PAT 498/598 (Fall 2024)

Special Topics:
Generative AI for Music and Audio Creation

Lecture 11: Midterm Review

Instructor: Hao-Wen Dong

Review – Background

What is Artificial Intelligence?

AI is the study of how to make computers **do things at which, at the moment, people are better**.

– Elaine Rich and Kevin Knight, 1991

1997 → **2016** → **20??**

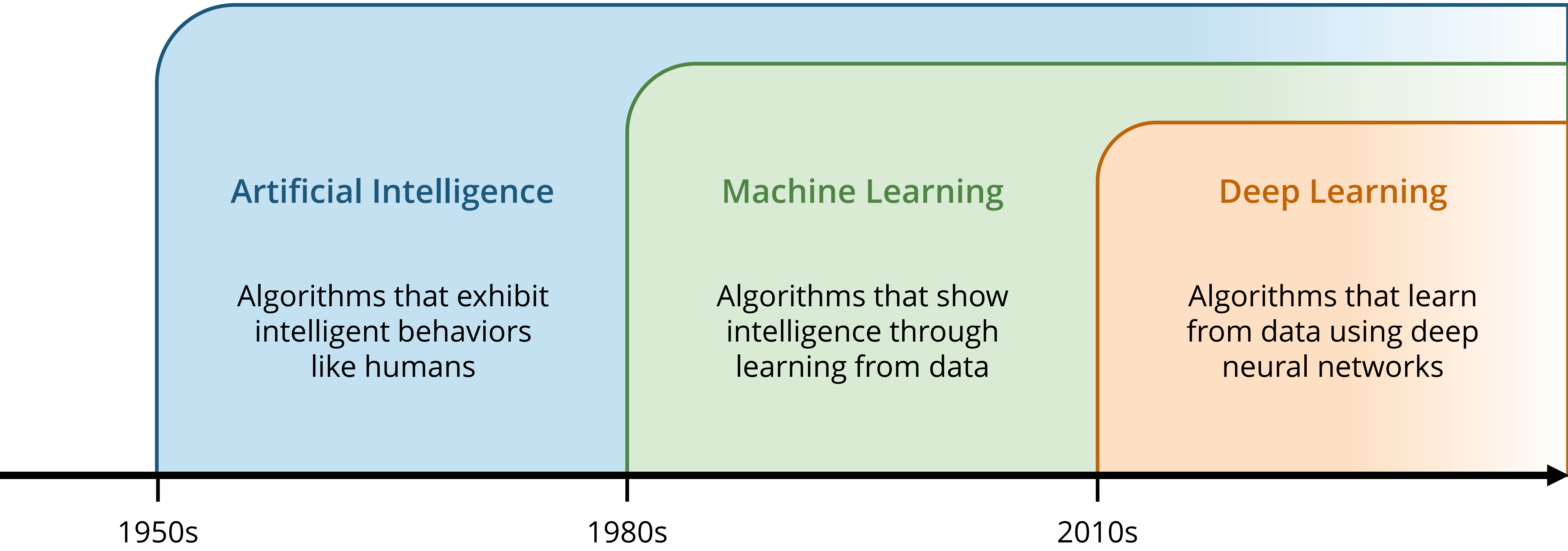
(Source: Britannica)

(Source: The Guardian)

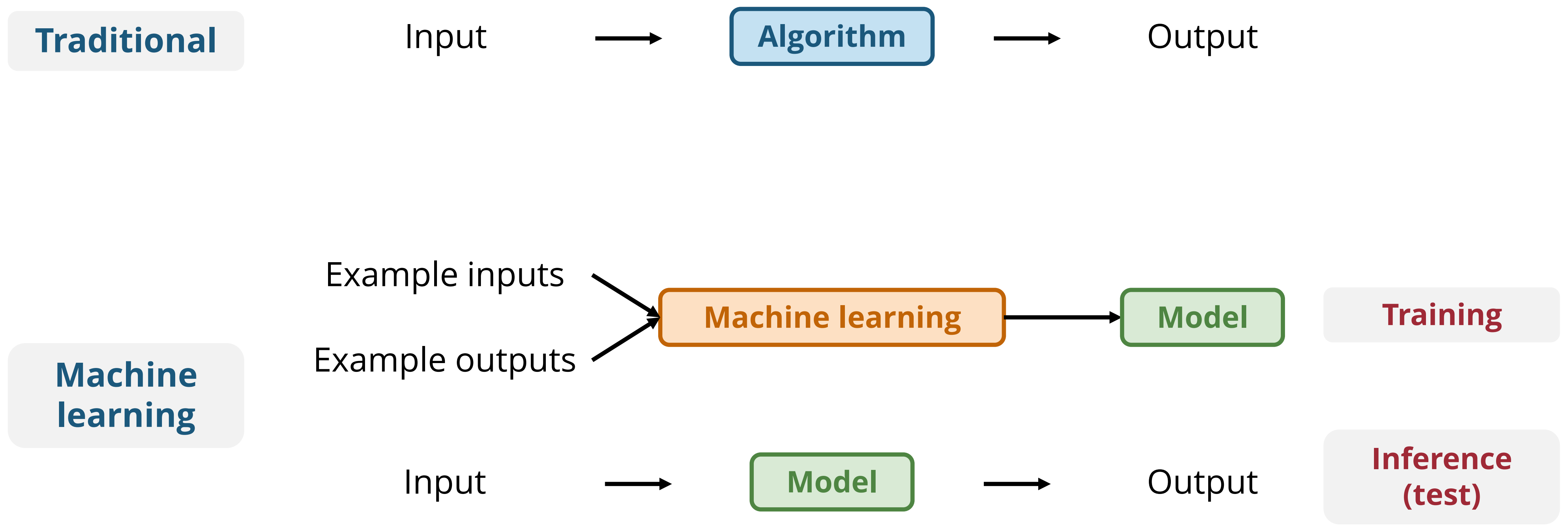
(Source: SC2HL)

Elaine Rich and Kevin Knight, *Artificial Intelligence.* United Kingdom: McGraw-Hill, 1991.
https://www.britannica.com/topic/Deep-Blue
https://www.theguardian.com/technology/2016/mar/15/alphago-what-does-google-advanced-software-go-next
https://www.youtube.com/watch?v=PFMRDm_H9Sg

AI vs ML vs DL


Artificial Intelligence
Algorithms that exhibit intelligent behaviors like humans
Machine Learning
Algorithms that show intelligence through learning from data
Deep Learning
Algorithms that learn from data using deep neural networks
1950s
1980s
2010s

Machine Learning



Components of a Machine Learning Model

Optimization

Defining inputs & outputs

Improve on task T,

with respect to performance metric P,

based on experience E

Loss function
(objective function)

Training data

Deep learning is almost the same as machine learning by this definition!

What's special about deep learning?

Types of Machine Learning

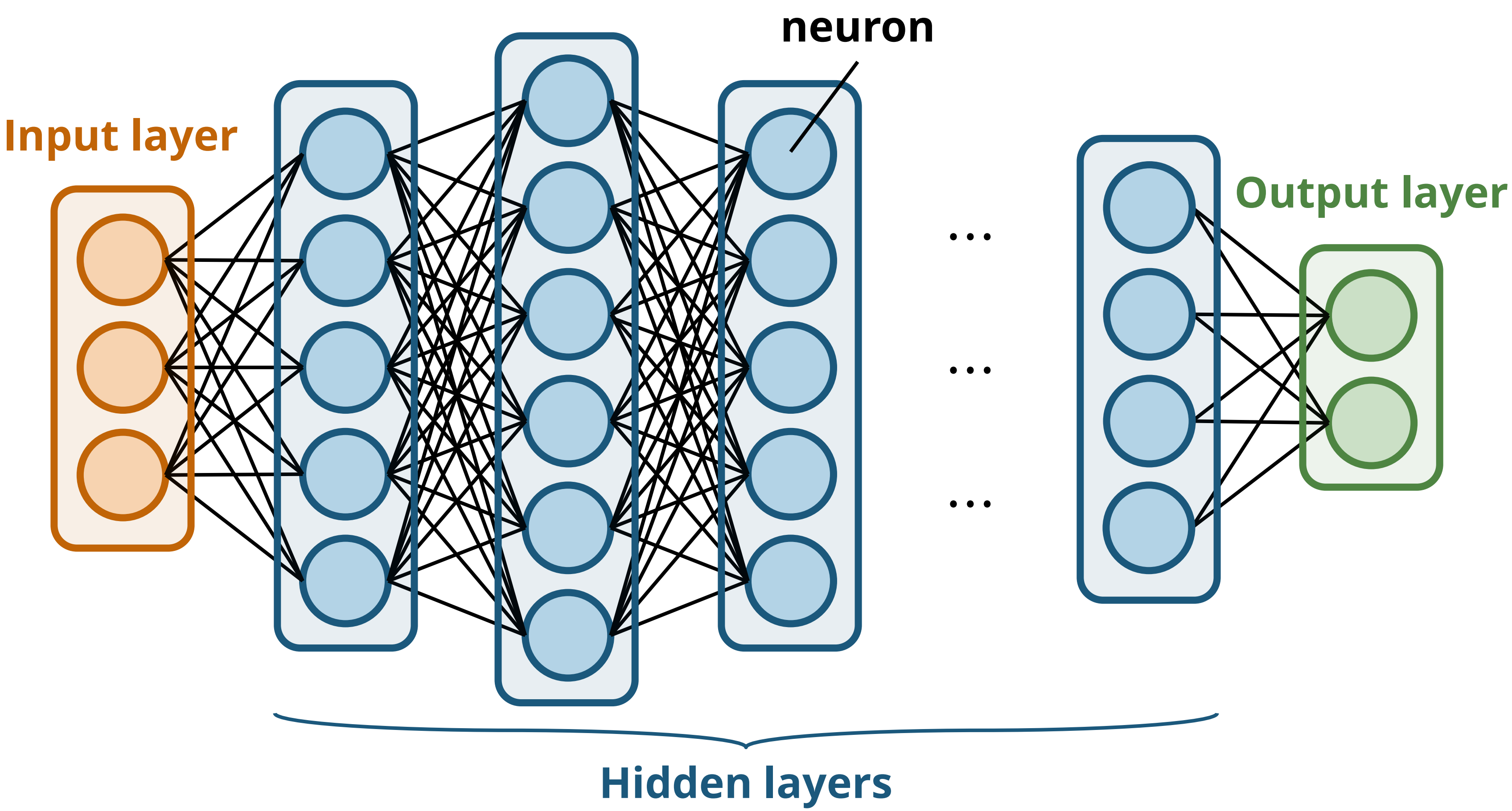
- **Supervised learning** — Given **pairs of example inputs and outputs**
  - **Classification**: *discrete* outputs
  - **Regression**: *continuous* outputs
- **Unsupervised learning** — Given ***only* example inputs**
  - **Self-supervised learning**
- **Semi-supervised learning** — Given **example inputs** and **a few example outputs**
- **Reinforcement learning** — Given **scalar rewards** for **a sequence of actions**

Many generative AI models based on self-supervised learning!

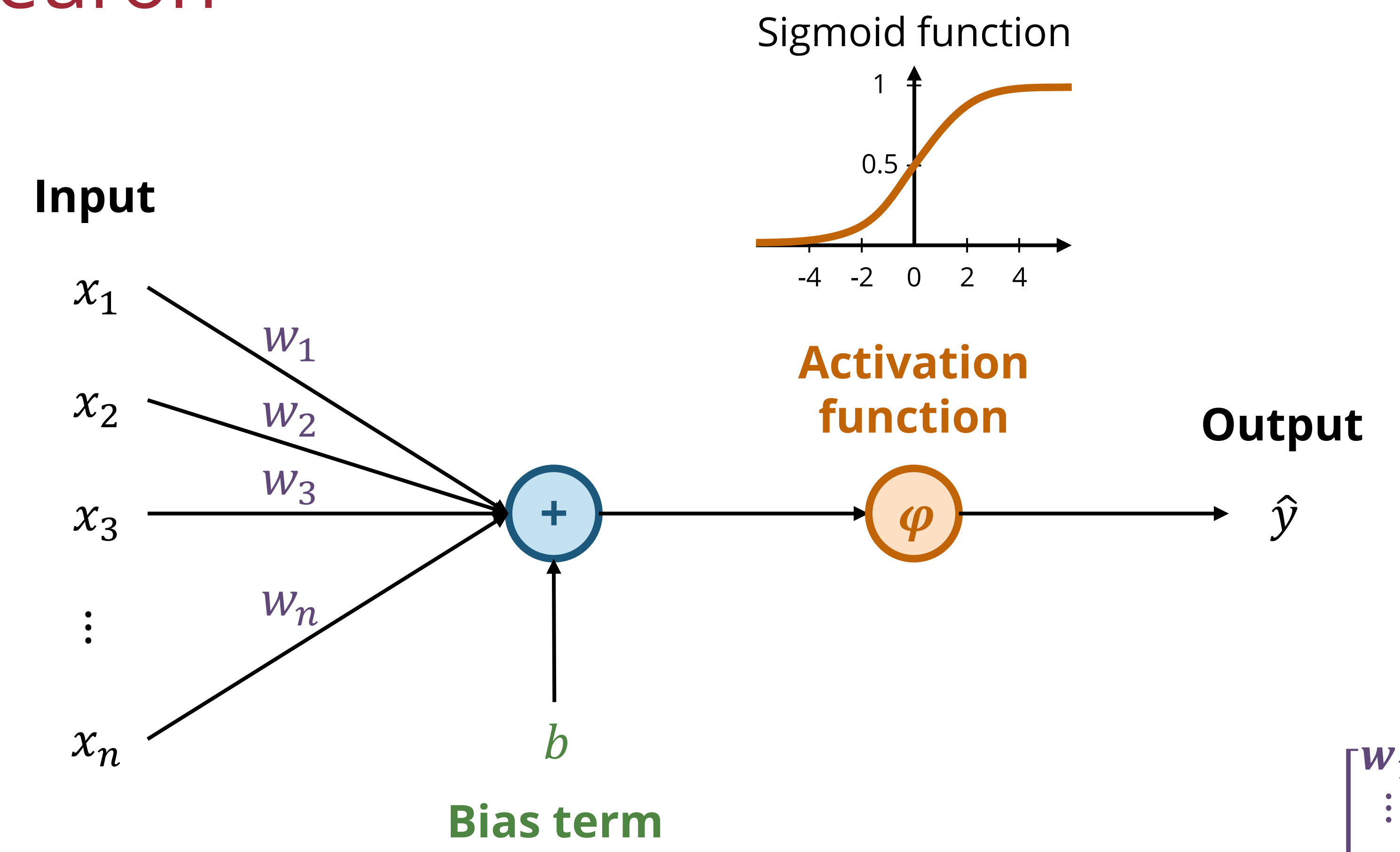
Review – Deep Learning Fundamentals

What is Deep Learning?

- A type of machine learning that uses **deep neural networks**



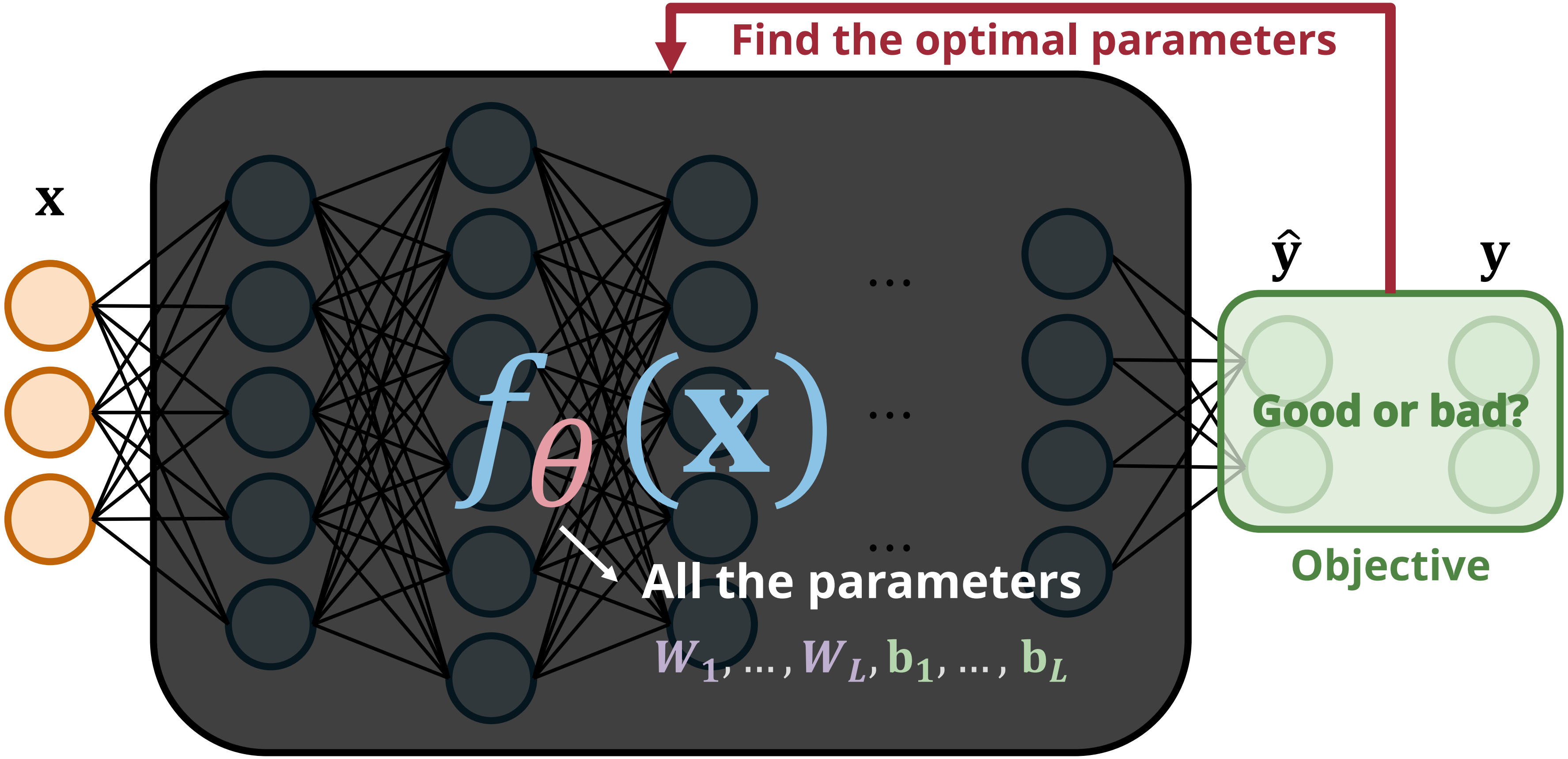
Inside a Neuron



$$\hat{\boldsymbol{y}} = \varphi(w_1x_1 + w_2x_2 + \cdots + w_nx_n + b) = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \varphi(\overset{\begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}}{\mathbf{w}} \cdot \overset{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}{\mathbf{x}} + b)$$

Neural Networks are Parameterized Functions

- A neural network represents **a set of functions**



Training a Neural Network

Build a neural network
(which defines a set of functions)

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

↓

Define the objective
(i.e., what is good for a function)

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}_k, \mathbf{y}_k)$$

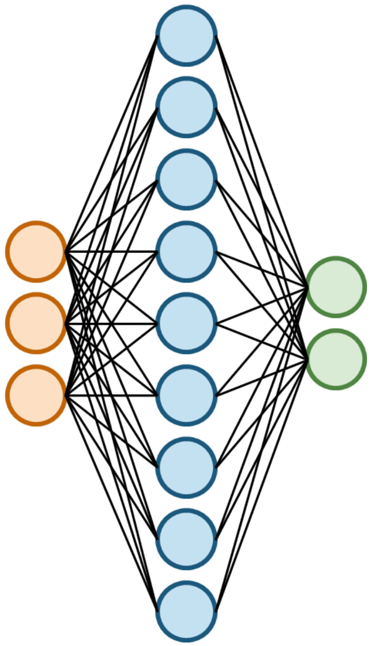
↓

Find the optimal parameters
(which leads to the best function)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

