CS 4602

Introduction to Machine Learning

“The Other” Networks and Learning Strategies

Instructor: Po-Chih Kuo

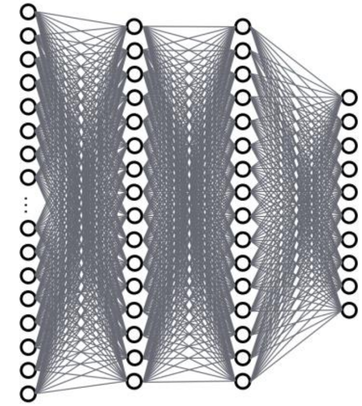
Roadmap

- Introduction and Basic Concepts
- Regression
- Bayesian Classifiers
- Decision Trees
- KNN
- Linear Classifier
- Neural Networks
- Deep learning
- Convolutional Neural Networks
- RNN/Transformer
- Reinforcement Learning
- Model Selection and Evaluation
- Clustering
- Data Exploration & Dimensionality reduction

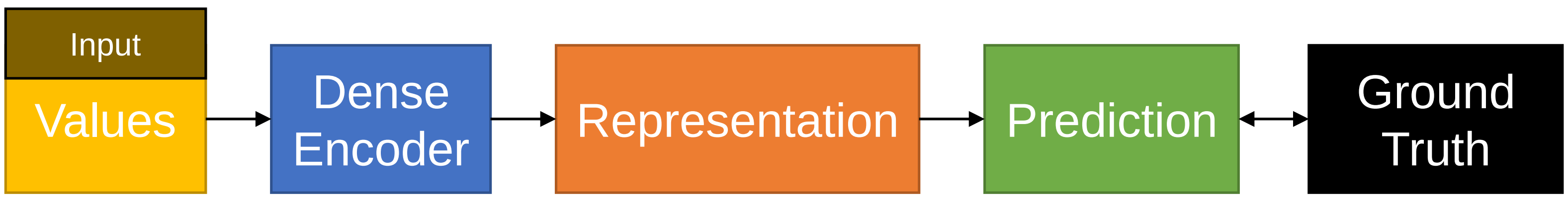
Learning, learning, learning

- Supervised learning
  - **FFNNs**
  - **CNNs**
  - **RNNs**
  - **Encoder Decoder**
- Unsupervised learning
  - **Autoencoder**
  - **GAN**
- Self-Supervised learning
- Reinforcement learning
- Transfer learning
- Federated learning

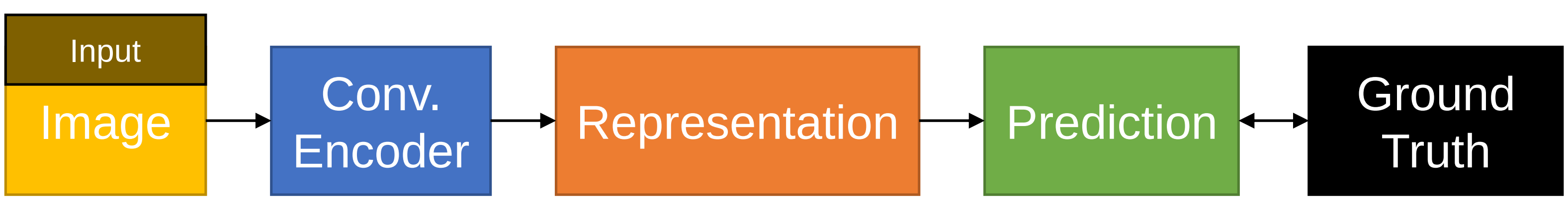
Supervised learning



- **Feed Forward Neural Networks (FFNNs)** - classification and regression based on features.

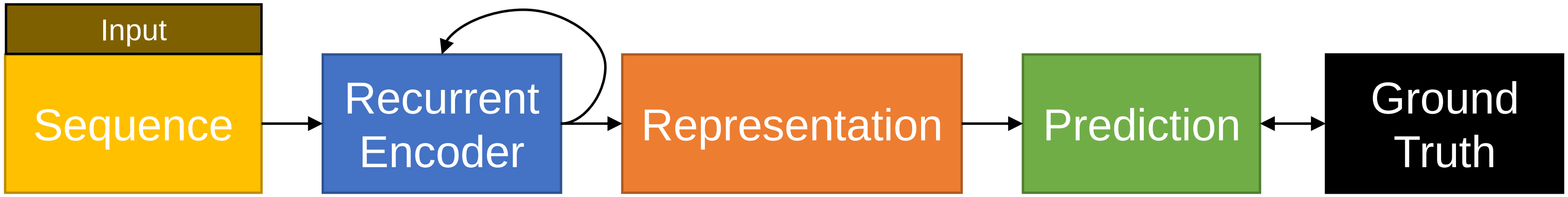


- **Convolutional Neural Networks (CNNs)** - image classification, object detection, video action recognition



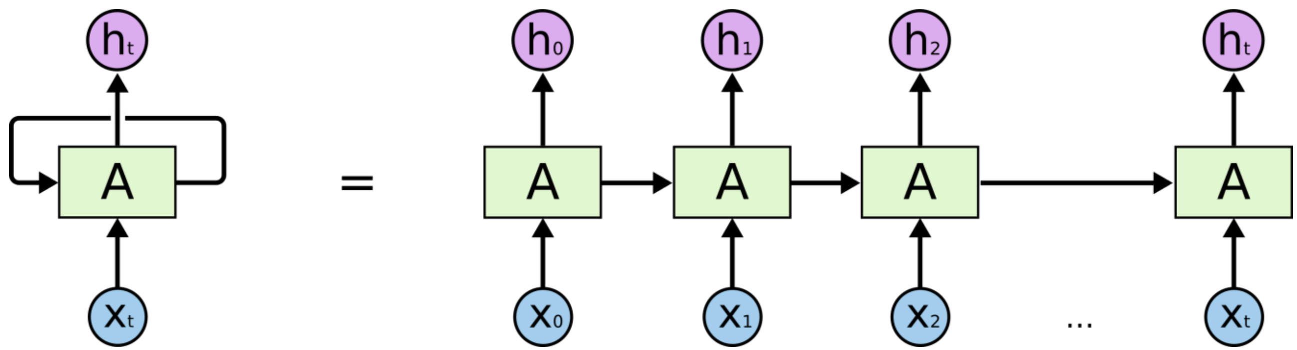
Supervised learning (cont.)

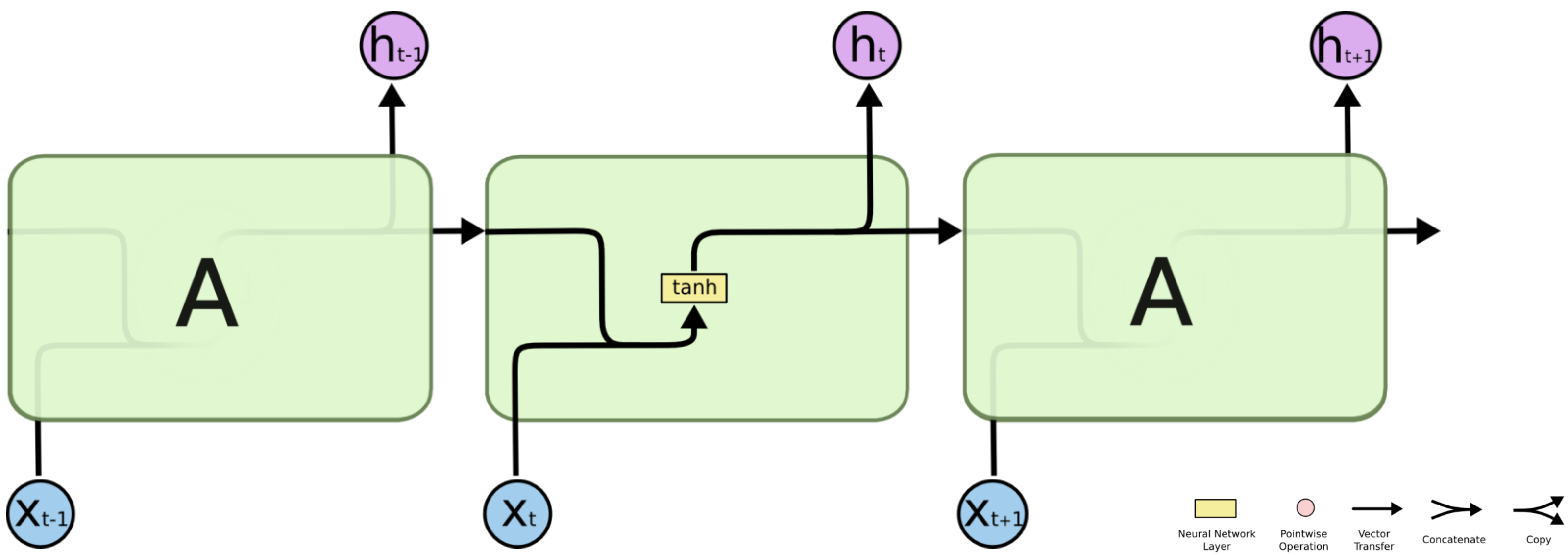
- **Recurrent Neural Networks (RNNs)** - language modeling, speech recognition/generation



