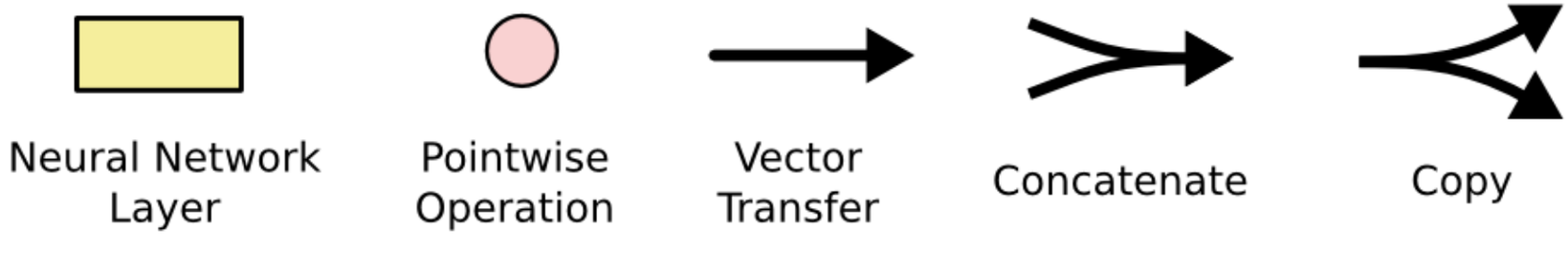


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

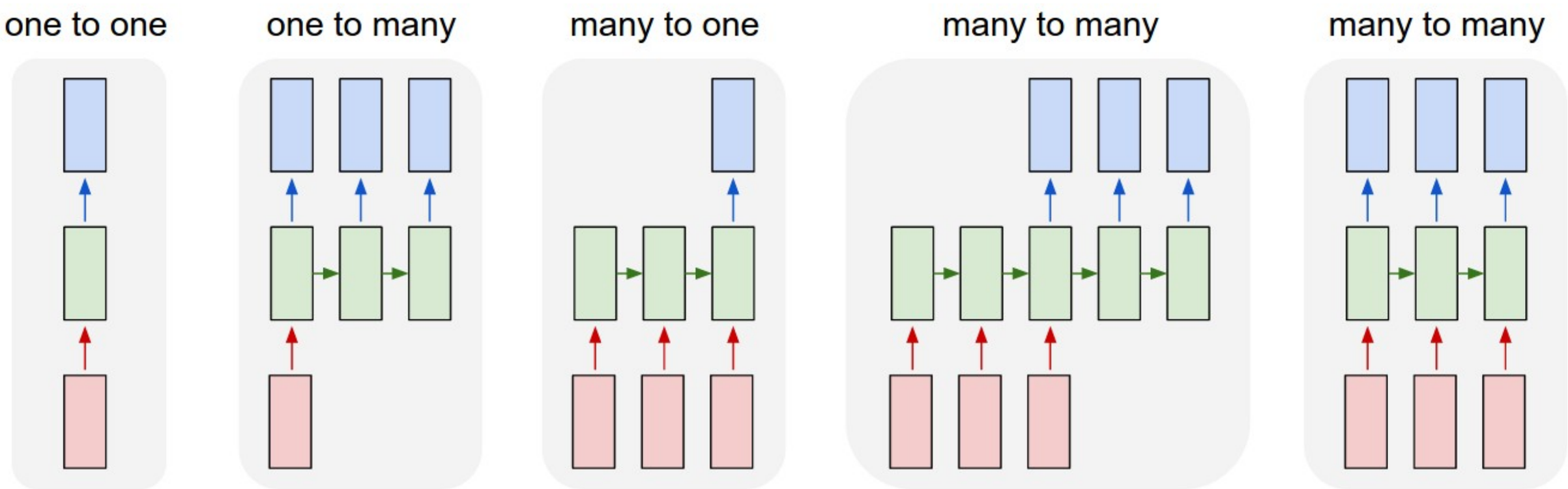
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



Recurrent Neural Networks



RNNs variants. Application examples?

Image from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Big Families of Neural Networks

- Multi Layer Perceptron (MLP)
- Convolutional Neural Networks (CNNs)
- Autoencoders (AEs)
  - Variational Autoencoders (VAEs)
- Generative Adversarial Networks (GANs)
- Recurrent Neural Networks (RNNs)
  - Long Short Temporal Memory (LSTMs)

- Graph Neural Networks (GNNs)

Big Families of Neural Networks

- Multi Layer Perceptron (MLP)
- Convolutional Neural Networks (CNNs)
- Autoencoders (AEs)
  - Variational Autoencoders (VAEs)
- Generative Adversarial Networks (GANs)
- Recurrent Neural Networks (RNNs)
  - Long Short Temporal Memory (LSTMs)

- Graph Neural Networks (GNNs)
- Attention Mechanism - Transformer architecture

Not covered in this unit

- Graph Neural Networks (GNNs)
- Attention Mechanism + Transformer architecture
- Data augmentation - key in unsupervised learning

Summary

- State the problem you want to solve:
  - Classification, regression, segmentation, …
  - State input and output
  - Narrow down the application domain (can be broad)
- Chose a “vanilla” architecture
- Chose the losses
- Evaluate properly (cross validation - test - validation sets)
- Be aware of (and explore) the biases in the dataset
- Push the boundaries

References

- Distill Resources
https://distill.pub/about

- Christopher Olah’s blog
https://colah.github.io/

- Karpathy Blog on RNN effectiveness
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

- Elgendy, Mohamed. Deep Learning for Vision Systems. Simon and Schuster, 2020.