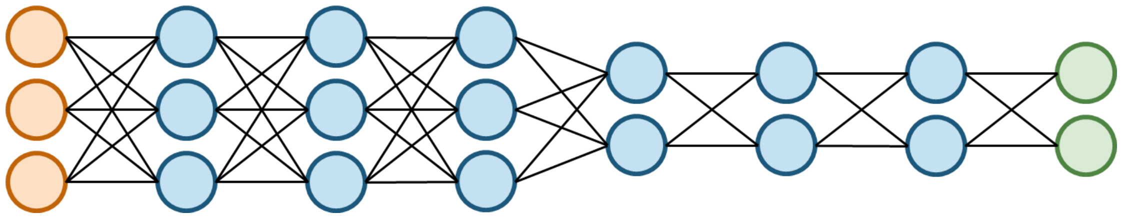
Shallow vs Deep Neural Networks – In Practice

Shallow neural nets



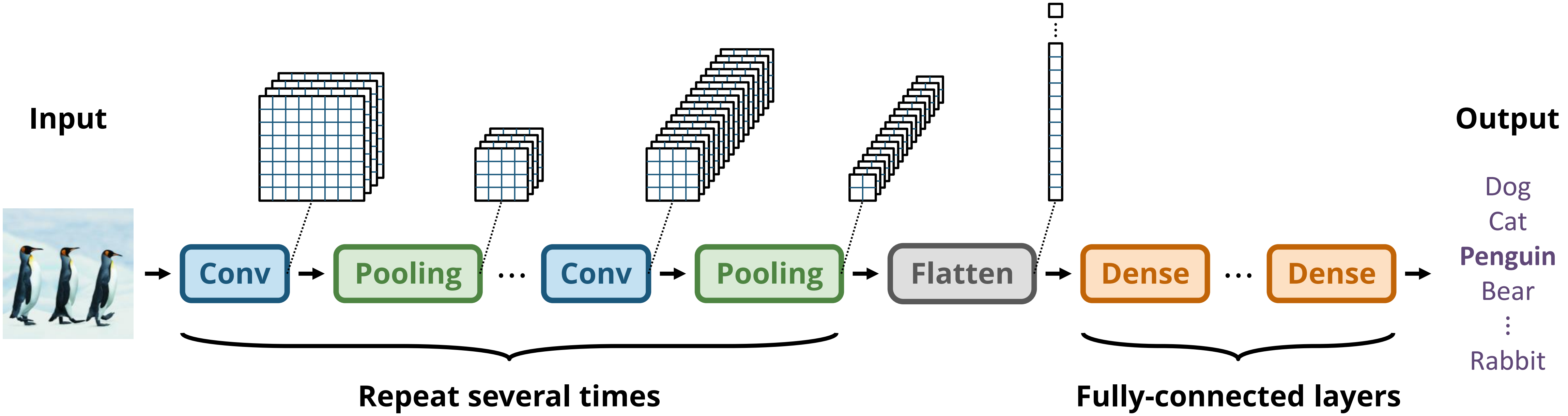
Less expressive
(less parameter efficient)

Deep neural nets

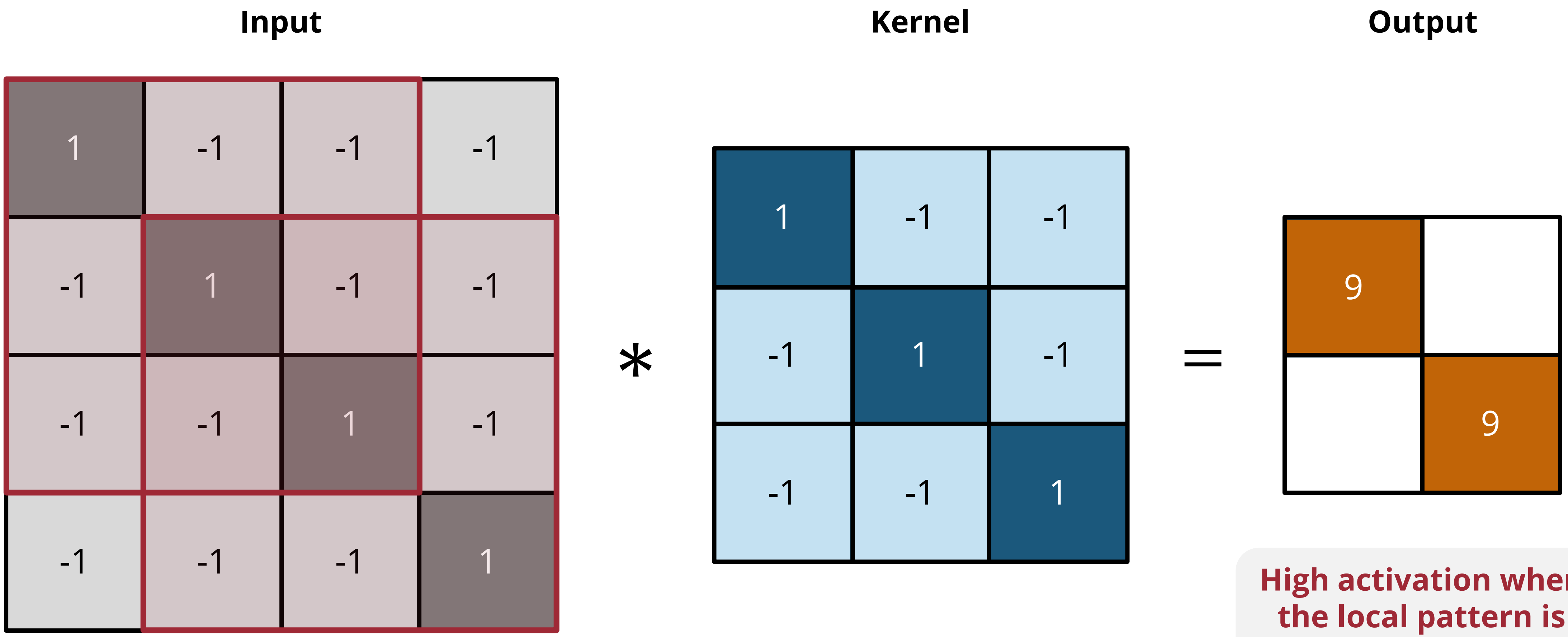


More expressive
(more parameter efficient)

Convolutional Neural Network (CNNs)



2D Convolution



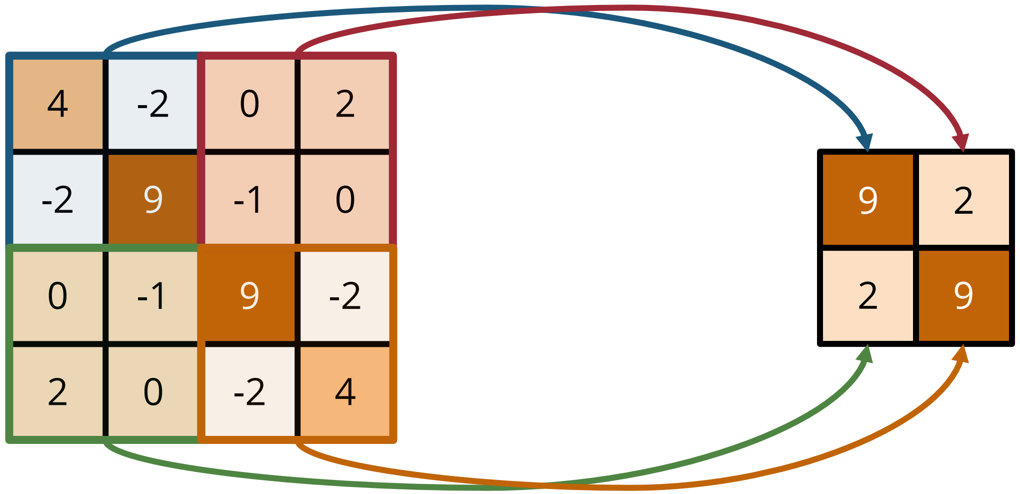
High activation when the local pattern is close to the kernel

2D Convolution

$$\underset{\textbf{Input}}{\begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}} * \underset{\textbf{Kernel}}{\begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix}} = \underset{\textbf{Output}}{\begin{bmatrix} & -1 \\ -1 & \end{bmatrix}}$$

Low activation when the local pattern differs from the kernel

Max Pooling Layer



Downsample and keep the strongest activation in each block

Learned CNN Kernels in a Trained AlexNet

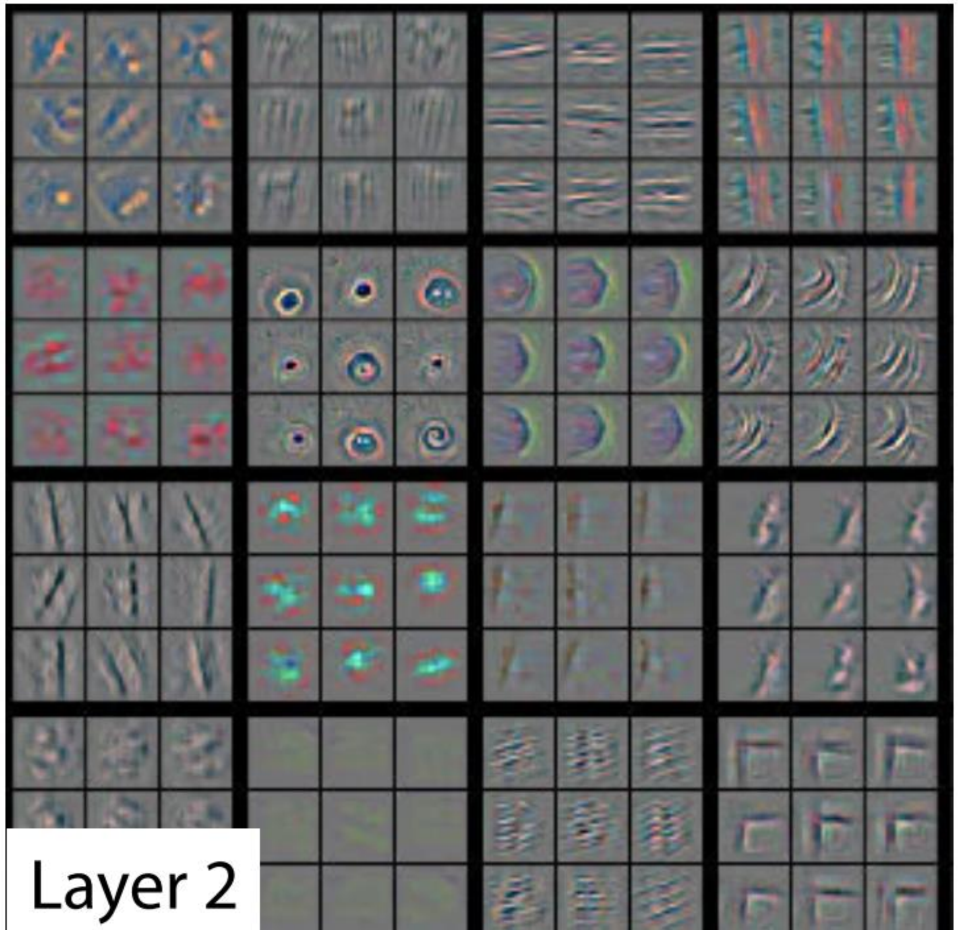
Layer 1

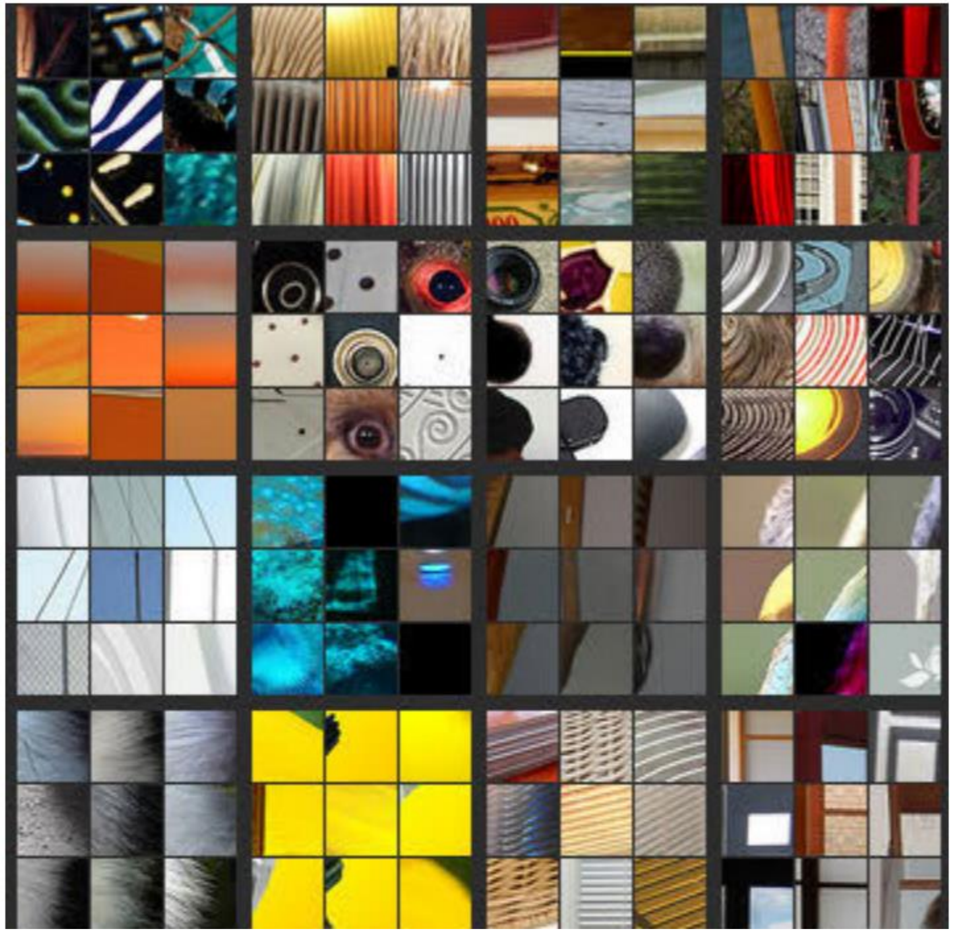
Learned CNN kernels

Top activations

Matthew D. Zeiler and Rob Fergus, "Visualizing and Understanding Convolutional Networks," *ECCV*, 2014.

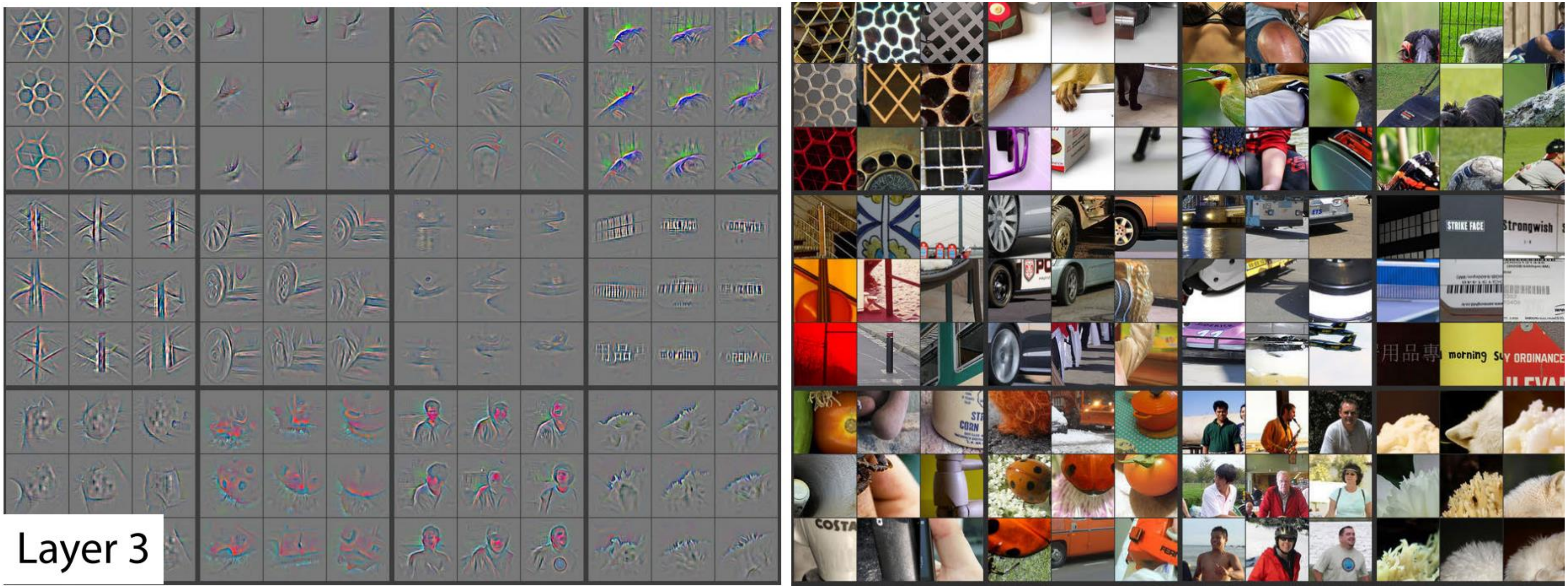
Learned CNN Kernels in a Trained AlexNet





Matthew D. Zeiler and Rob Fergus, "Visualizing and Understanding Convolutional Networks," *ECCV*, 2014.

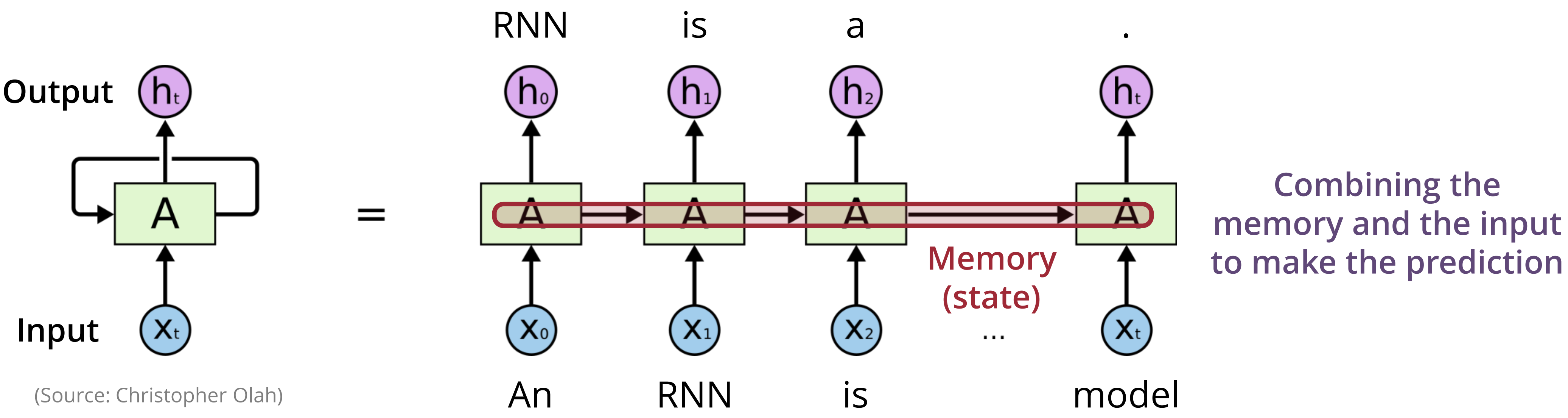
Learned CNN Kernels in a Trained AlexNet



Matthew D. Zeiler and Rob Fergus, "Visualizing and Understanding Convolutional Networks," *ECCV*, 2014.

What is an RNN (Recurrent Neural Network)?