- **Encoder Decoder Architectures** - semantic segmentation, machine translation

Recurrent Neural Network (RNN)

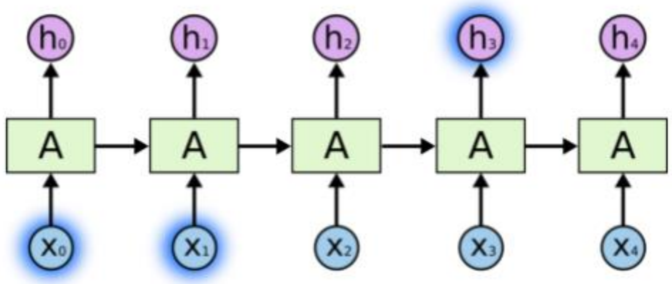




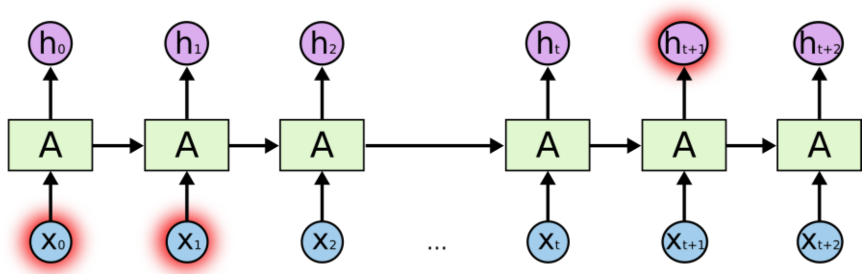
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-Short Term Dependencies

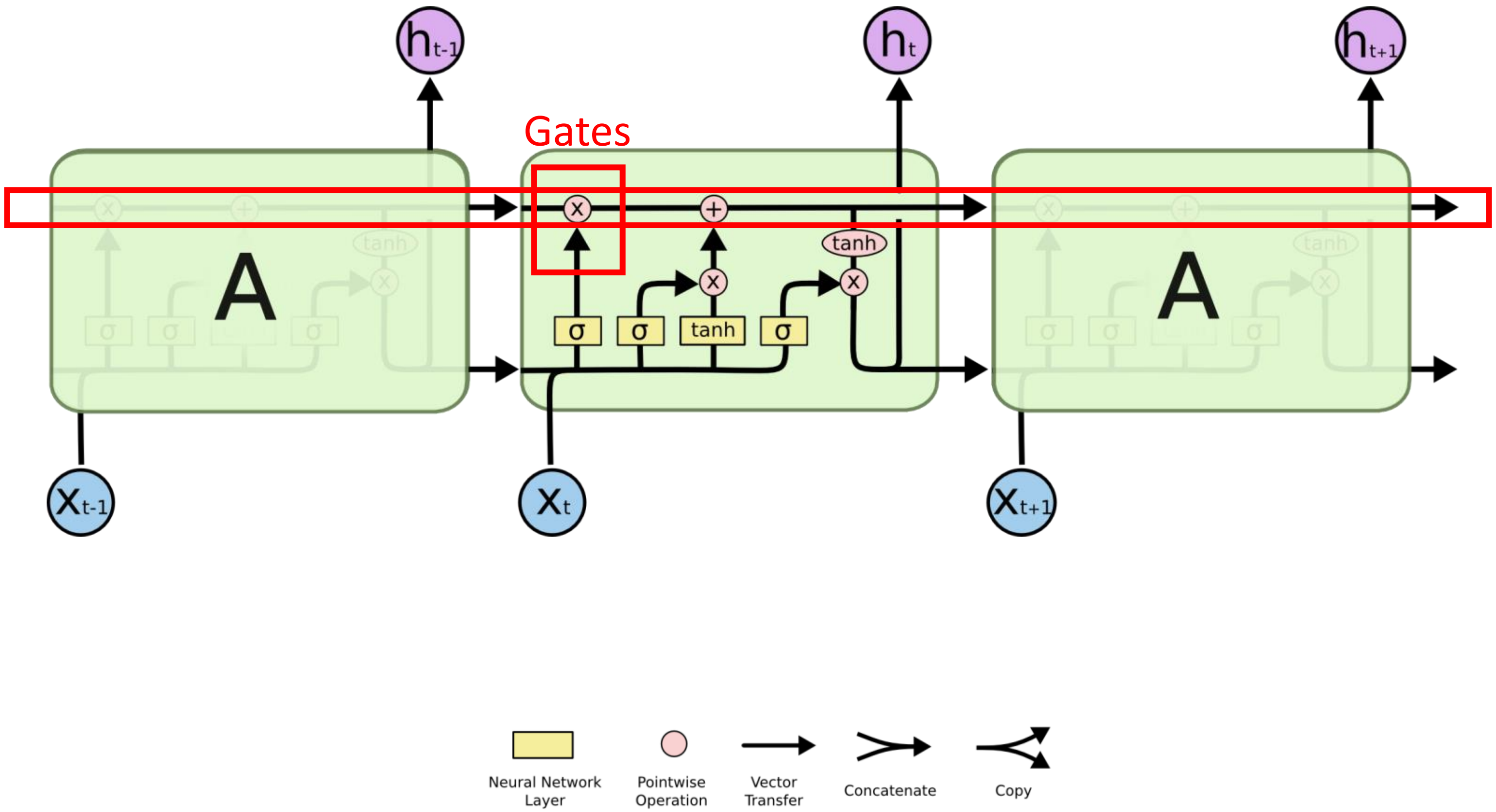
- Short-Term

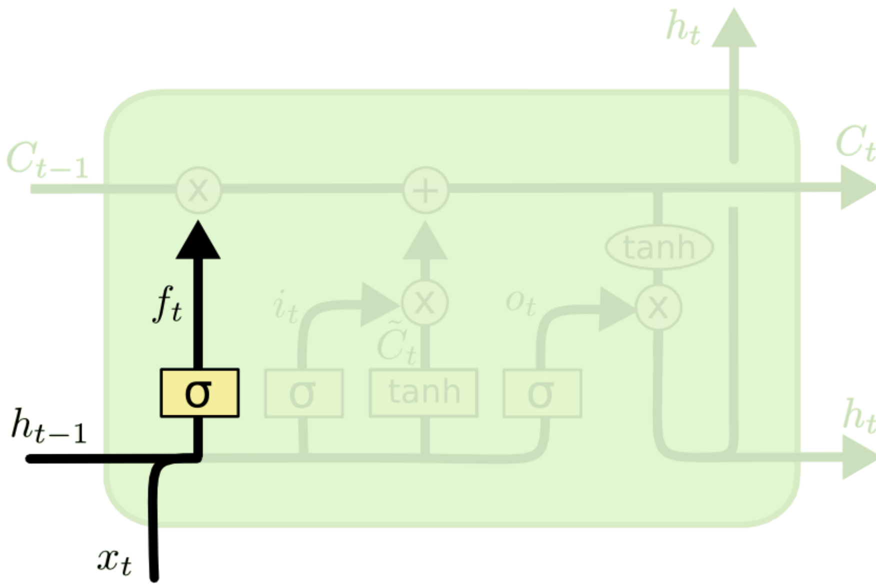


- Long-Term

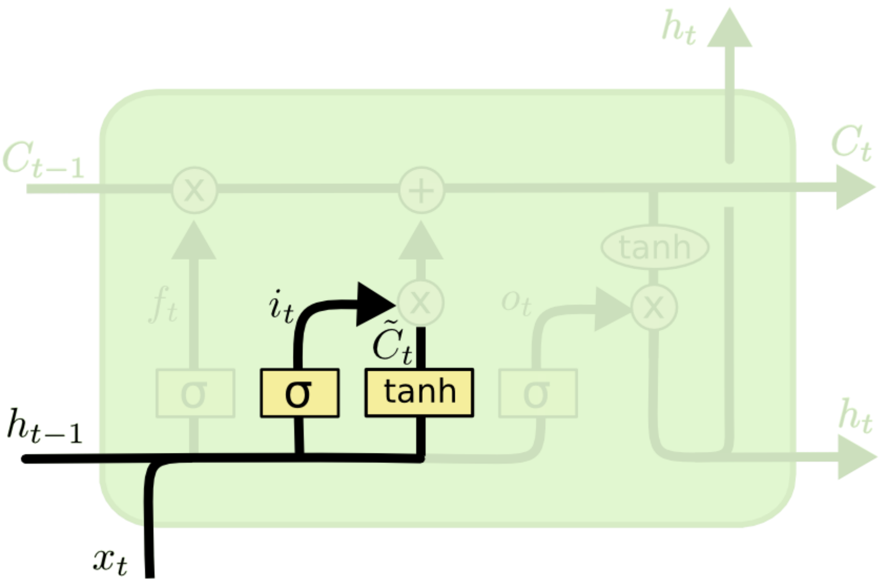


Long-Short Term Memory (LSTM)

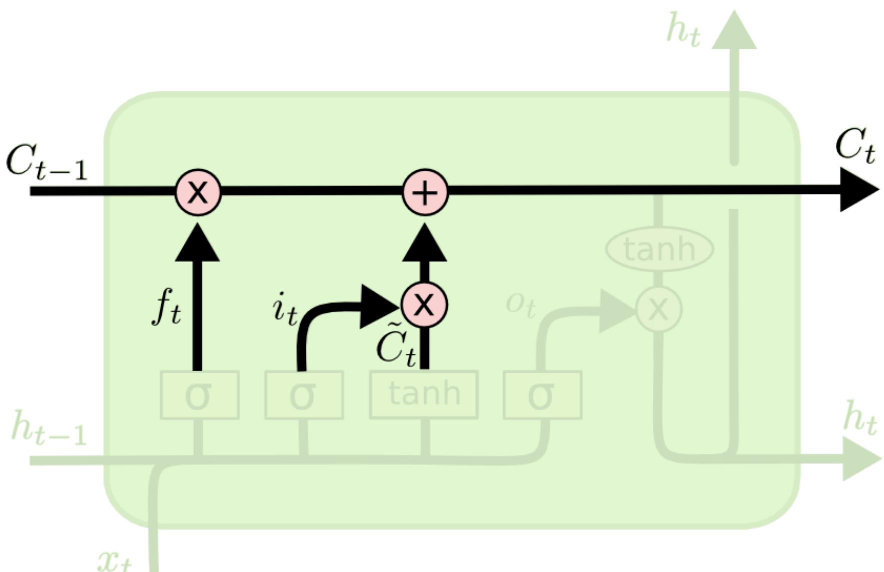




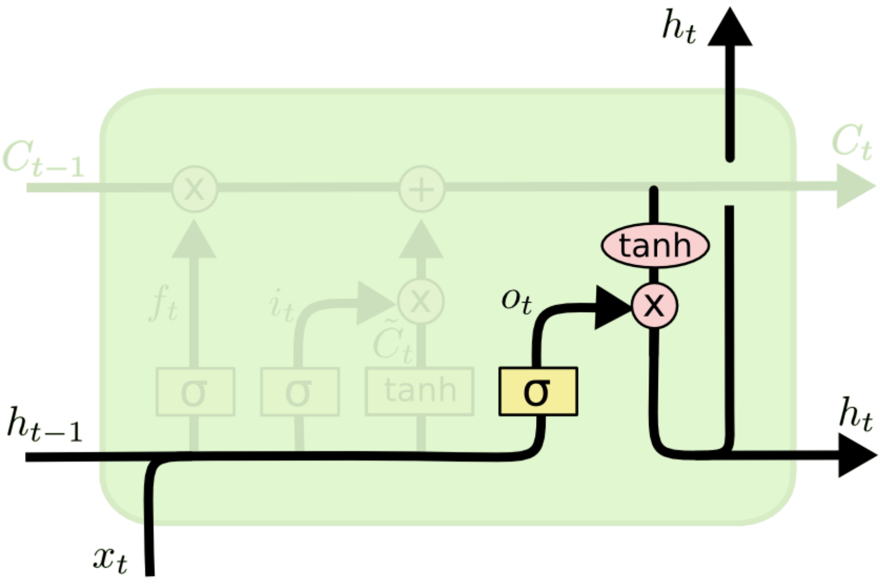
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



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



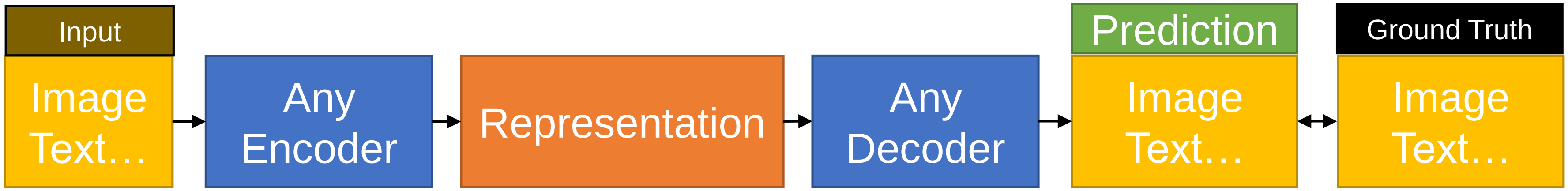
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

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

Supervised learning (cont.)

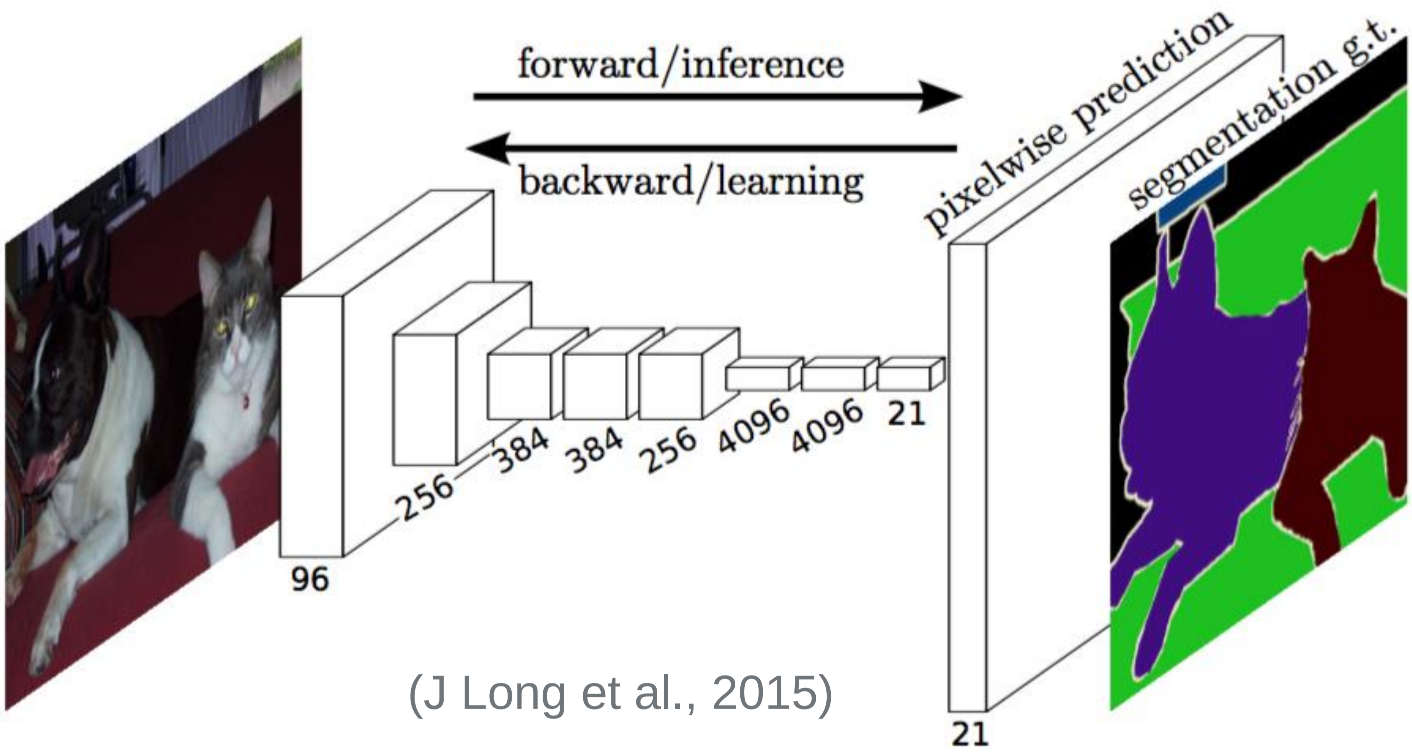
- **Recurrent Neural Networks (RNNs)** - language modeling, speech recognition/generation