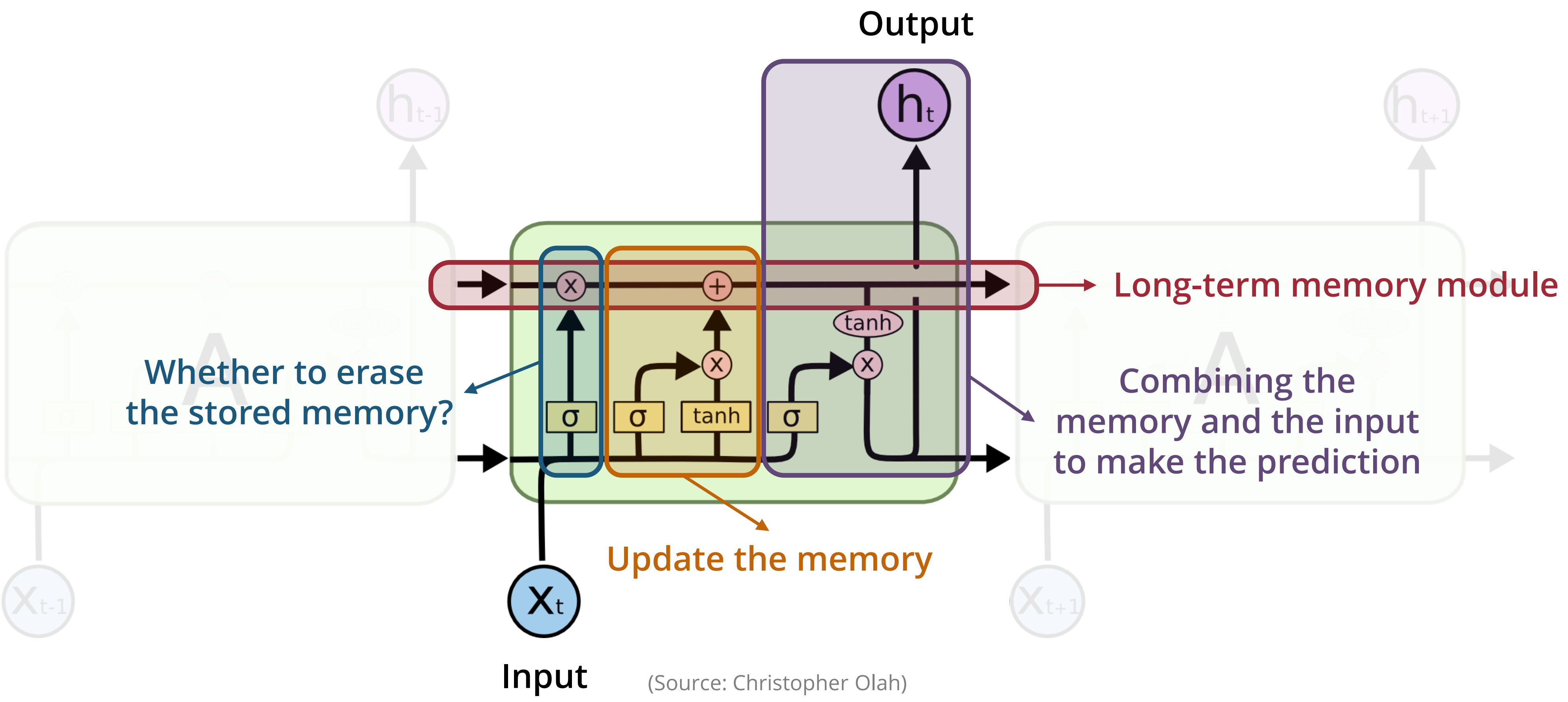
- A type of neural networks that have **loops**
- Widely used for **modeling sequences** (e.g., in natural language processing)



(Source: Christopher Olah)

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

Demystifying LSTMs



(Source: Christopher Olah)

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

Training a Neural Network

Build a neural network
(which defines a set of functions)

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

↓

Define the objective
(i.e., what is good for a function)

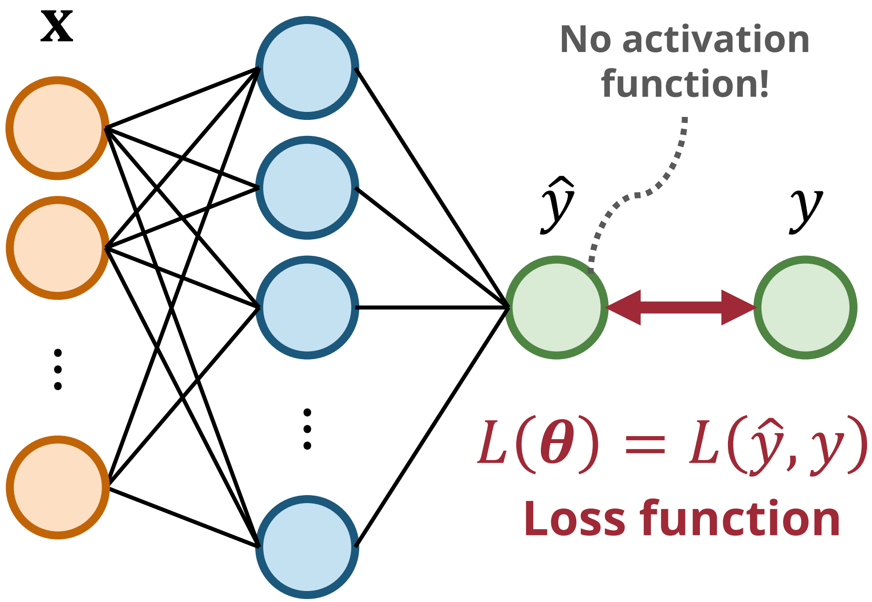
$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}_k, \mathbf{y}_k)$$

↓

Find the optimal parameters
(which leads to the best function)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Common Loss Functions for Regression



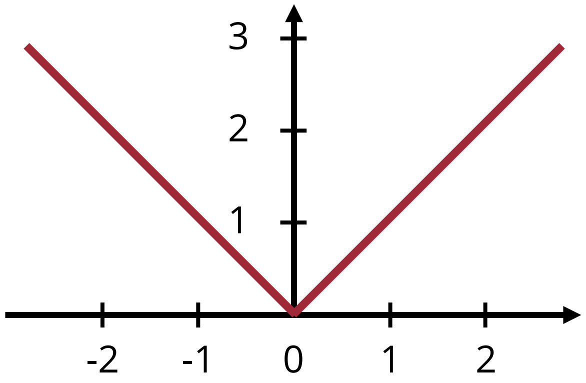
$$L(\boldsymbol{\theta}) = L(\hat{y}, y)$$

Loss function

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{y}_k, y_k)$$

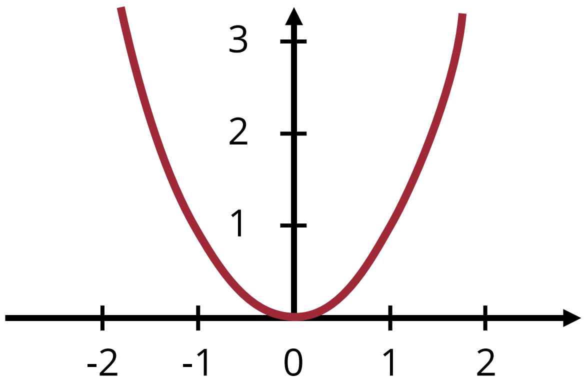
L1 loss

$$L(\hat{y}, y) = |\hat{y} - y|$$



L2 loss

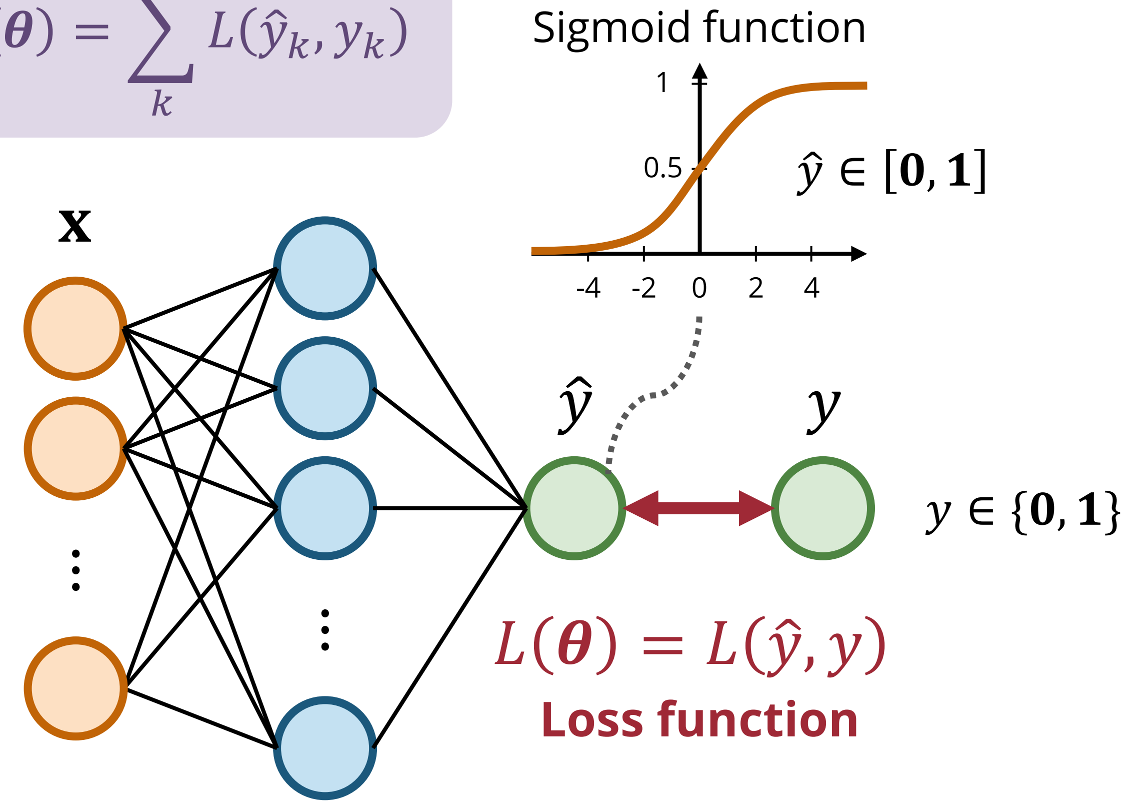
$$L(\hat{y}, y) = (\hat{y} - y)^2$$



Binary Cross Entropy for Binary Classification

- **Logistic regression** approaches classification like regression

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{y}_k, y_k)$$

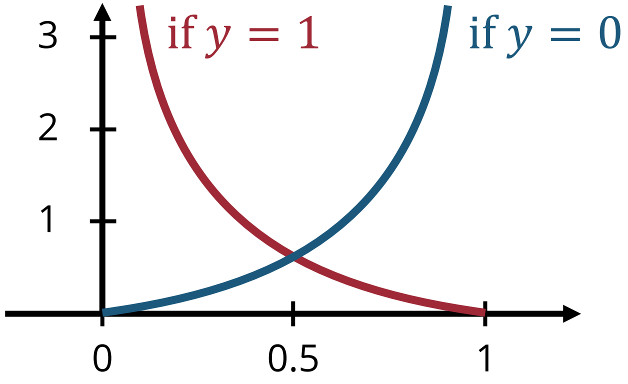


Binary cross entropy

(Also called log loss)

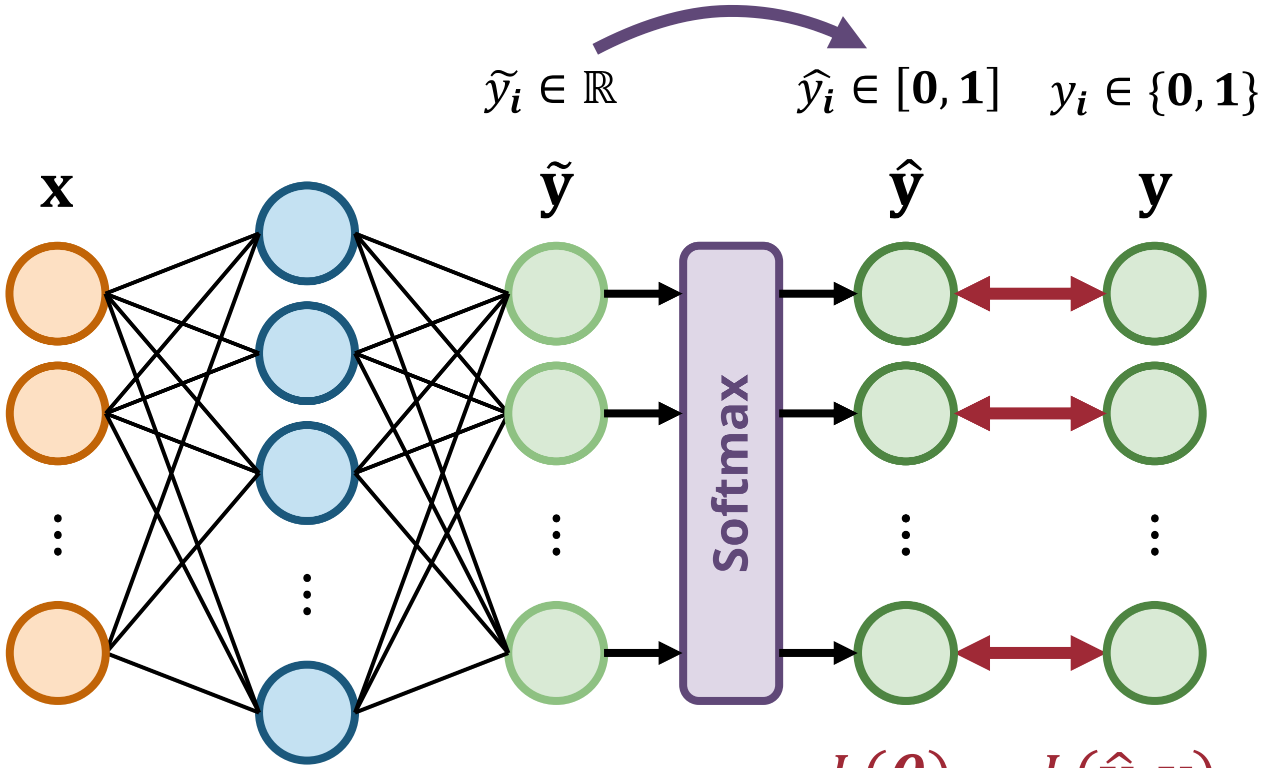
$$L(\hat{y}, y) = \begin{cases} -\log \hat{y}\,, & \text{if } y = 1 \\ -\log(1-\hat{y})\,, & \text{if } y = 0 \end{cases}$$

$$= -y \log \hat{y} - (1-y)\log(1-\hat{y})$$



Cross Entropy for Multiclass Classification

Real-valued numbers to probability-like numbers



$$L(\boldsymbol{\theta}) = L(\hat{\mathbf{y}}, \mathbf{y})$$

Loss function

Softmax

$$\hat{y}_i = \frac{e^{\tilde{y}_i}}{\sum_{j=1}^{n} e^{\tilde{y}_j}}$$

Cross entropy

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i}^{n} y_i \log \hat{y}_i$$

$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}_k, \mathbf{y}_k)$$

Cross Entropy for Multiclass Classification

Binary Cross Entropy

Only one of them will be one!

$$L(\hat{y}, y) = -\boxed{y}\log \hat{y} - \boxed{(1-y)}\log(1-\hat{y})$$

Cross Entropy

Only one of them will be one!

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\boxed{y_1}\log \hat{y}_1 - \boxed{y_2}\log \hat{y}_2 - \cdots - \boxed{y_i}\log \hat{y}_n$$

$$= -\boxed{\sum_{i}^{n} y_i \log \hat{y}_i}$$

Log likelihood

Training a Neural Network

Build a neural network
(which defines a set of functions)

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

Define the objective
(i.e., what is good for a function)

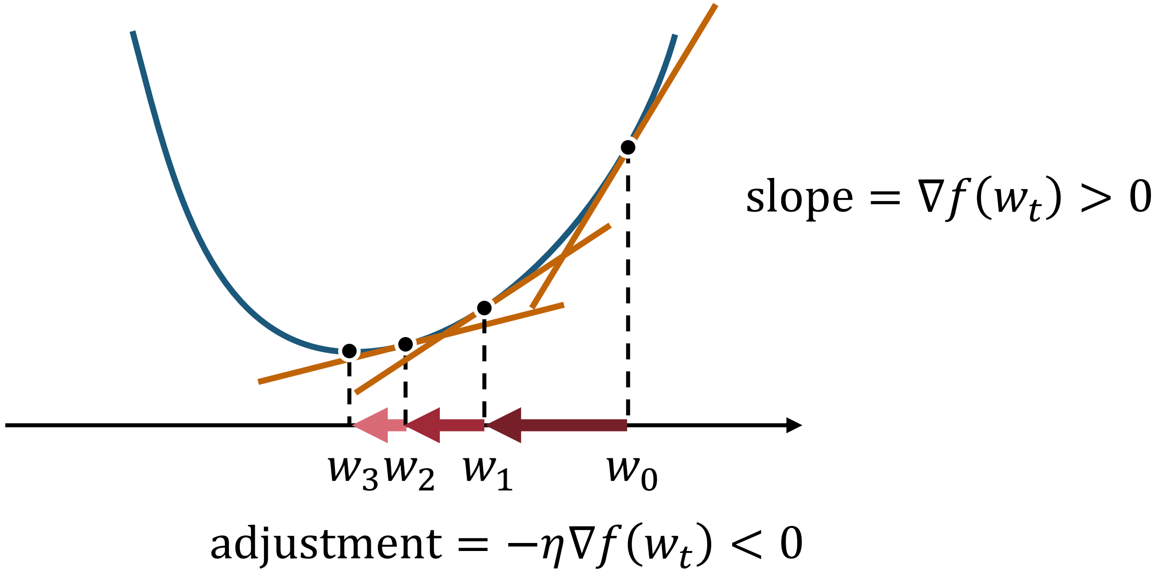
$$Loss(\boldsymbol{\theta}) = \sum_{k}^{N} L(\hat{\mathbf{y}}_k, \mathbf{y}_k)$$

Find the optimal parameters
(which leads to the best function)

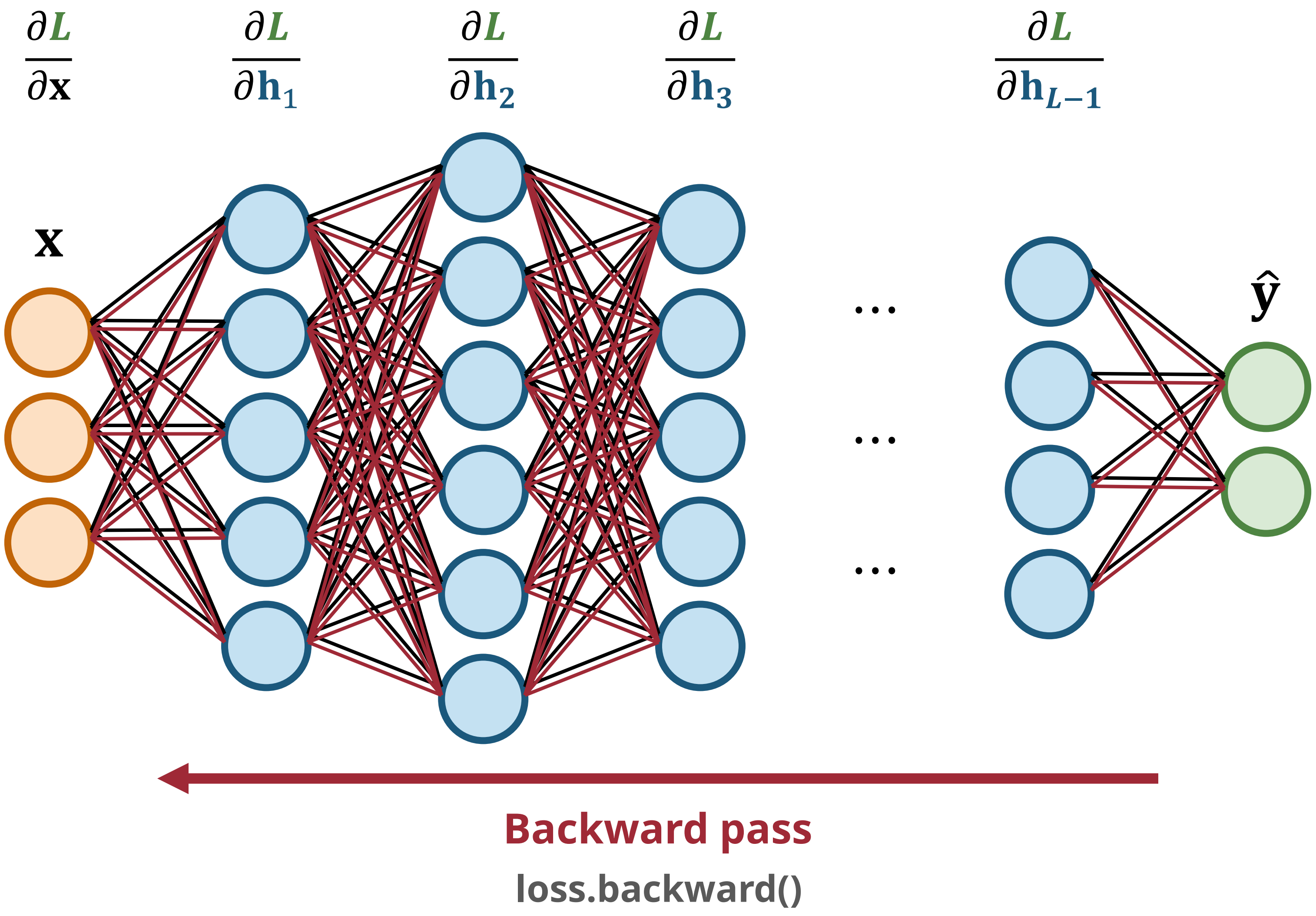
$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