- **Encoder Decoder Architectures** - semantic segmentation, machine translation



Encoder Decoder

- Fully Convolutional Networks for Semantic Segmentation



(J Long et al., 2015)

Transformer

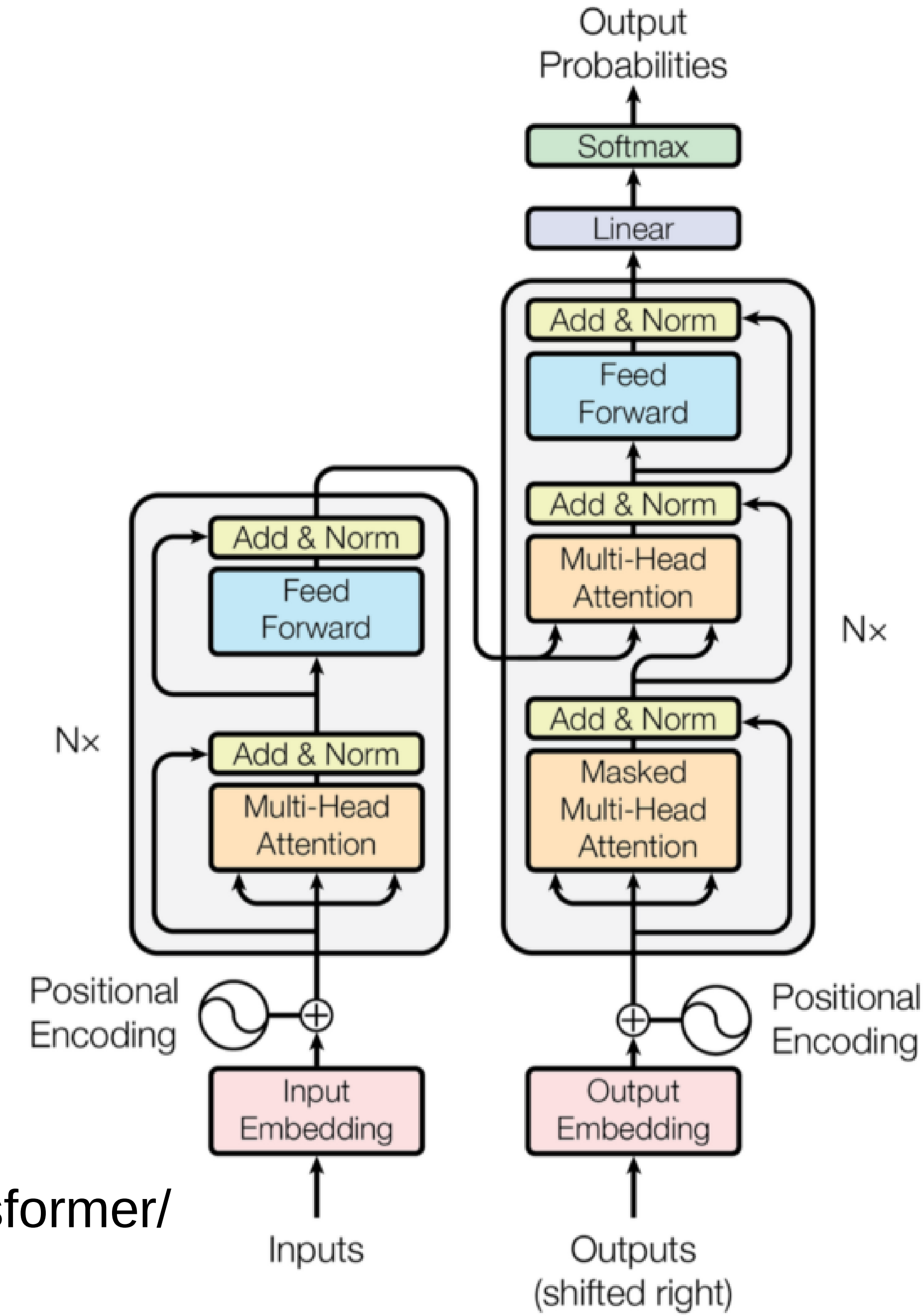


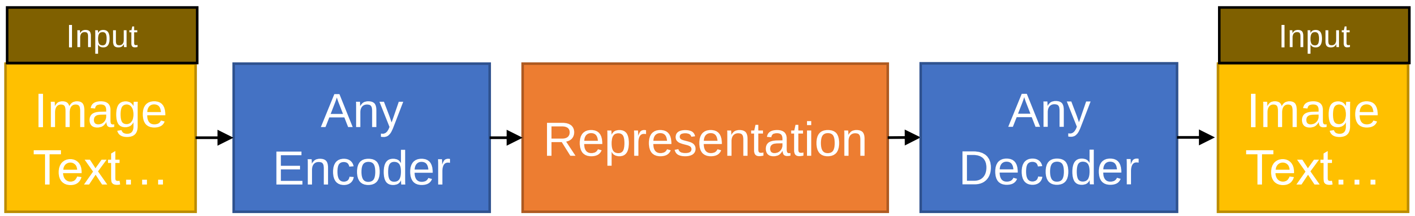
Figure 1: The Transformer - model architecture.

- Further reading: https://theaisummer.com/transformer/

"Attention Is All You Need"