Forward Pass & Backward Pass

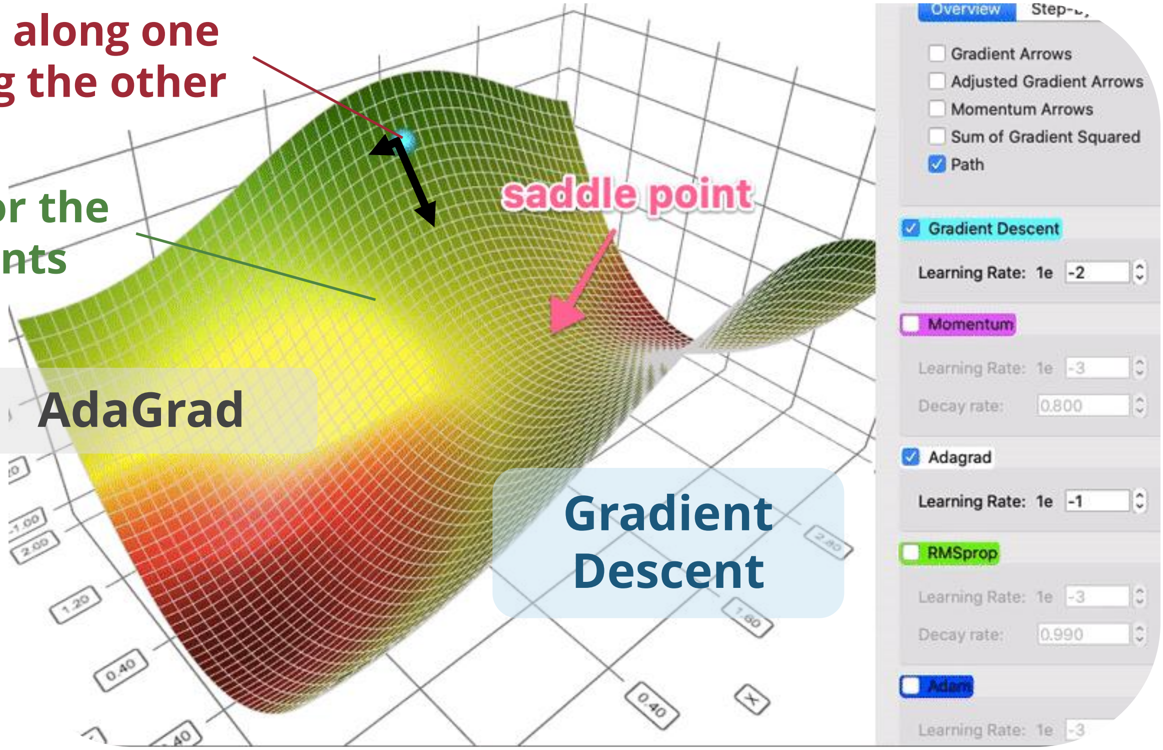


Gradient-based Adaptive Learning Rate

- **Intuition**: **Compensate axis that has little progress** by comparing the current gradients to the previous gradients

Large gradients along one axis, small along the other

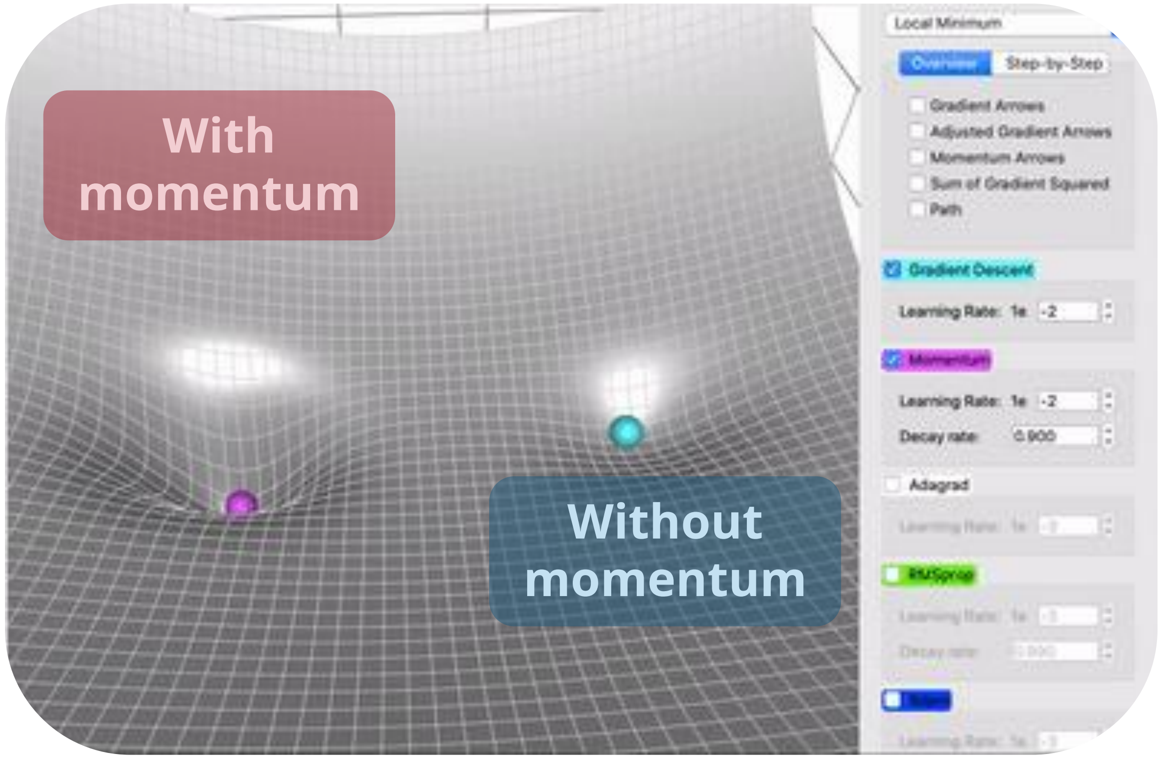
Use larger learning rate for the axis with smaller gradients



towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

Momentum

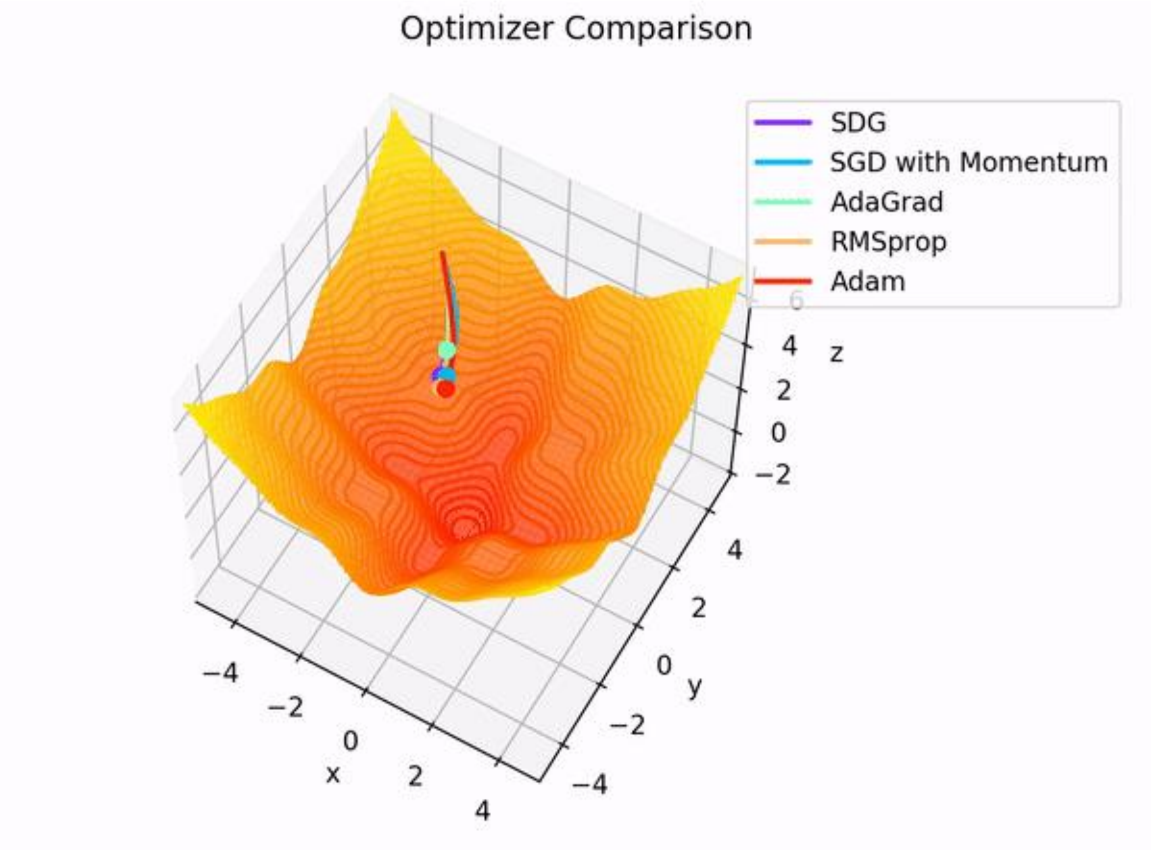
- **Intuition**: Maintain the momentum to **escape from local minima**



Comparison of Optimizers

- **Momentum**
  - Gets you out of spurious local minima
  - Allows the model to explore around

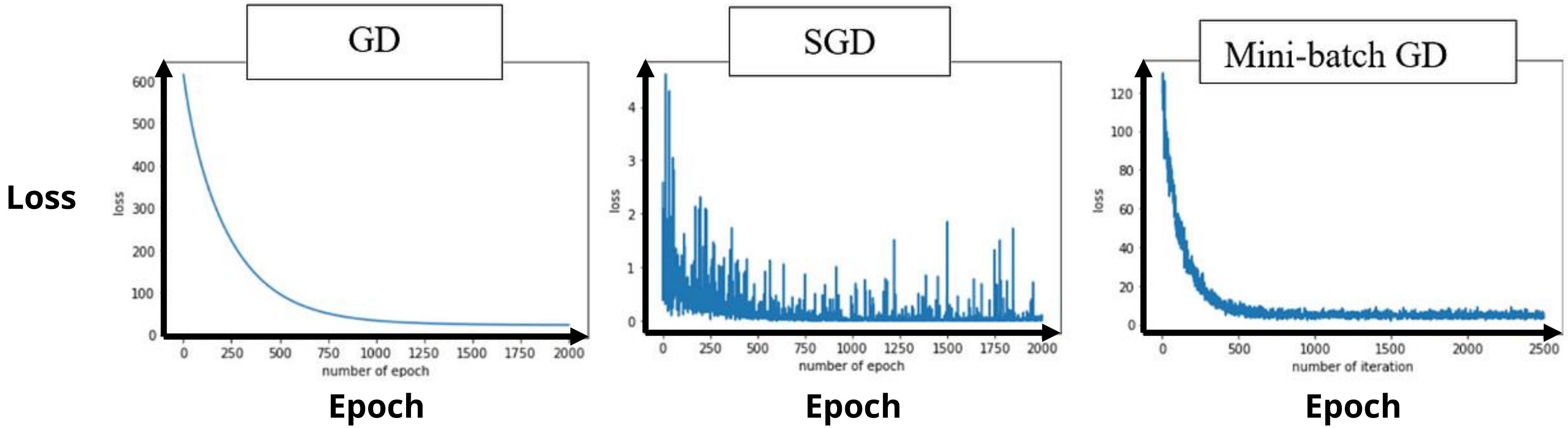
- **Gradient-based adaption**
  - Maintains steady improvement
  - Allows faster convergence



medium.com/@LayanSA/complete-guide-to-adam-optimization-1e5f29532c3d

Mini-batch Gradient Descent

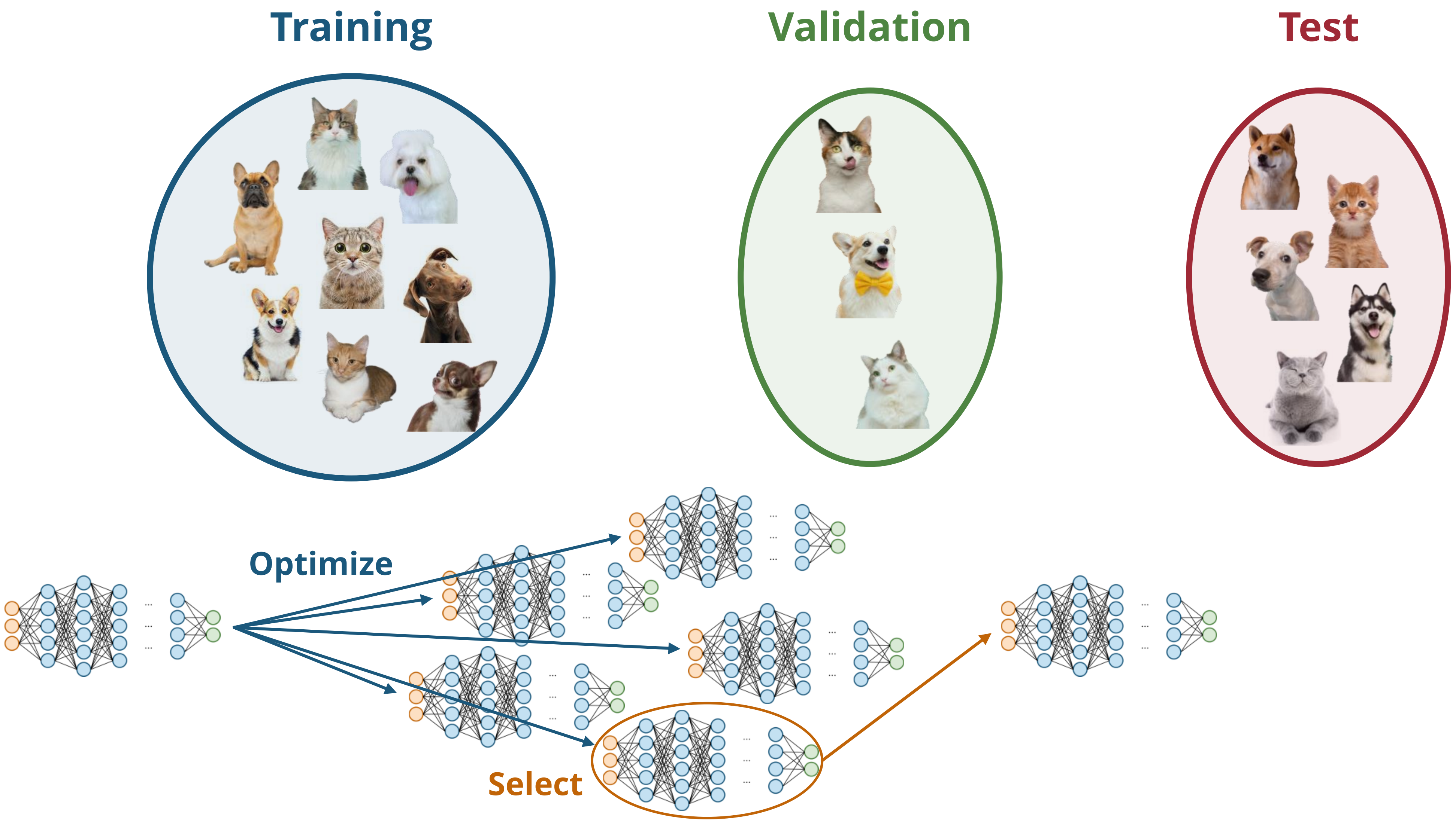
- **Intuition:** **Estimate** the gradient using **several random training samples**