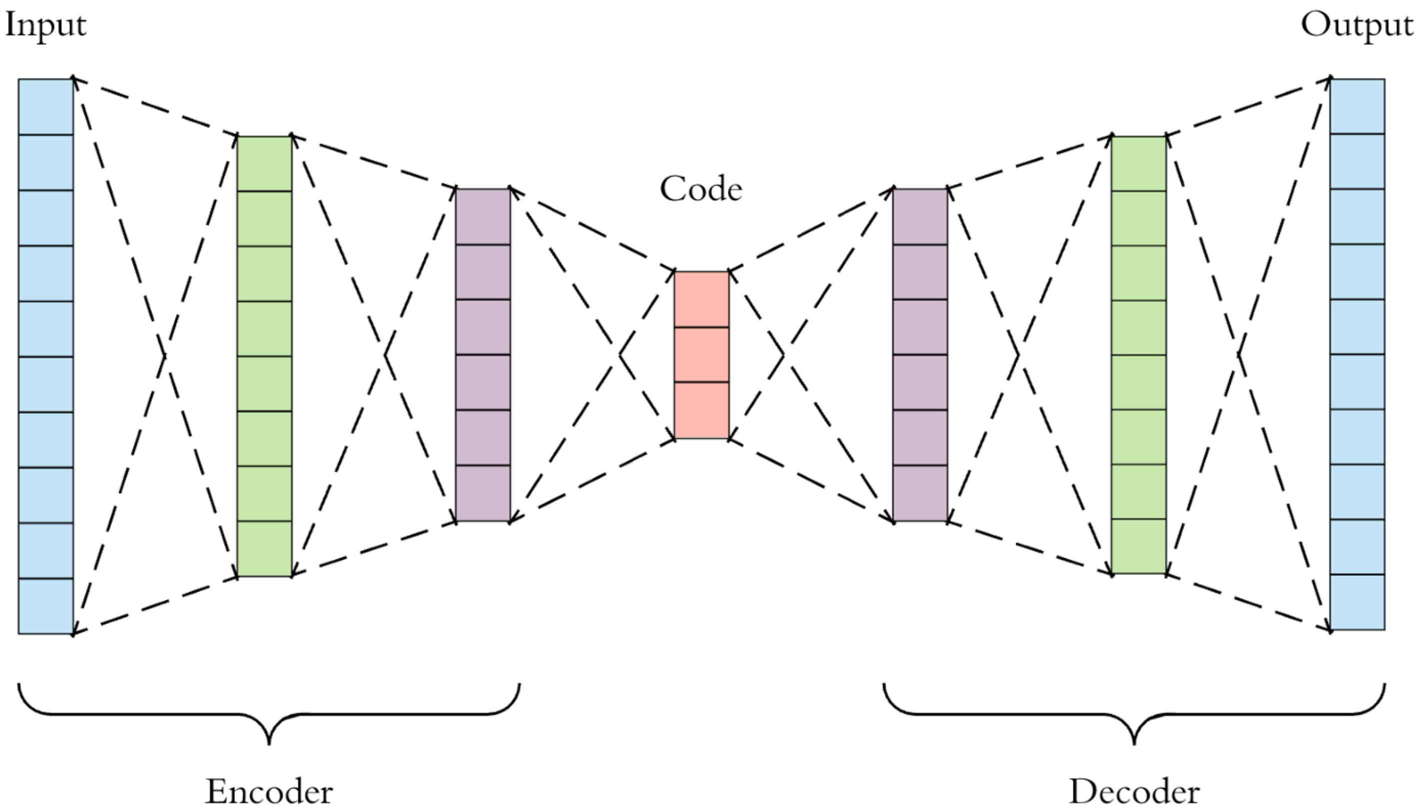
Unsupervised learning

- **Autoencoder** - unsupervised embeddings, denoising

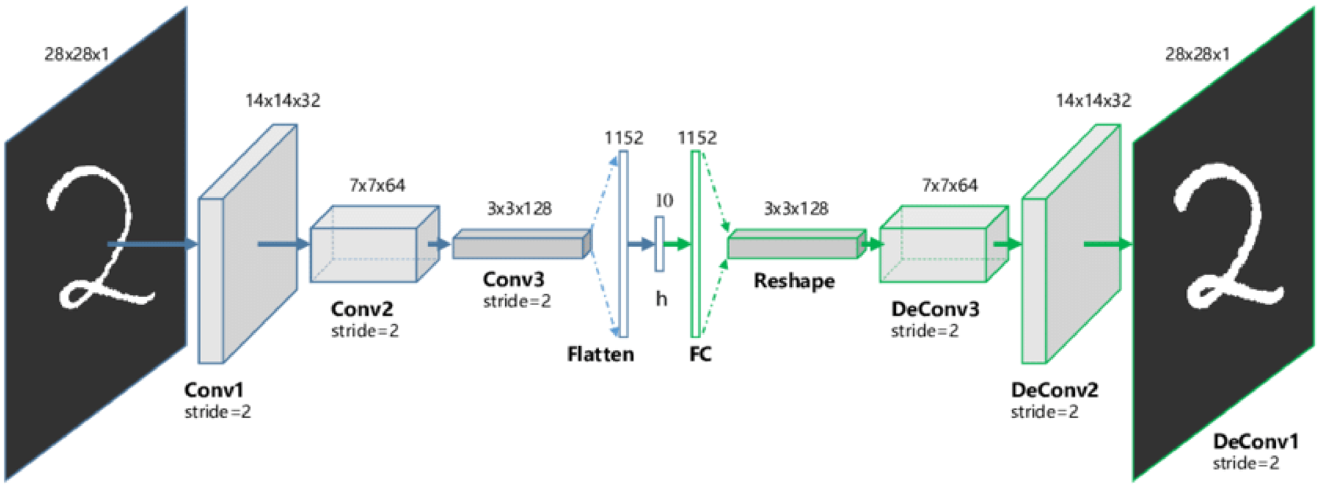


- **Generative Adversarial Networks (GANs)** - generation of realistic images

AutoEncoder

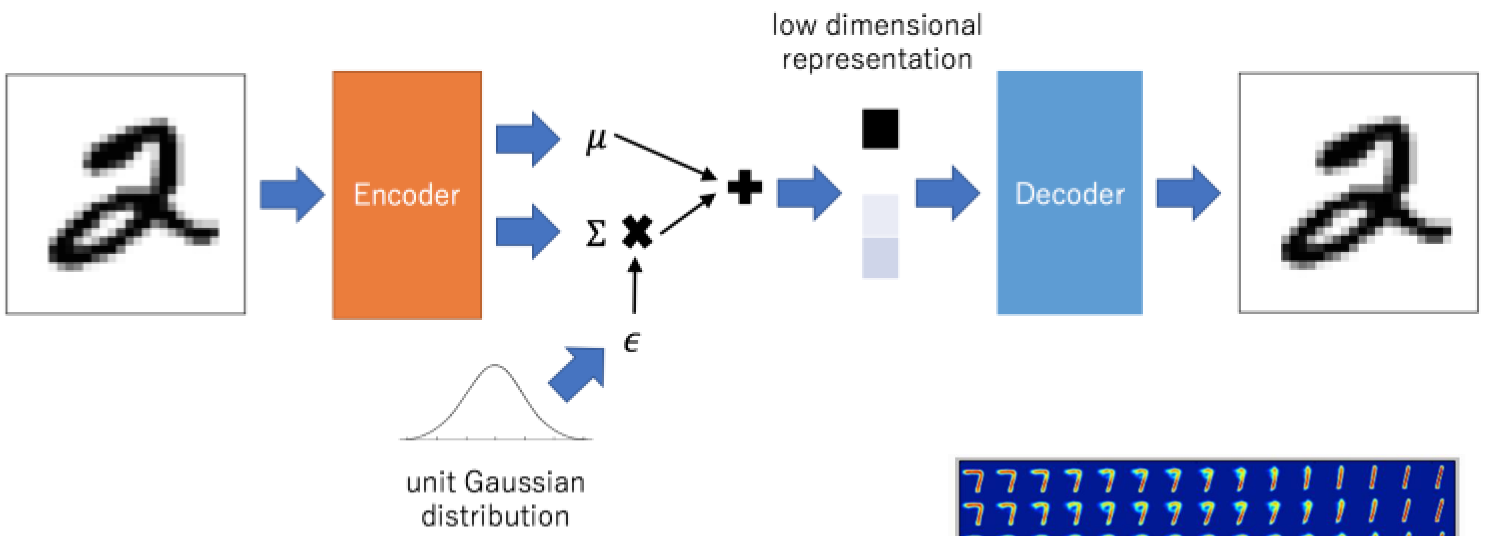


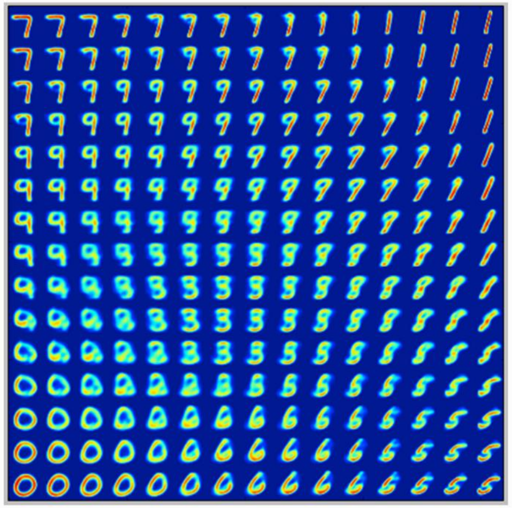
Convolutional AutoEncoder (CAE)



(Guo et al., 2017)

Variational AutoEncoder (VAE)