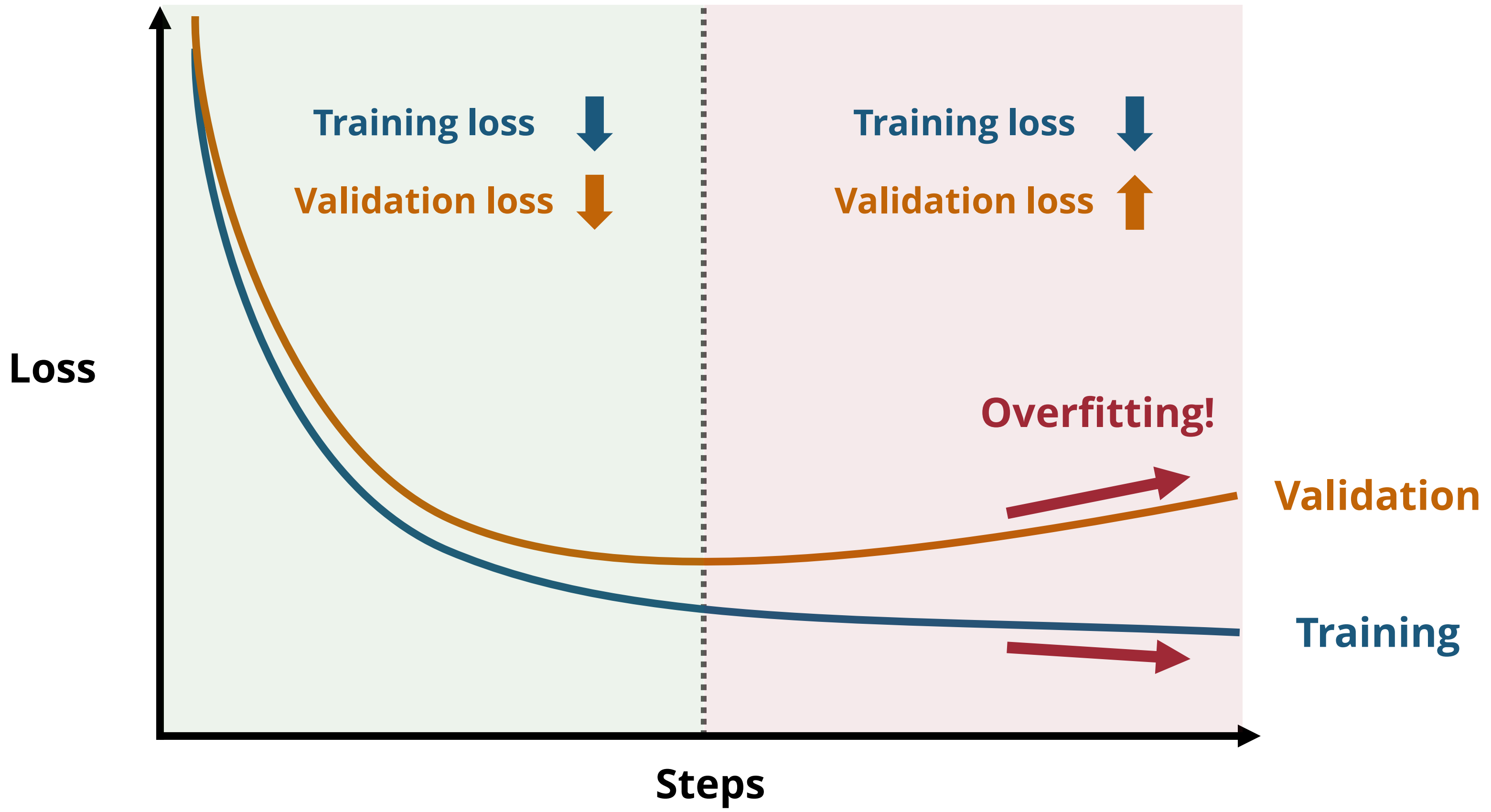
$\text{batch size} = N$ $\quad$ $\text{batch size} = 1$ $\quad$ $1 < \text{batch size} < N$

analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/

Training–Validation–Test Pipeline



Training vs Validation Losses



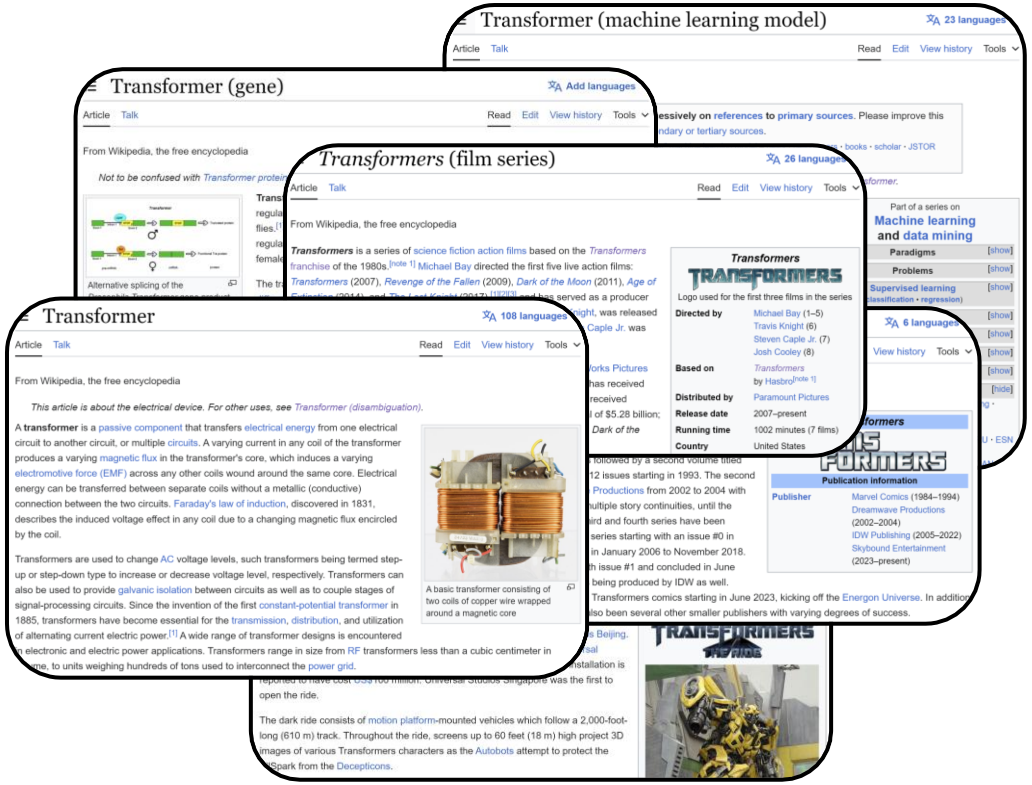
Review – Deep Generative Models

Network Architectures vs Training Frameworks

Network architectures	Training frameworks
Multilayer perceptron (MLP)	Autoregressive
Convolutional neural networks (CNNs)	Autoencoders
Recurrent neural networks (RNNs)	Variational autoencoders (VAEs)
Transformers	Generative adversarial networks (GANs)
ResNets	Diffusion models
U-Nets	Consistency models
⋮	⋮

Language Models

- Predicting the next word **given the past sequence of words**



A transformer is a ________

Language Models (Mathematically)

- A class of machine learning models that learn the next word probability

$$P(\underbrace{x_i}_{\text{Next word}} \mid \underbrace{x_1, x_2, \ldots, x_{i-1}}_{\text{Previous words}})$$

$P($ electrical $\mid$ A transformer is a $)$ ↑

$P($ character $\mid$ A transformer is a $)$ ↑

$P($ gene $\mid$ A transformer is a $)$ ↑

$P($ model $\mid$ A transformer is a $)$ ↑

$P($ food $\mid$ A transformer is a $)$ ↓

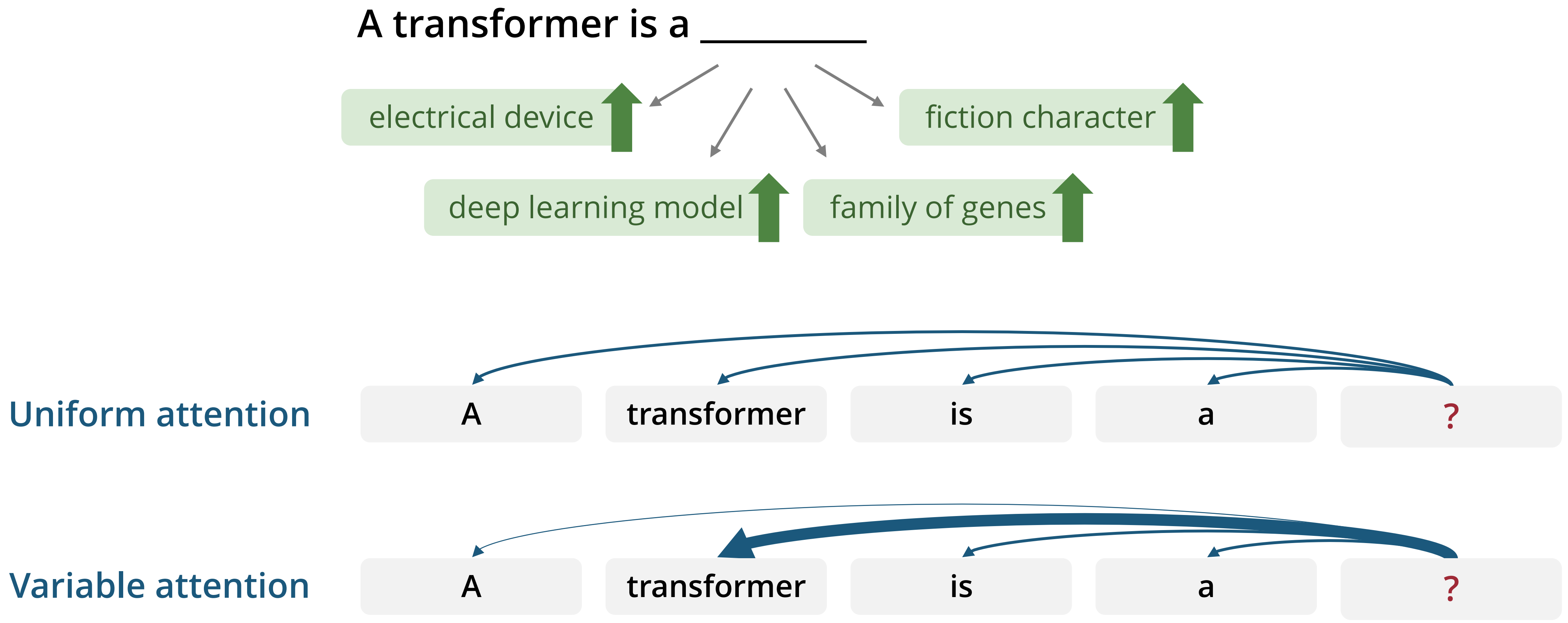
$P($ musical $\mid$ A transformer is a $)$ ↓

Language Models – Generation

- How do we generate a new sentence using a trained language model?

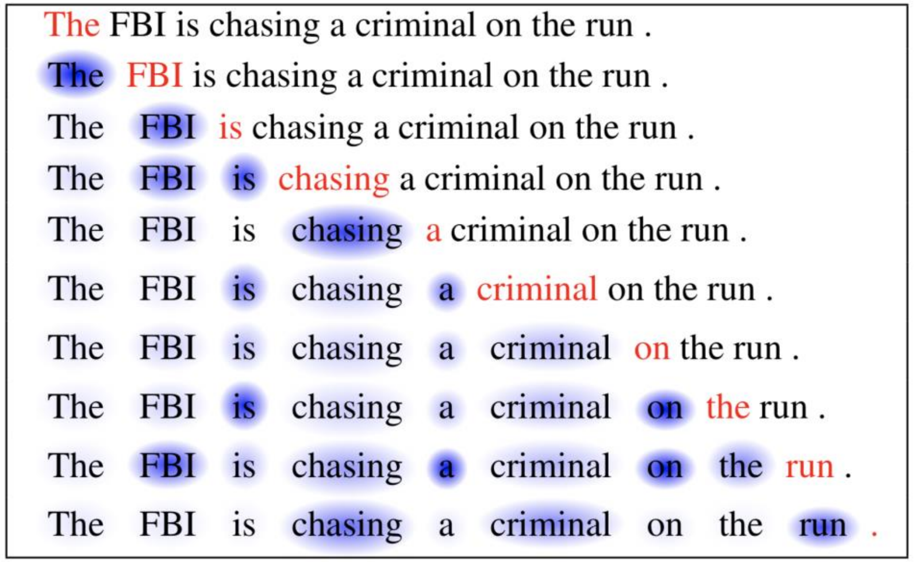
A transformer is a	→	Model	→	deep
A transformer is a deep	→	Model	→	learning
A transformer is a deep learning	→	Model	→	model
A transformer is a deep learning model	→	Model	→	introduced
A transformer is a deep learning model introduced	→	Model	→	in
A transformer is a deep learning model introduced in	→	Model	→	2017

Demystifying Transformers

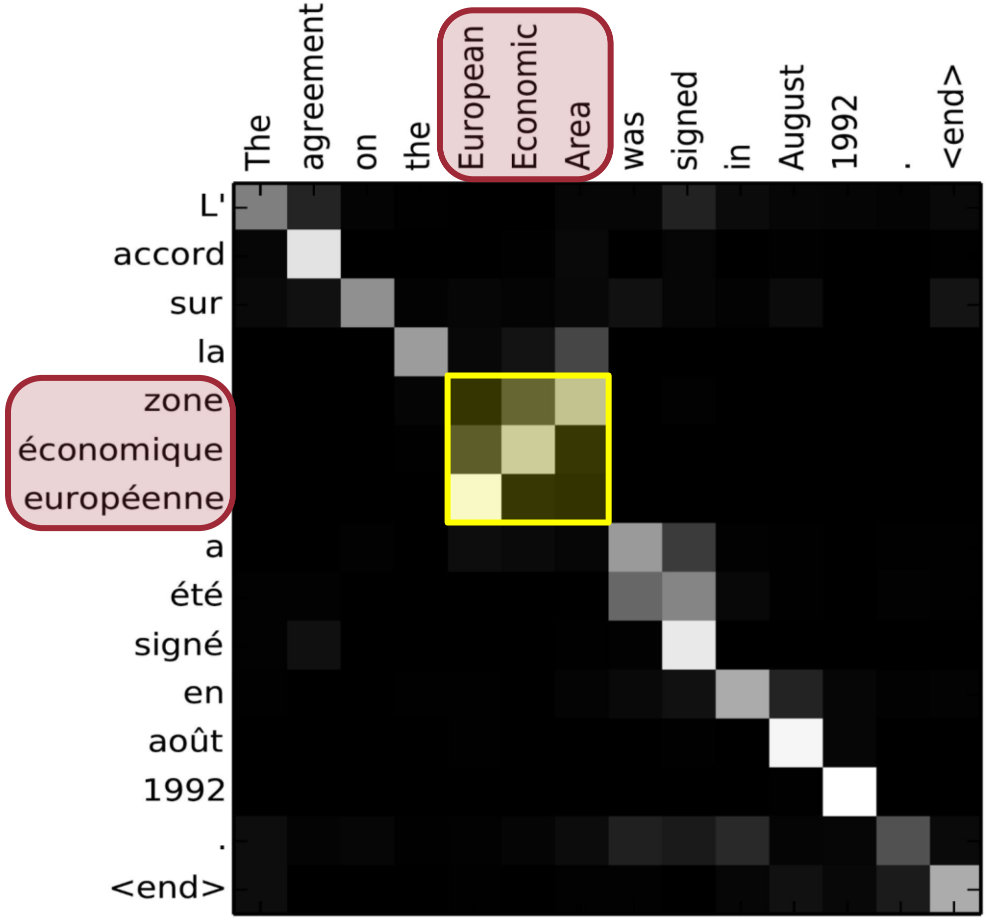


Transformers learn what to attend to from big data!

What does a Transformer Learn?



(Source: Cheng et al., 2016)

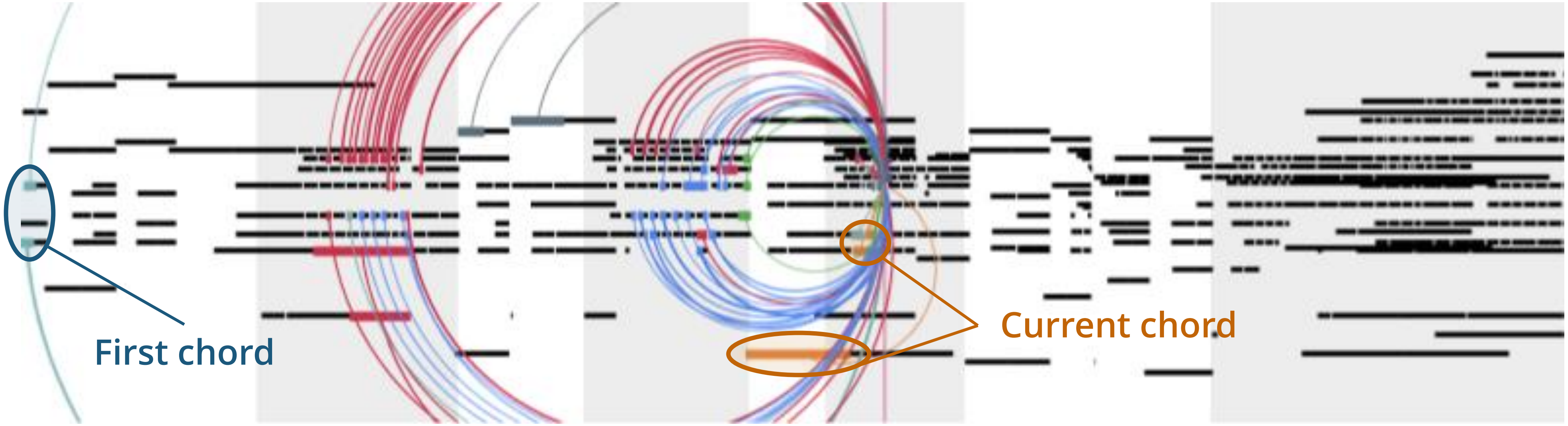


(Source: Bahdanau et al., 2015)

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *ICLR*, 2015.
Jianpeng Cheng, Li Dong, and Mirella Lapata, "Long Short-Term Memory-Networks for Machine Reading," *EMNLP*, 2016.

What does a Transformer Learn?

(Each color represents an attention head)



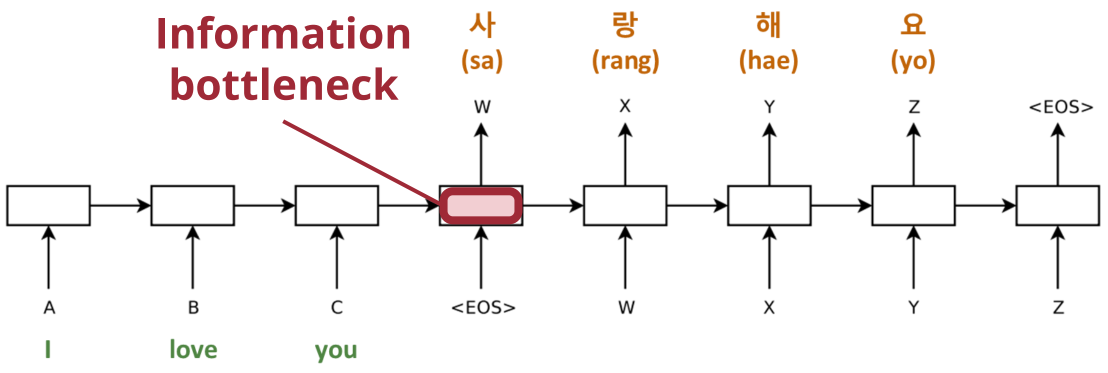
(Source: Huang et al., 2018)

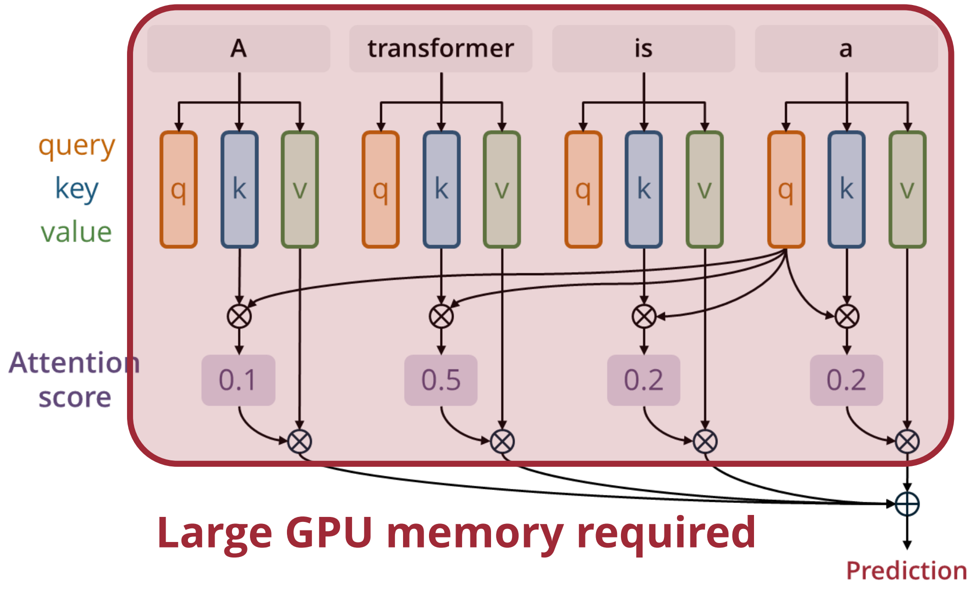
Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck, "Music Transformer: Generating Music with Long-Term Structure," *Magenta Blog*, December 13, 2018.

Seq2seq vs Transformers

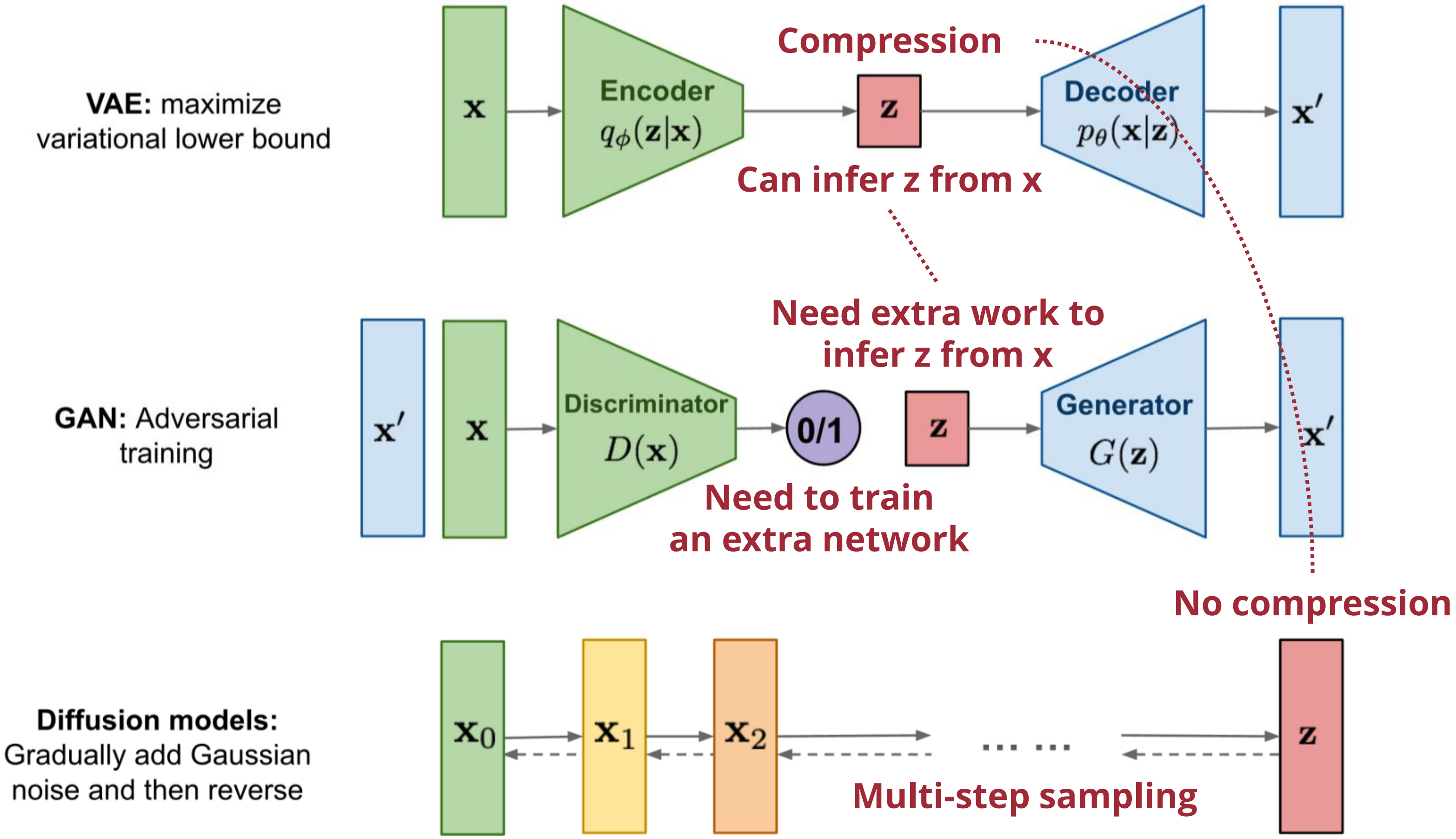
Seq2seq

Transformers





Comparison of Deep Generative Models

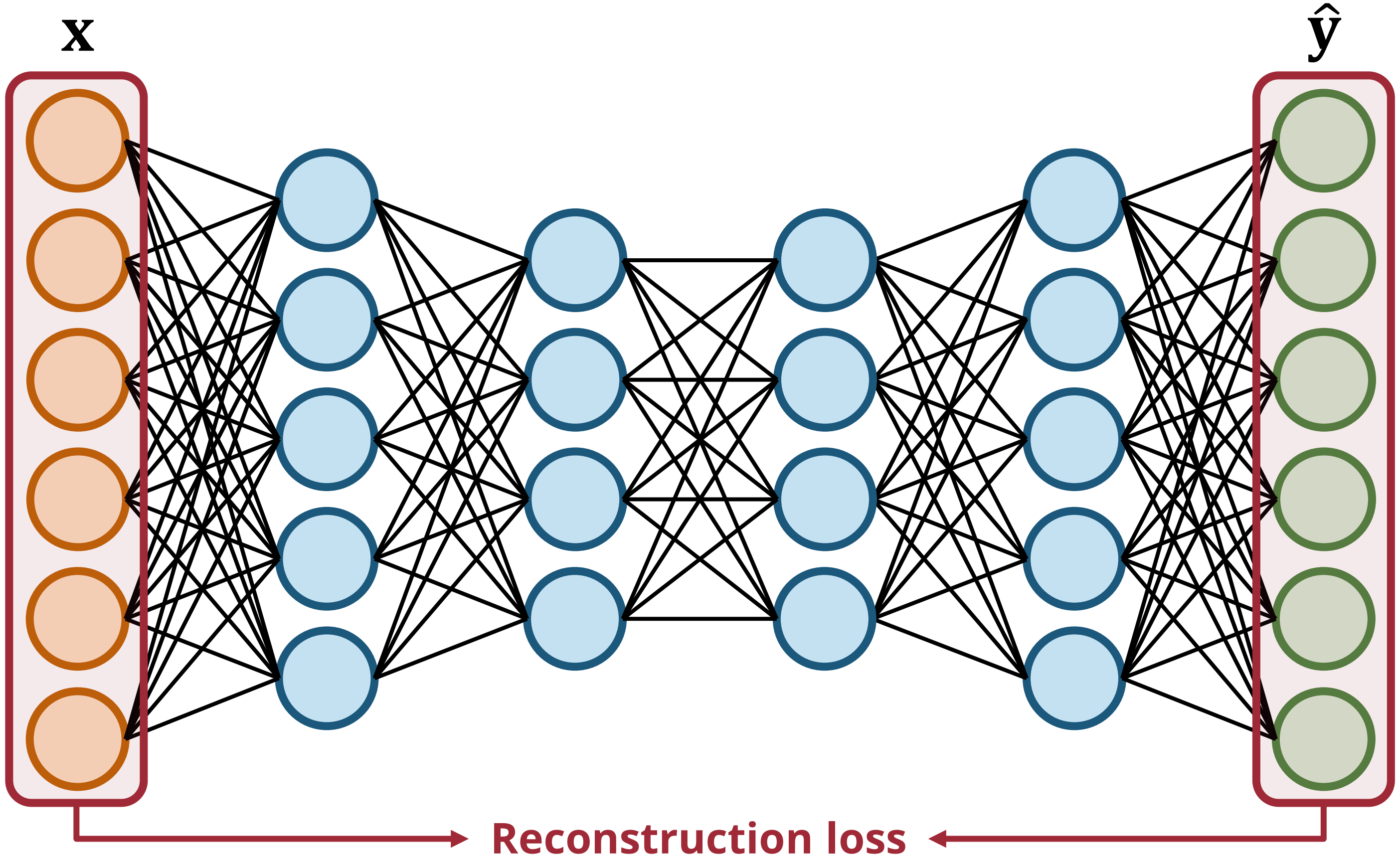


(Source: Weng, 2021)

lilianweng.github.io/posts/2021-07-11-diffusion-models/

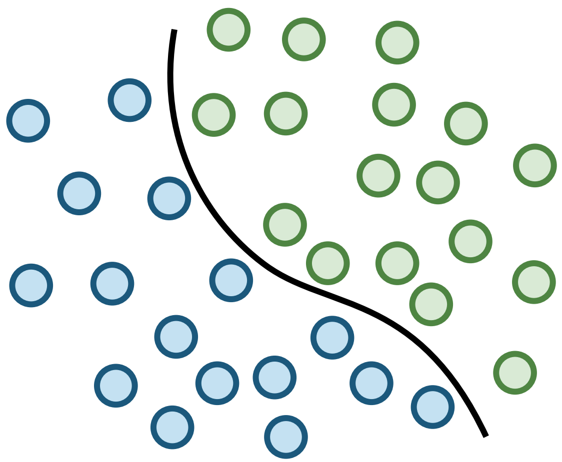
Autoencoders

- A neural network where the **input and output are the same**



Discriminative vs Generative Models

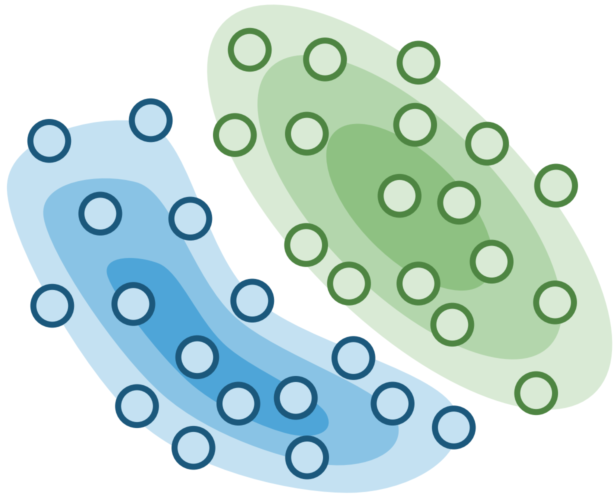
Discriminative



Discriminative models learn the decision boundary

$P(y|x)$

Generative

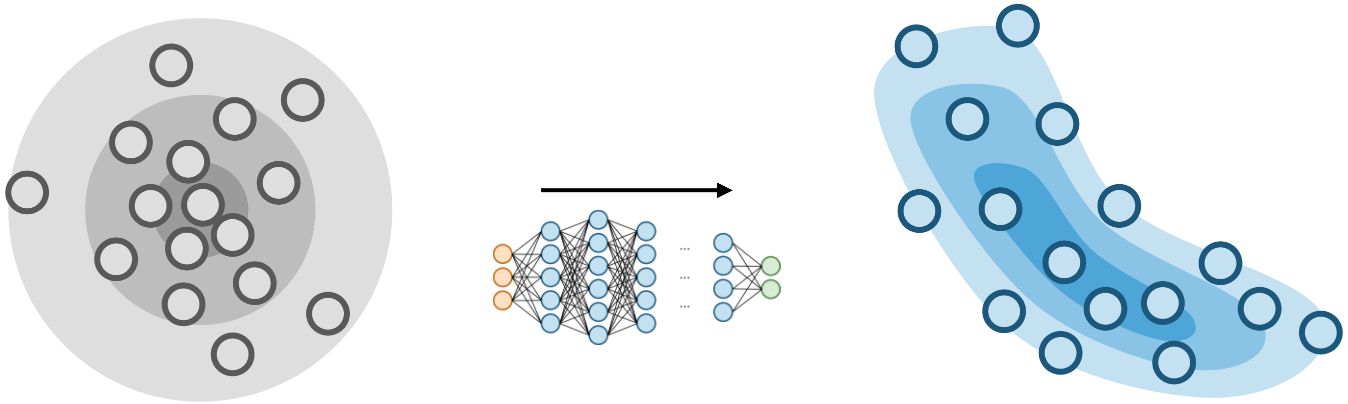


Generative models learn the underlying distribution

$P(x)$ or $P(x|y)$

Generating Data from a Random Distribution

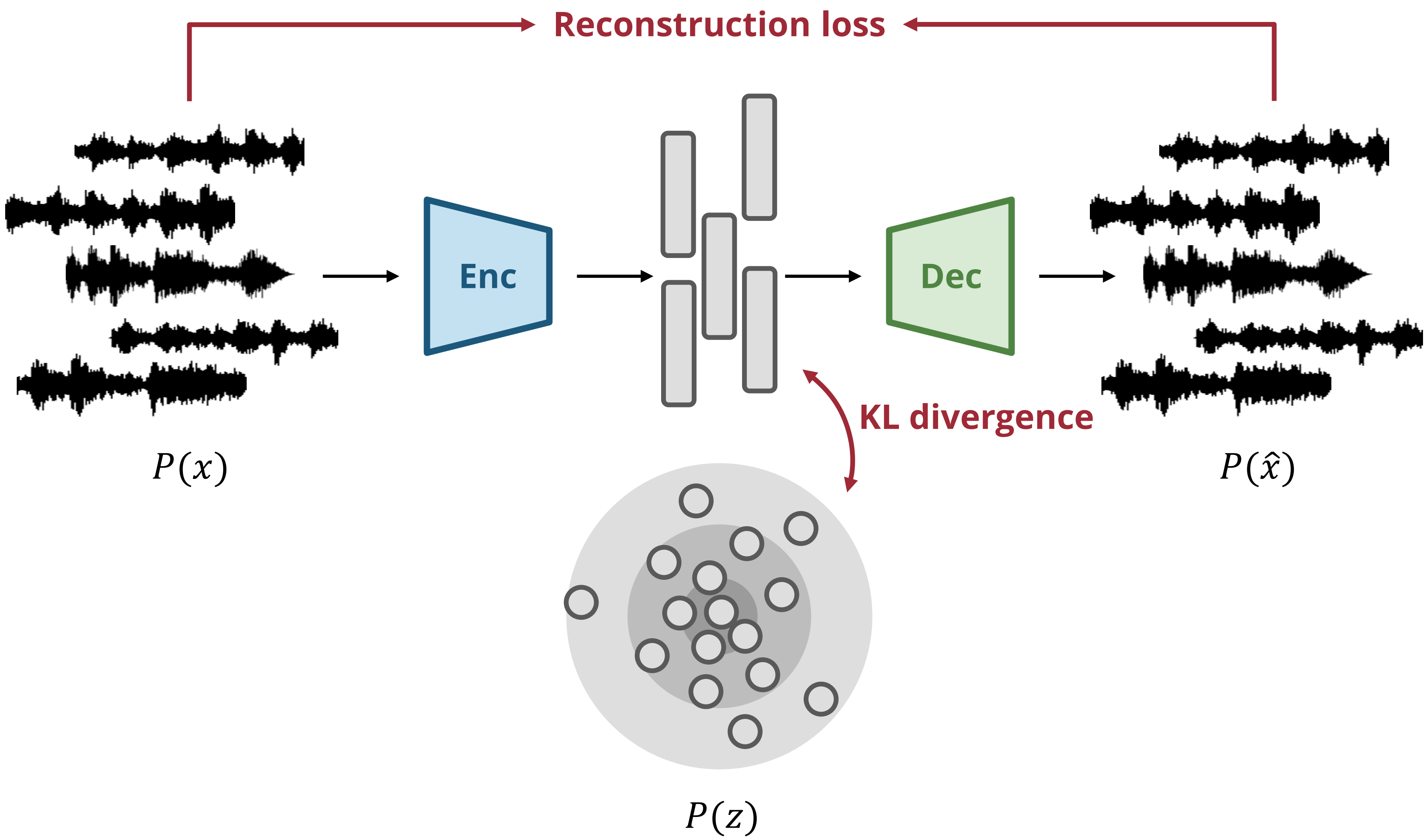
Random distribution **Data distribution**



$P(z)$ $P(x)$

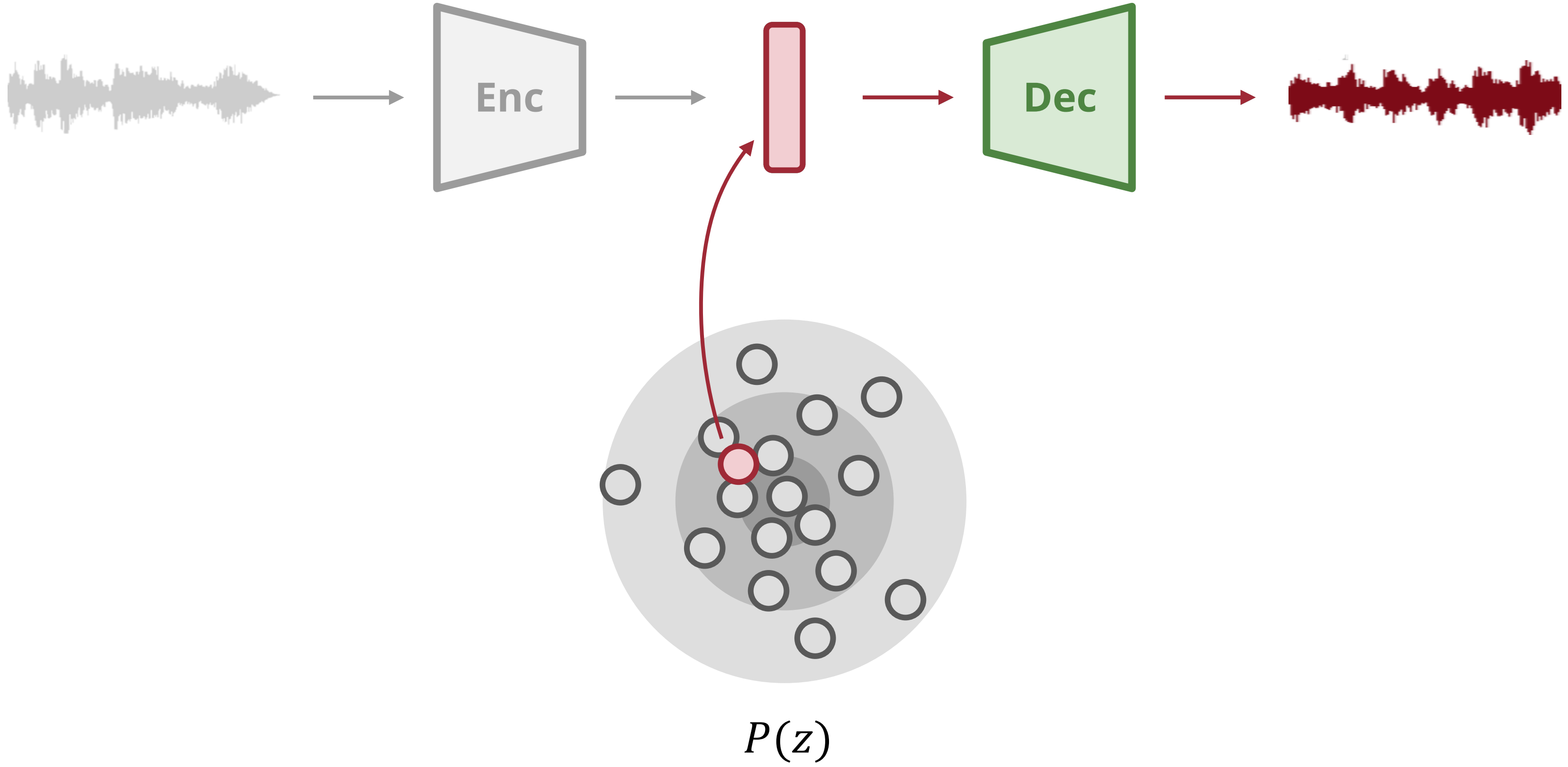
If we can learn this mapping, we can easily generate new samples from the data distribution

Variational Autoencoders (VAEs) – Training



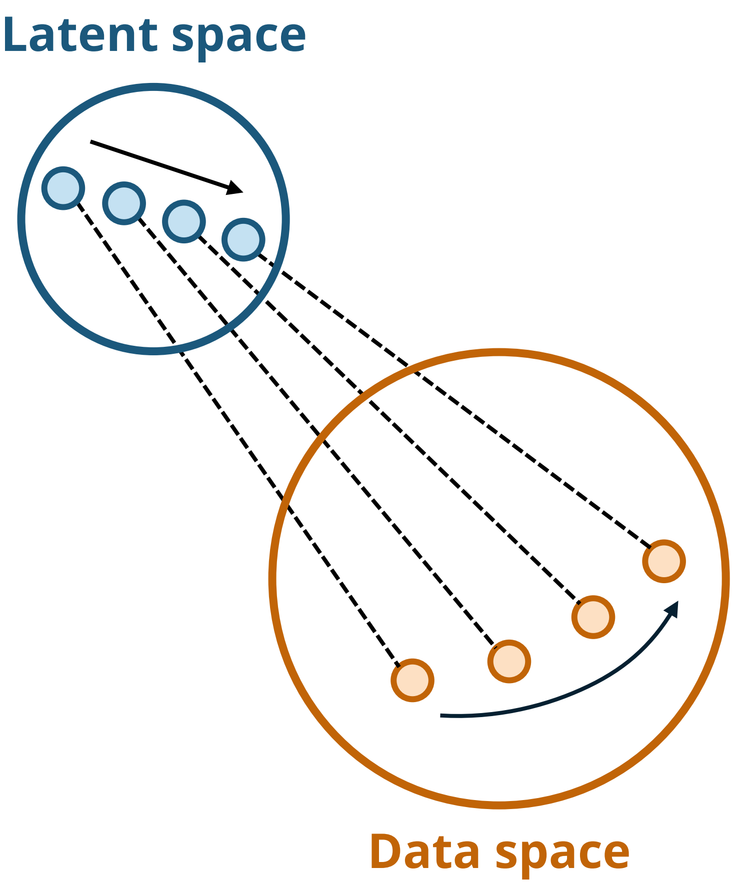
Diederik P Kingma and Max Welling, "Auto-Encoding Variational Bayes," *ICLR*, 2014.