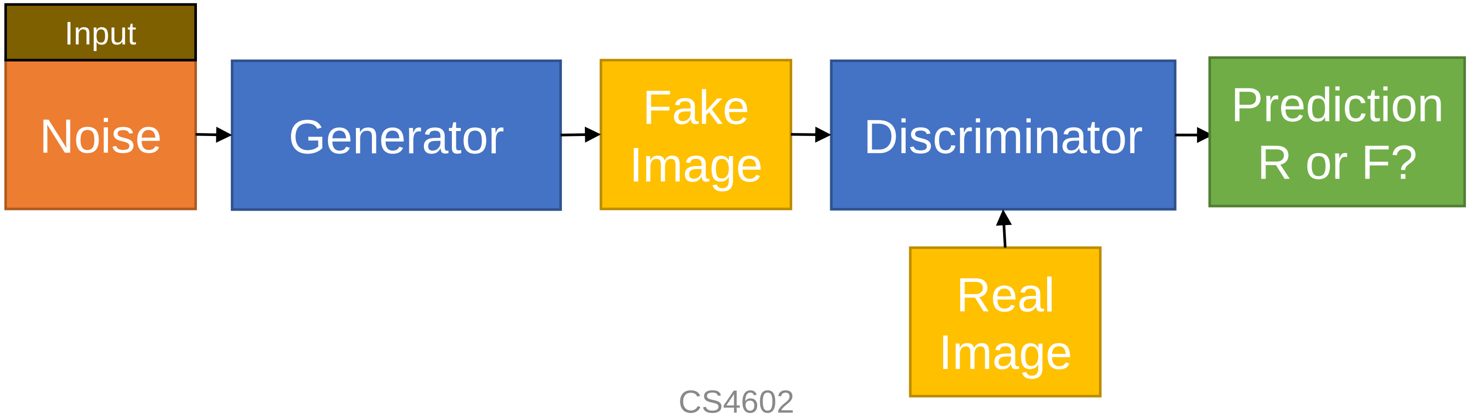




Unsupervised learning

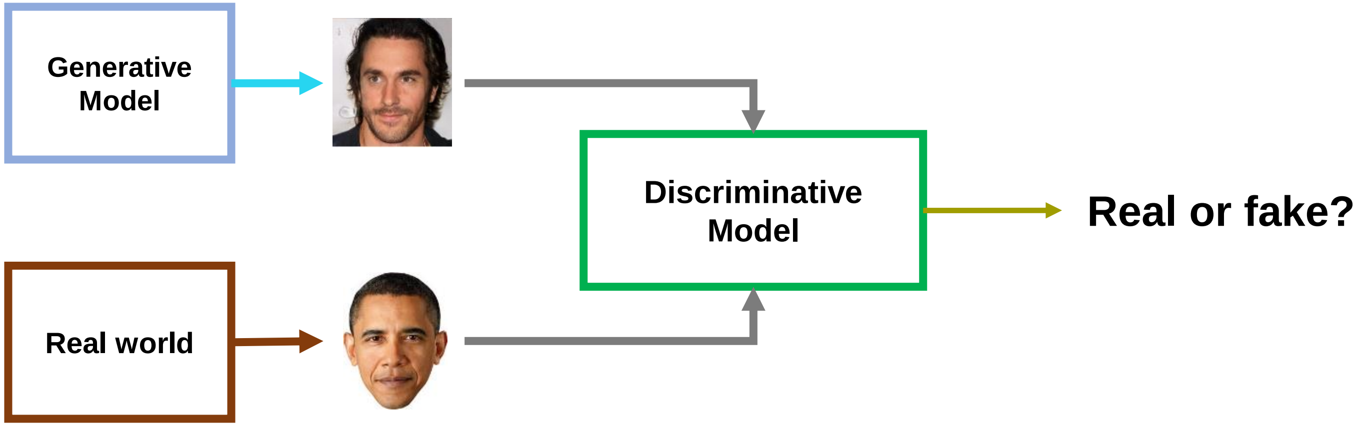
- **Autoencoder** - unsupervised embeddings, denoising

- **Generative Adversarial Networks (GANs)** - generation of realistic images

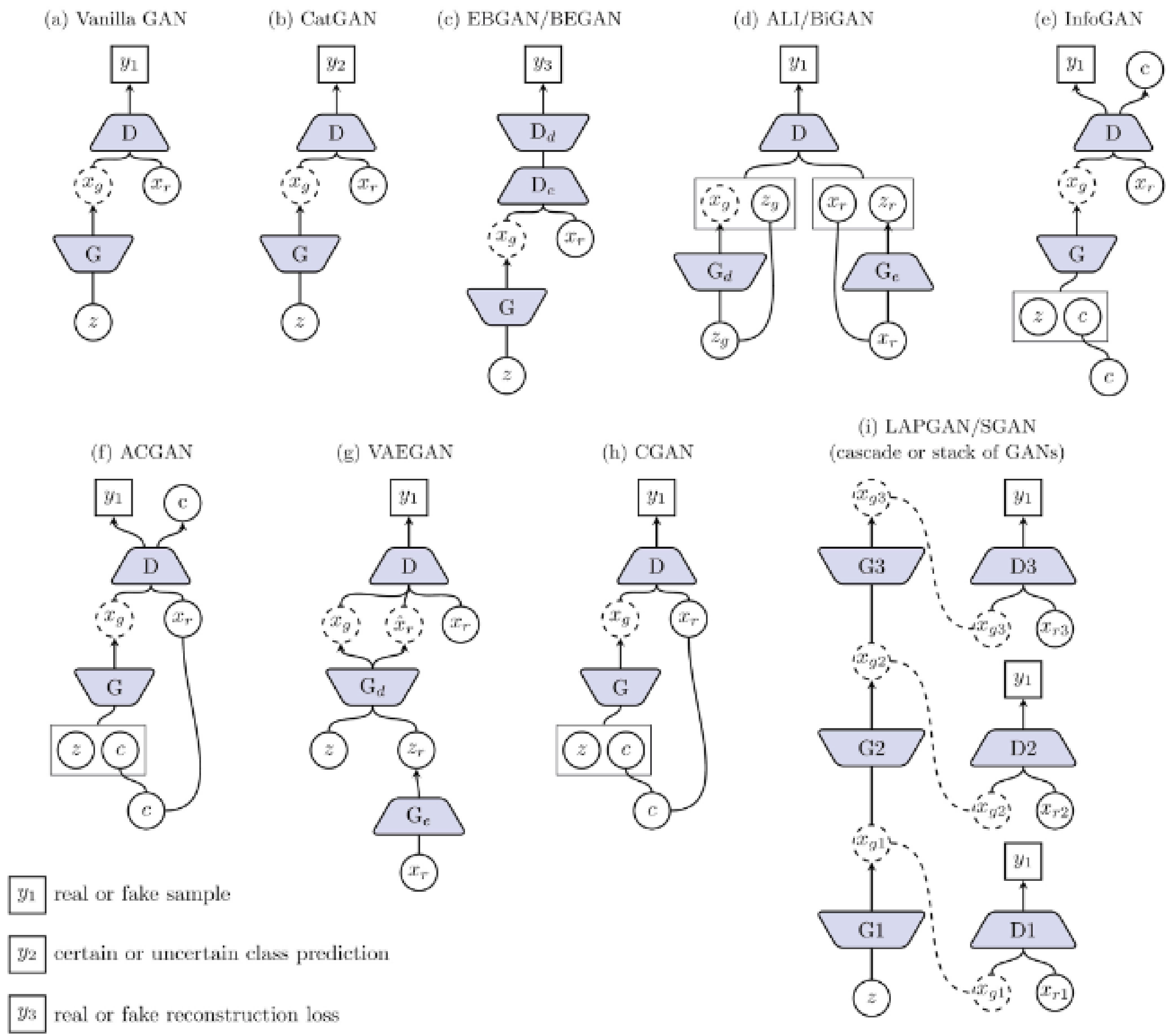


Generative Adversarial Networks (GAN)

- **GAN is the most interesting idea in the last ten years in machine learning.—Yann LeCun, Director, Facebook AI**



(Karras et al., 2018)