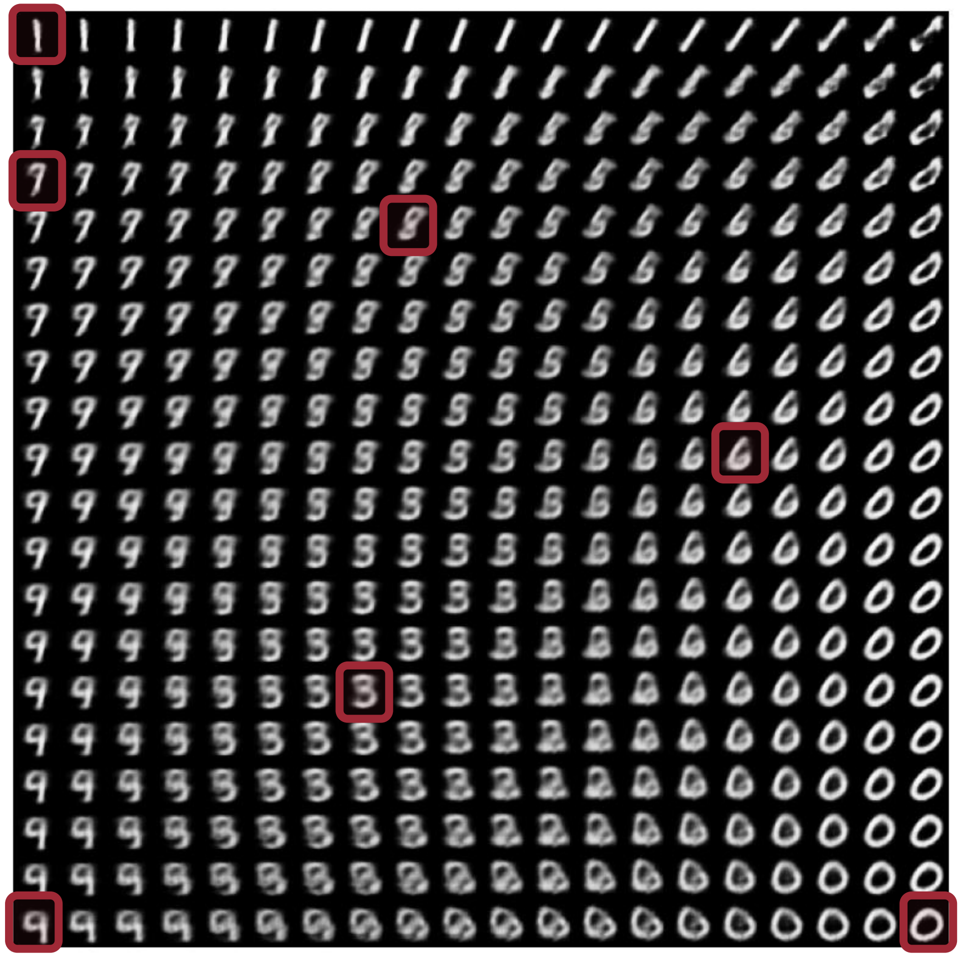
Variational Autoencoders (VAEs) – Generation



Diederik P Kingma and Max Welling, "Auto-Encoding Variational Bayes," *ICLR*, 2014.

Decoding the Latent Space of a VAE

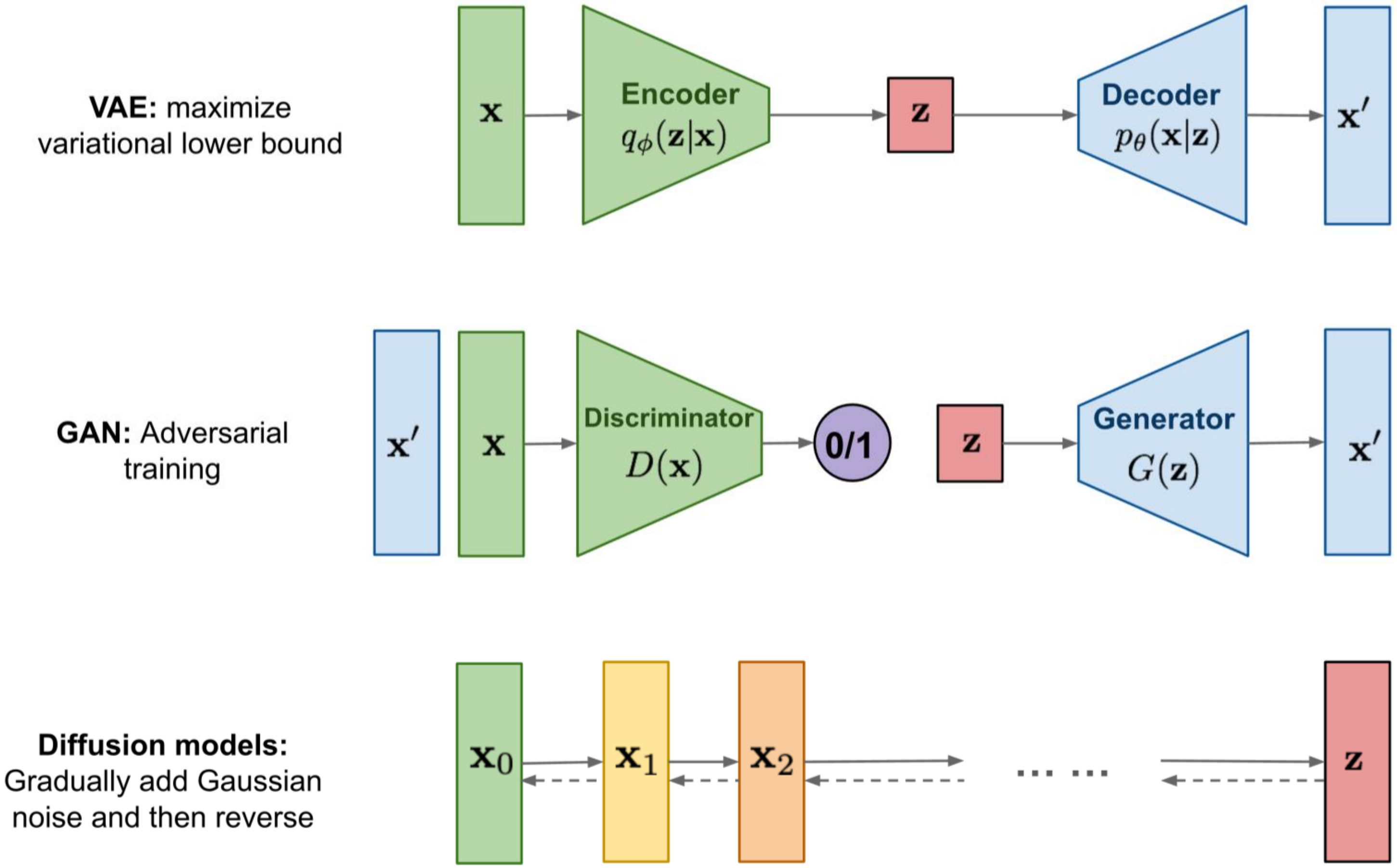




(Source: tensorflow.org)

tensorflow.org/tutorials/generative/autoencoder

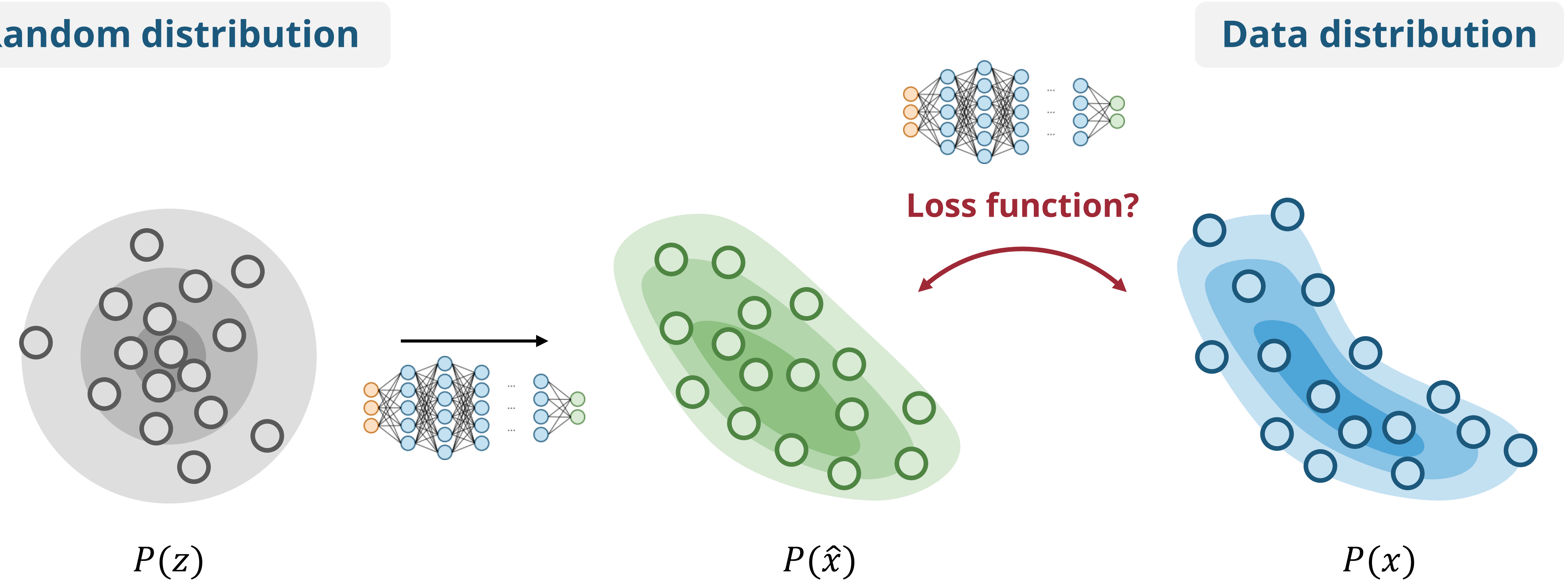
Comparison of Deep Generative Models



(Source: Weng, 2021)

lilianweng.github.io/posts/2021-07-11-diffusion-models/

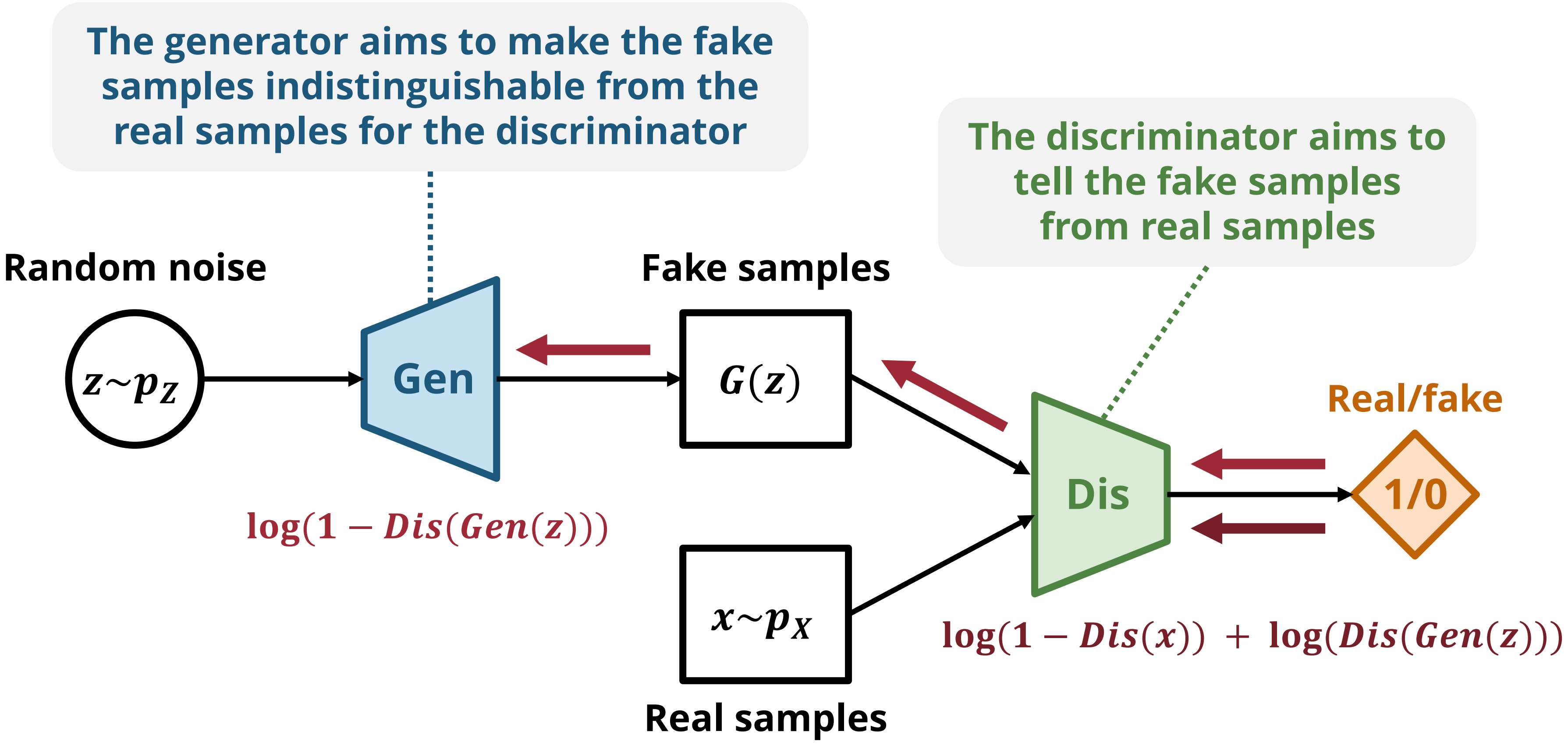
A Loss Function for Distributions



Unfortunately, no easy way to measure the difference between two distributions

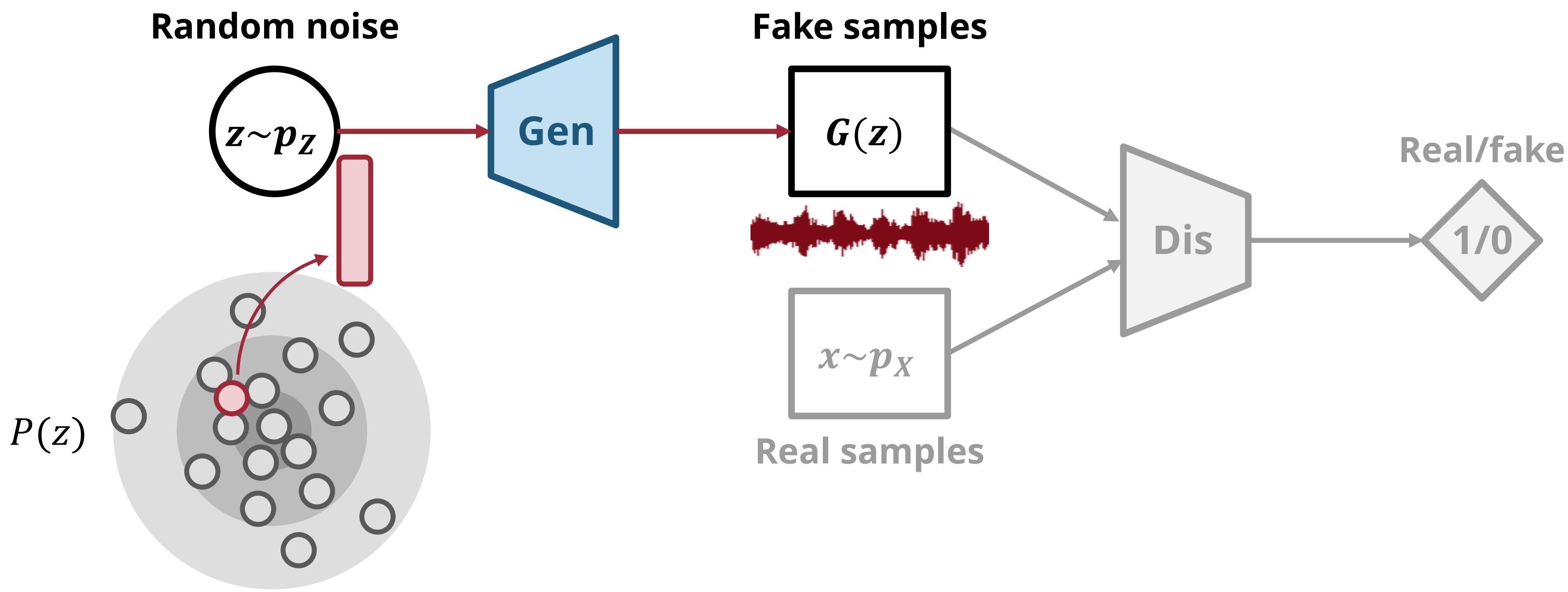
But what about another neural network!?

Generative Adversarial Nets (GANs) – Training



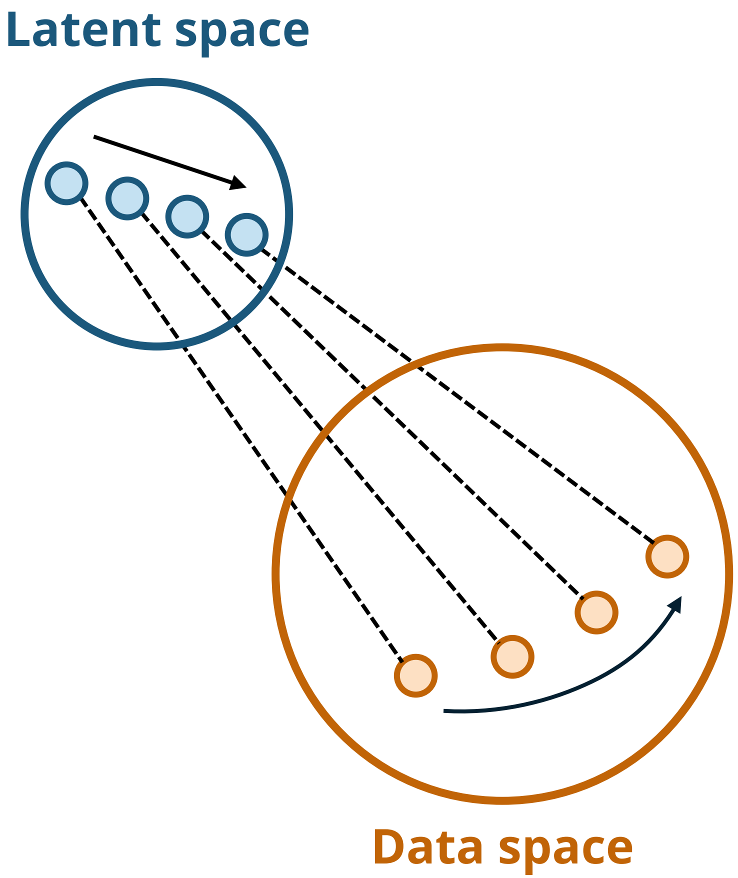
Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative Adversarial Networks," *NeurIPS*, 2014.