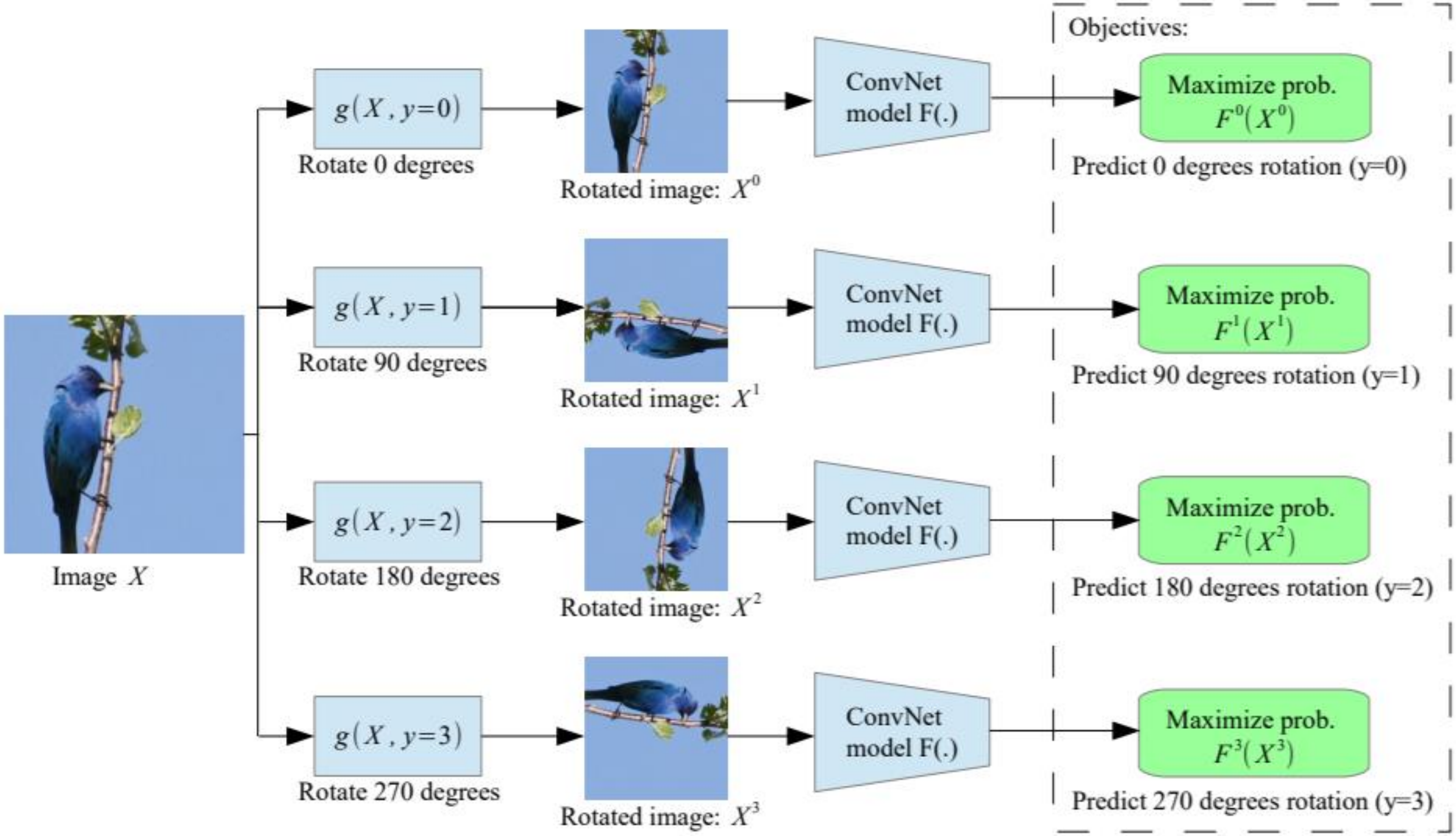


“Generative adversarial network in medical imaging: A review”, 2019

Self-Supervised Learning (SSL)

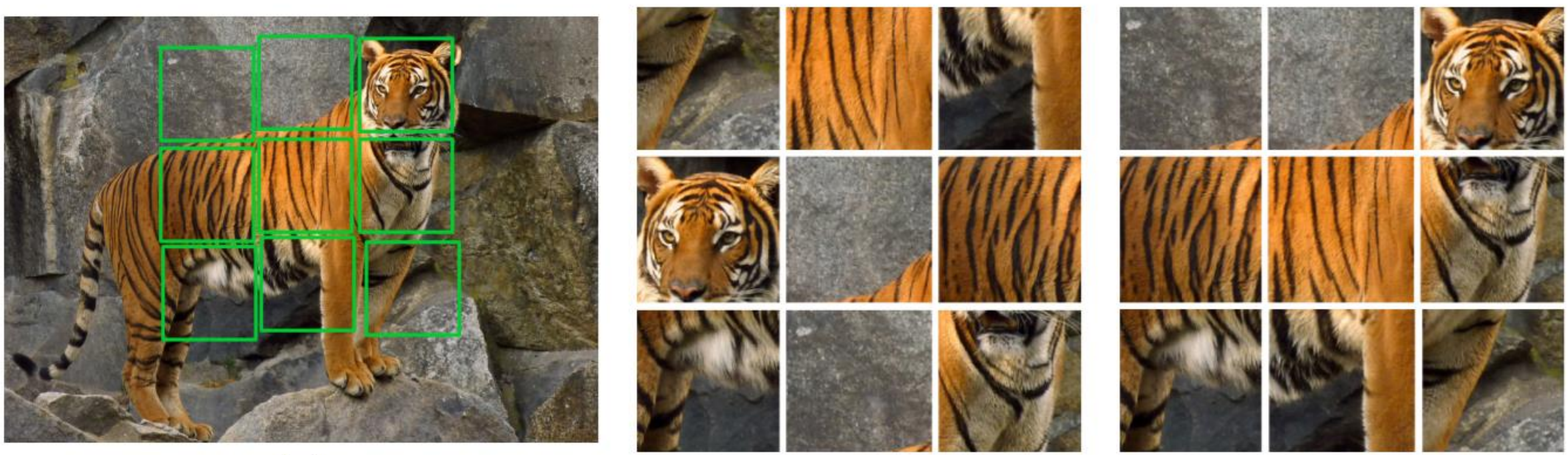
- A representation learning method where a supervised task is created out of the unlabelled data.
- Self-supervised learning is used to reduce the data labelling cost and leverage the unlabelled data pool.
- Some of the popular methods:
  - Rotation
  - Jigsaw Puzzle
  - Colorization

Rotation



Unsupervised Representation Learning by Predicting Image Rotations https://arxiv.org/pdf/1803.07728.pdf

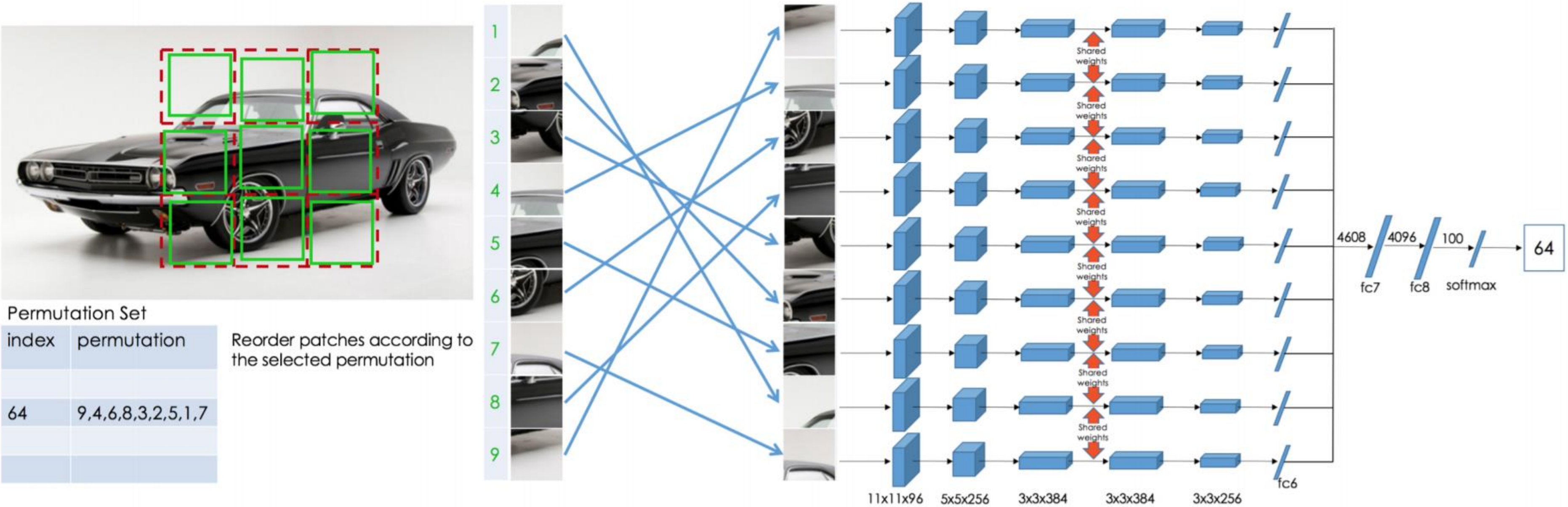
Jigsaw Puzzle



(a) (b) (c)

Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles https://arxiv.org/abs/1603.09246

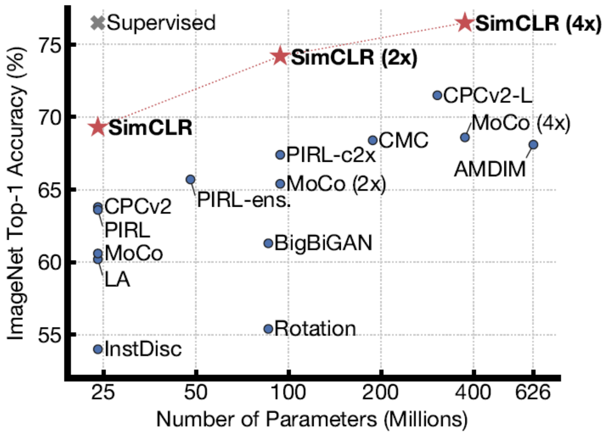
Learning the right positions

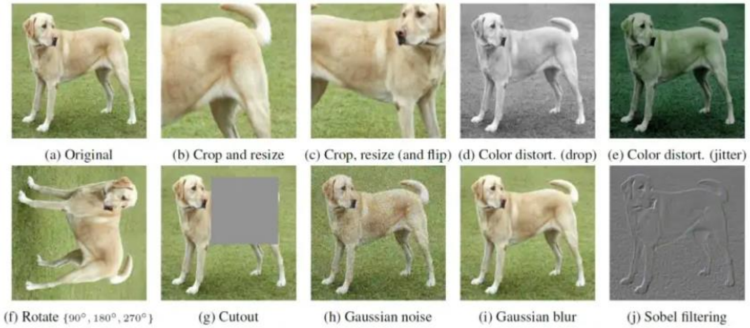


Given 9 tiles, there are $9! = 362{,}880$ possible permutations.