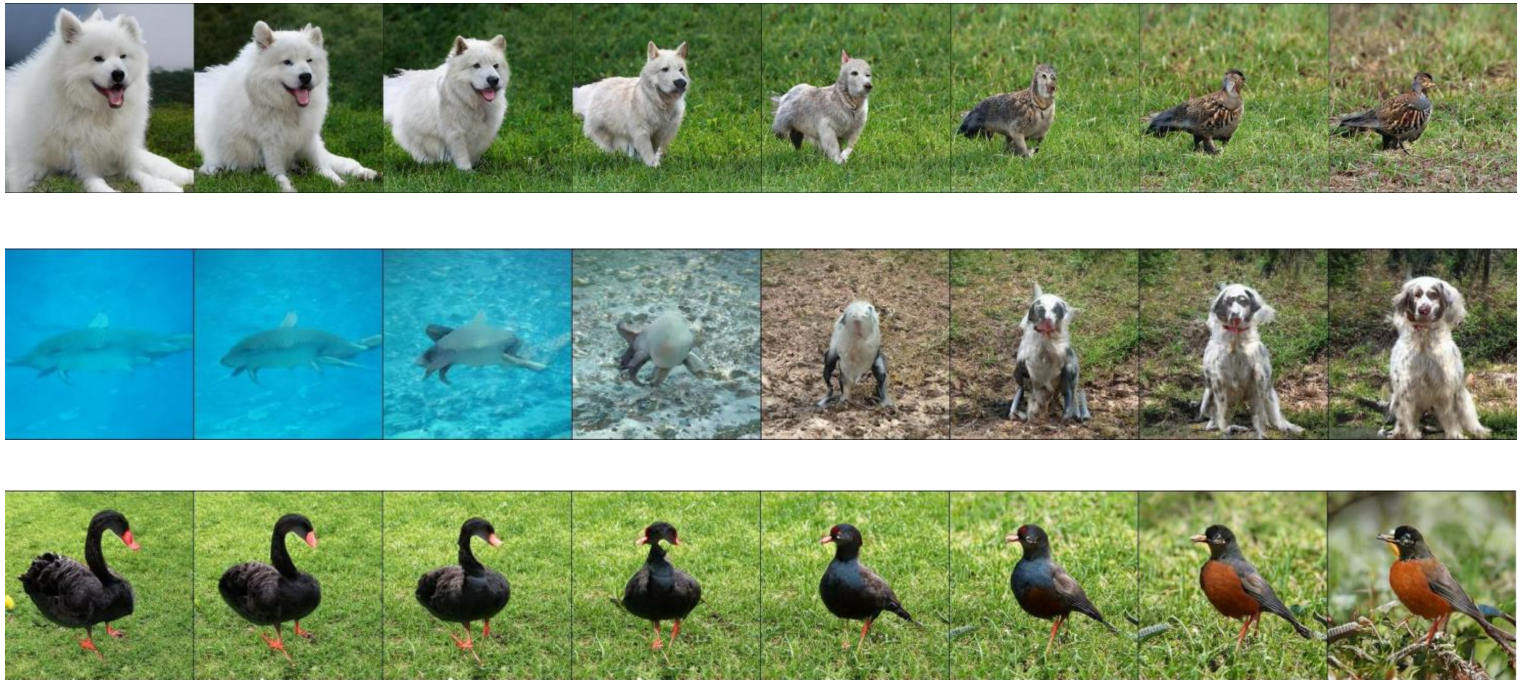
Generative Adversarial Nets (GANs) – Generation



Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative Adversarial Networks," *NeurIPS*, 2014.

Interpolation on the Latent Space

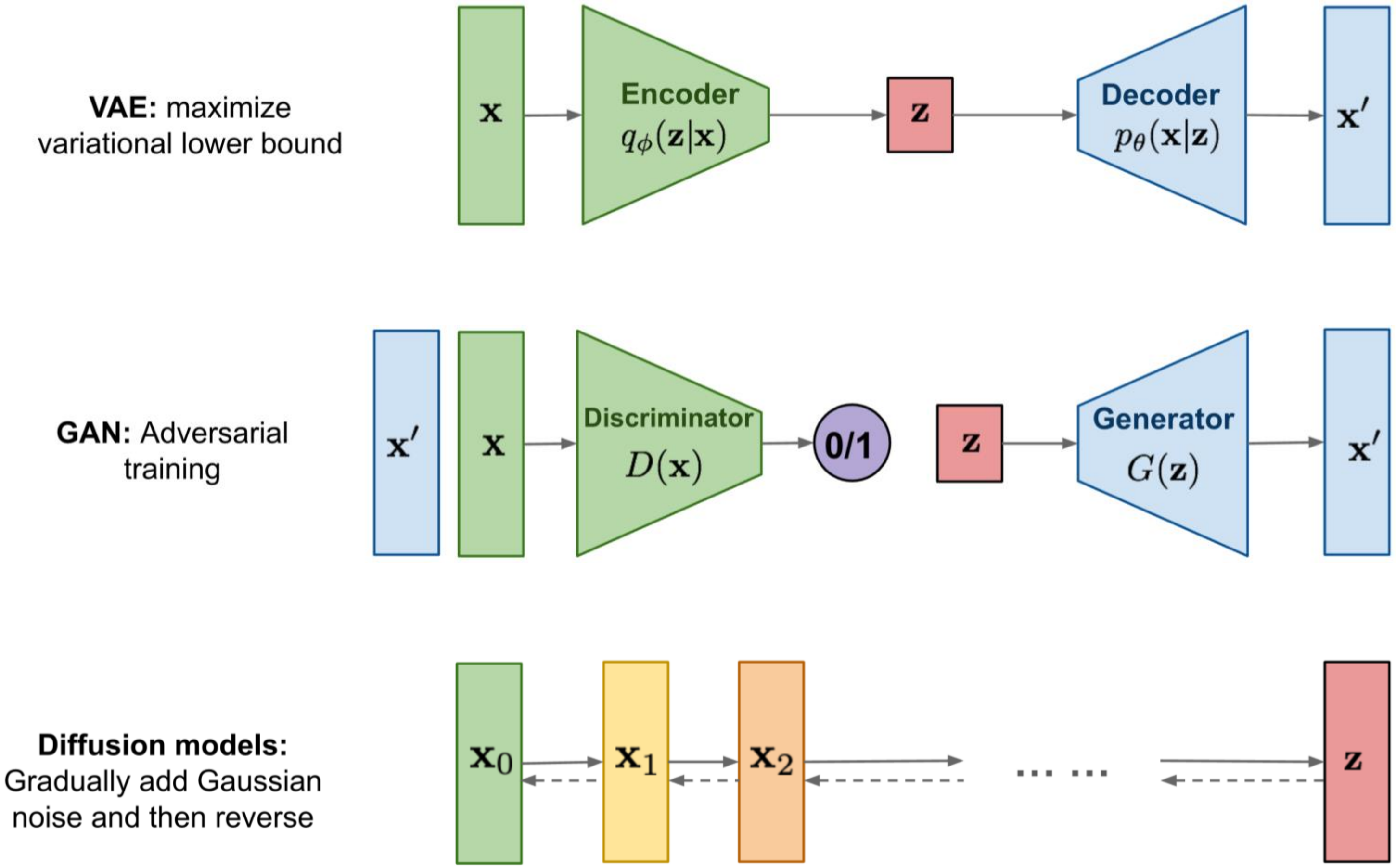




(Source: Brock et al., 2019)

Andrew Brock, Jeff Donahue, and Karen Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," *ICLR*, 2019.

Comparison of Deep Generative Models

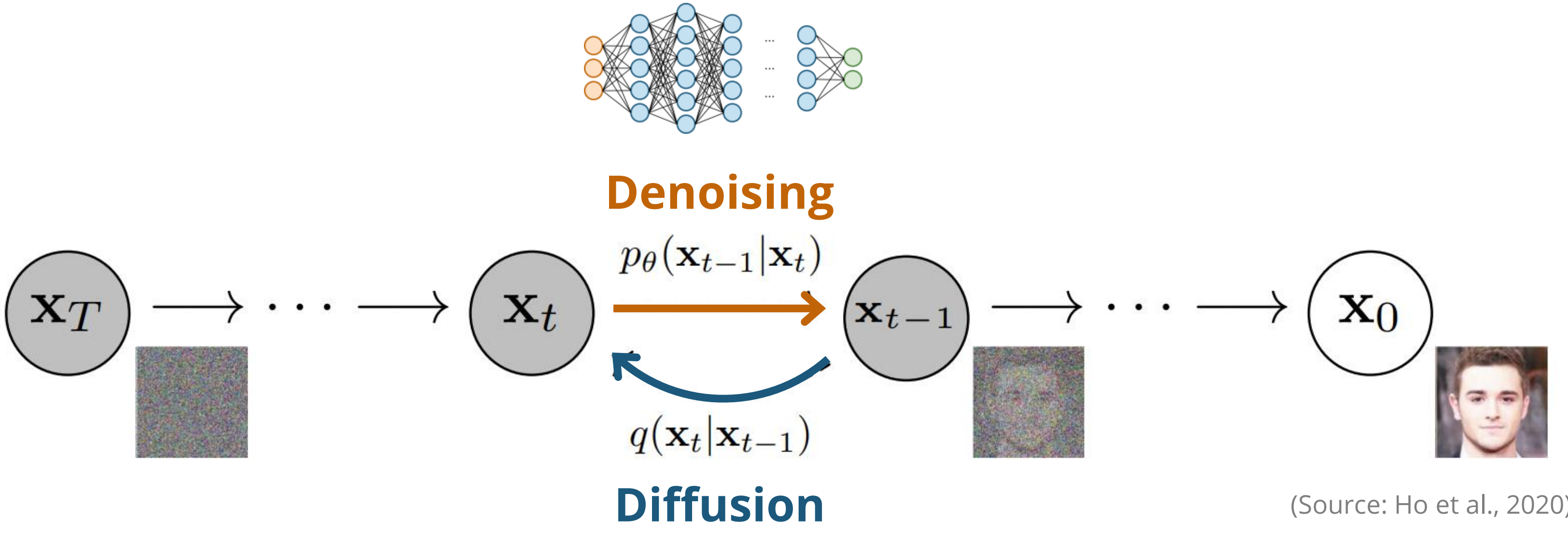


(Source: Weng, 2021)

lilianweng.github.io/posts/2021-07-11-diffusion-models/

Diffusion Models

- **Intuition**: Many denoising autoencoders stacked together

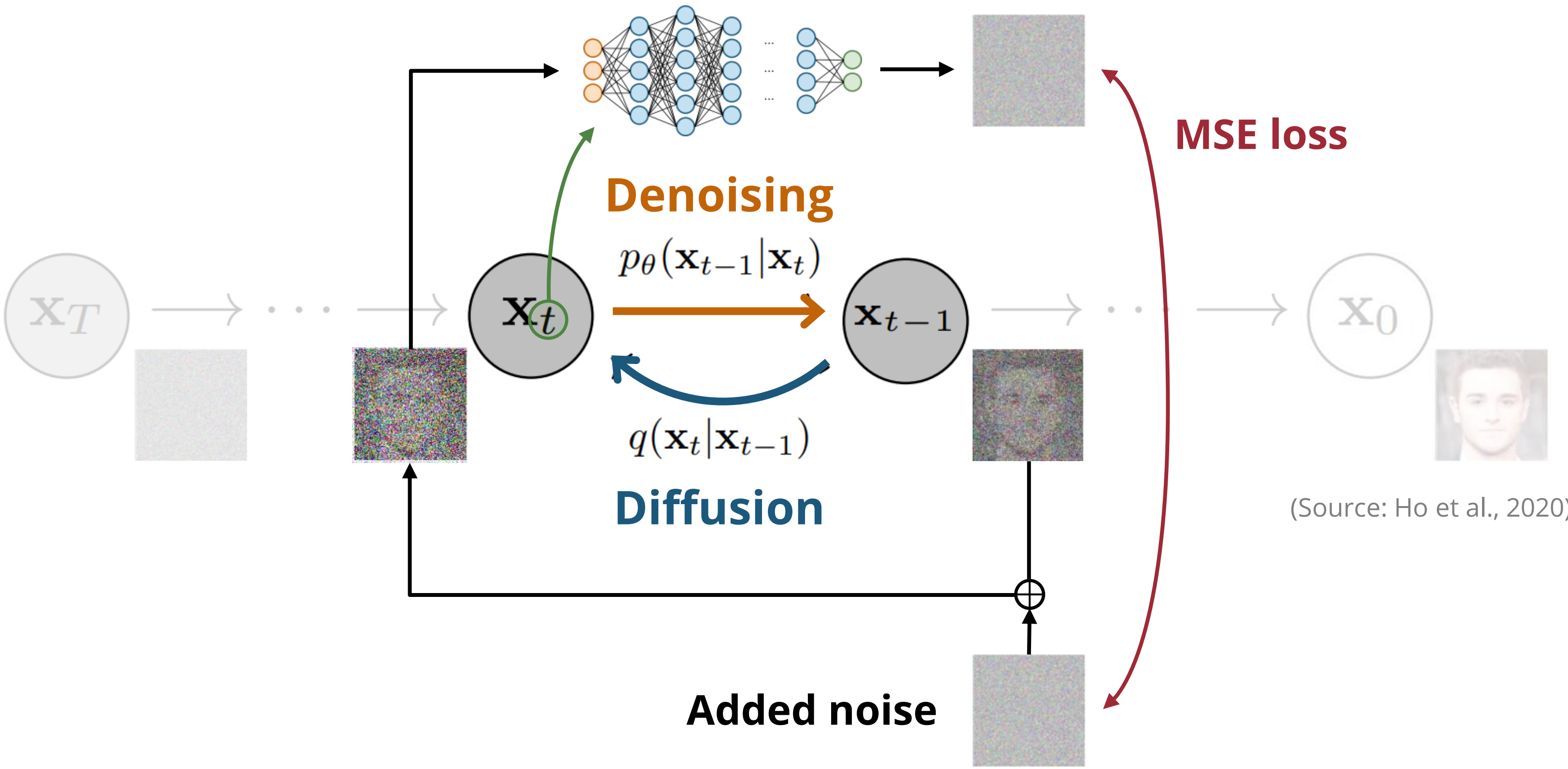


(Source: Ho et al., 2020)

Jonathan Ho, Ajay Jain, and Pieter Abbeel, "Denoising Diffusion Probabilistic Models," *NeurIPS*, 2020.

Diffusion Models – Training

- **Intuition**: Many denoising autoencoders stacked together



(Source: Ho et al., 2020)

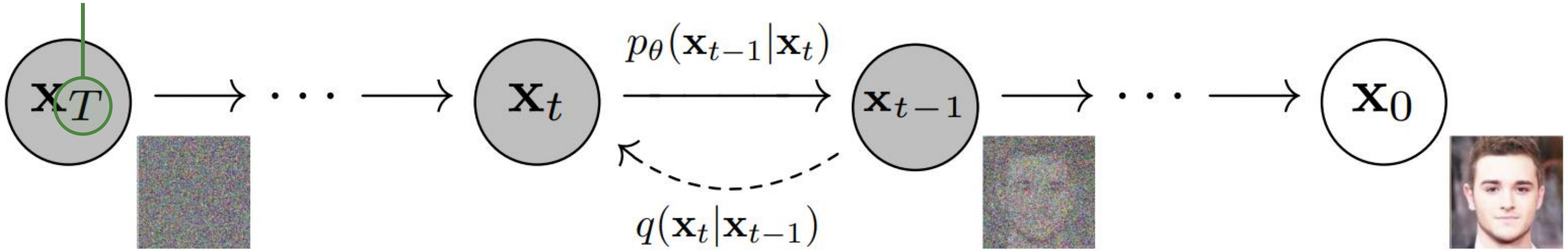
Jonathan Ho, Ajay Jain, and Pieter Abbeel, "Denoising Diffusion Probabilistic Models," *NeurIPS*, 2020.

Diffusion Models

- **Intuition**: Many denoising autoencoders stacked together

Remove noise gradually
(Backward diffusion process)

Usually, $T > 1000$

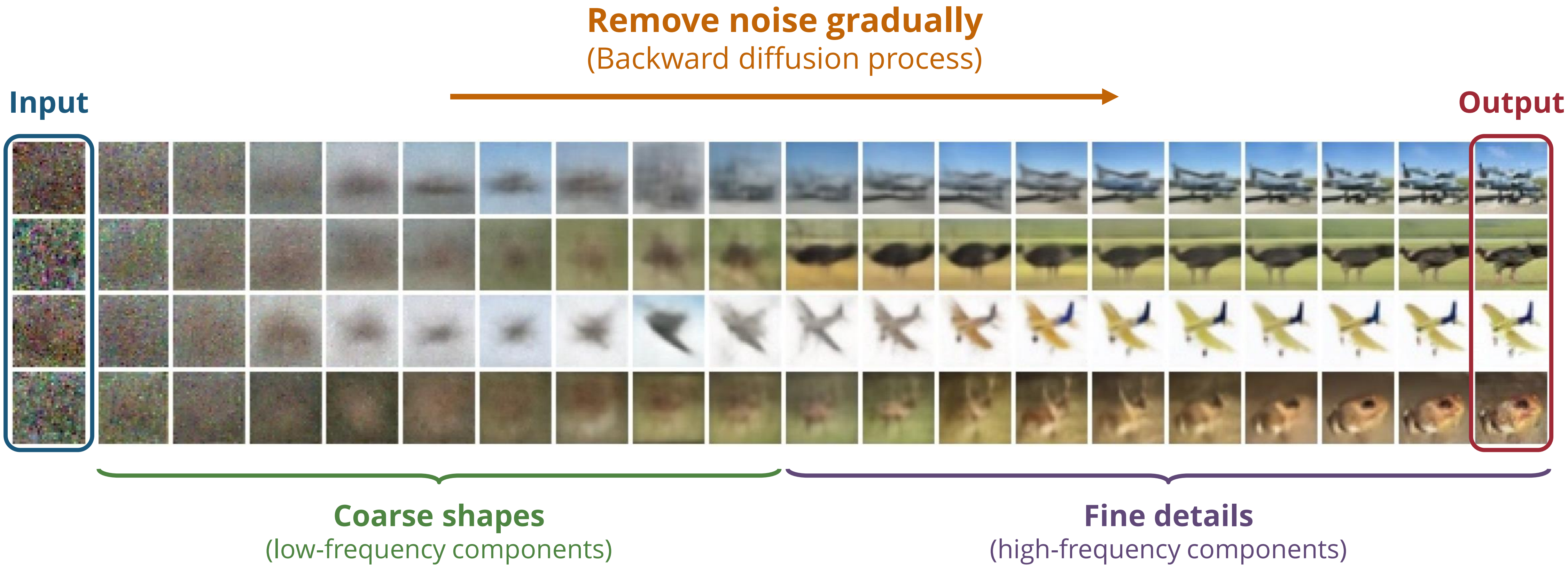


(Source: Ho et al., 2020)

Add noise gradually
(Forward diffusion process)

Jonathan Ho, Ajay Jain, and Pieter Abbeel, "Denoising Diffusion Probabilistic Models," *NeurIPS*, 2020.

Diffusion Models – Generation



(Source: Ho et al., 2020)

Jonathan Ho, Ajay Jain, and Pieter Abbeel, "Denoising Diffusion Probabilistic Models," *NeurIPS*, 2020.

Discussions

Discussions

- **Why** do we need machine learning?
- **When** should we use deep learning?
- **When can't** we use deep learning?

- Should we choose the best model based on validation loss or accuracy?
- Which generative model works best?
- Is overfitting always an issue?

Building Blocks of Modern AI Systems

Data

×

Model

×

Use Case

What we'll talk about next!

What we've been focused on!

What we'll talk about next!

That’s It for the First Half of This Course!