SSL + Contrastive Learning

- SimCLR: A Simple Framework for Contrastive Learning of Visual Representations





GIF for SimCLR (from Google AI Blog)

Reinforcement Learning

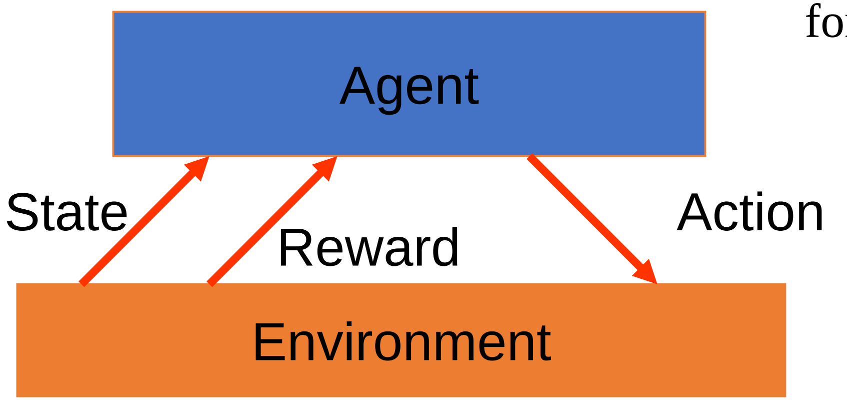
- **Deep Reinforcement Learning** - game playing, robotics in simulation, self-play, neural architecture search

Reinforcement Learning

- Supervised Learning: Immediate feedback (labels provided for every input).
- Unsupervised Learning: No feedback (no labels provided).
- Reinforcement Learning: Delayed scalar feedback (a number called reward).
- RL deals with agents that must sense & act upon their environment.
- Examples:
  - A robot cleaning the room and recharging its battery
  - Robot-soccer
  - Modeling the economy through rational agents
  - Learning how to fly a helicopter
  - Scheduling planes to their destinations

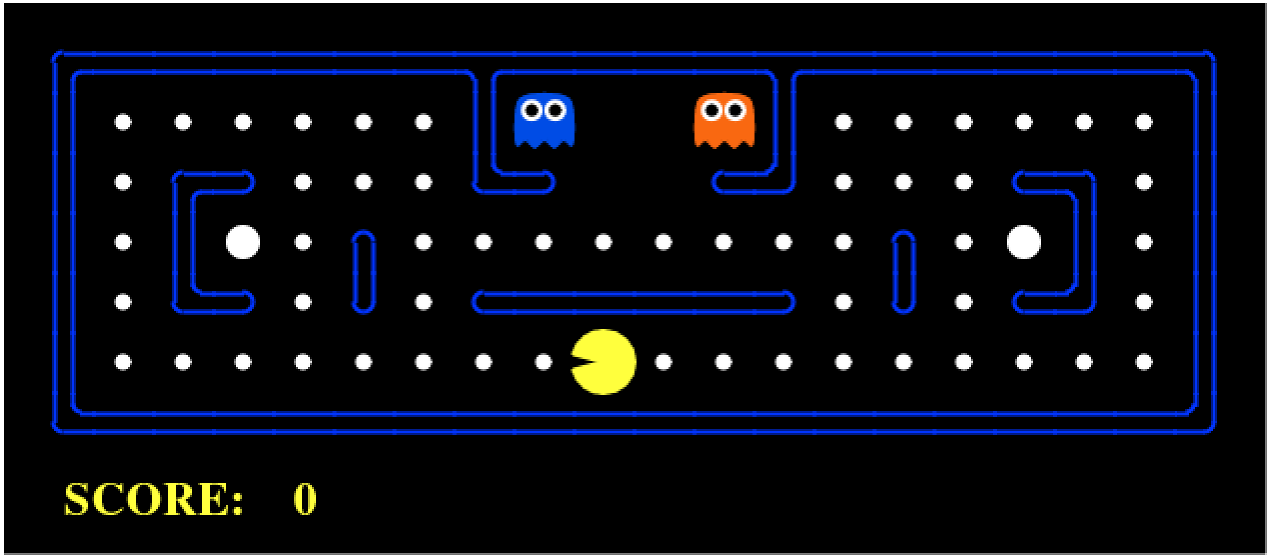
The Big Picture

- S– set of states
- A– set of actions
- T(s,a,s’) = P(s’|s,a)– the probability of transition from *s* to *s’* given action *a*
- R(s,a)– the expected reward for taking action *a* in state *s*



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r1]{a1} s2 \xrightarrow[r2]{a2} s3$$

Your action influences the state of the world which determines its reward


SCORE: 0

The Task

- To learn an optimal *policy* that maps states of the world to actions of the agent.
  I.e., if this patch of room is dirty, I clean it. If my battery is empty, I recharge it.

$$\pi : S \to A$$

- What is it that the agent tries to optimize?
  - the total discounted future reward:

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$$
$$= \sum_{i=0}^{\infty} \gamma^i r_{t+i}, 0 \le \gamma < 1$$

Note: immediate reward is worth more than future reward.

Value Function

- Let's say we have access to the optimal value function that computes the total future reward $V^*(s)$

- The optimal policy $\pi^*(s)$ is chosen by maximizing:

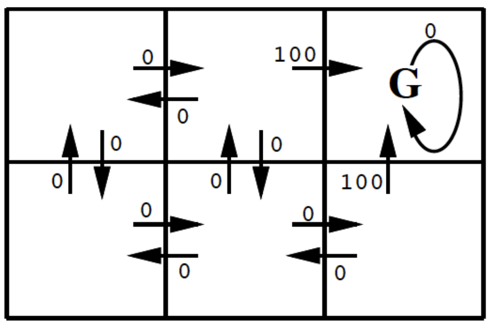
$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \left[r(s,a) + \gamma V^*(\delta(s,a)) \right]$$

- We assume that we know what the reward will be if we perform action "a" in state "s": $r(s,a)$

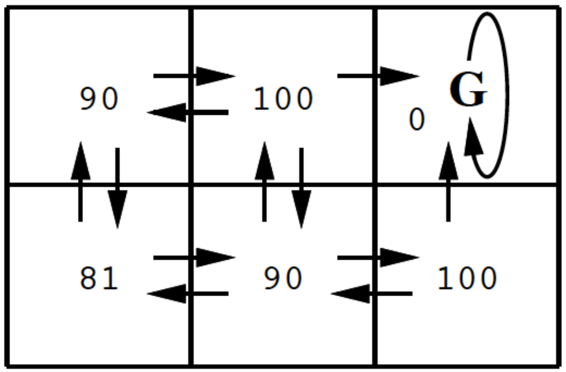
- We also assume we know what the next state of the world will be if we perform action "a" in state "s":

$$s_{t+1} = \delta(s_t, a)$$

Example: Find your way to G

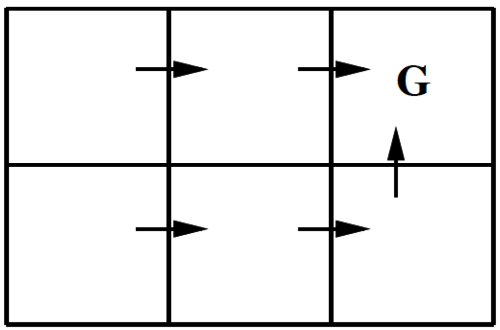


$r(s, a)$ (immediate reward) values



$V^*(s)$ values

$\gamma = 0.9$



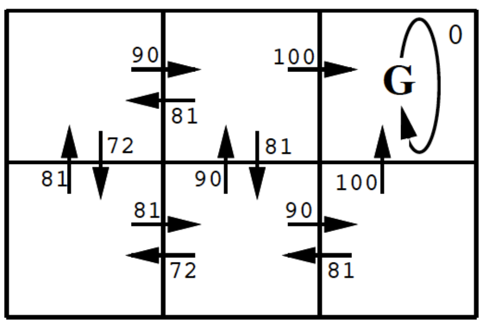
One optimal policy

Q-Function

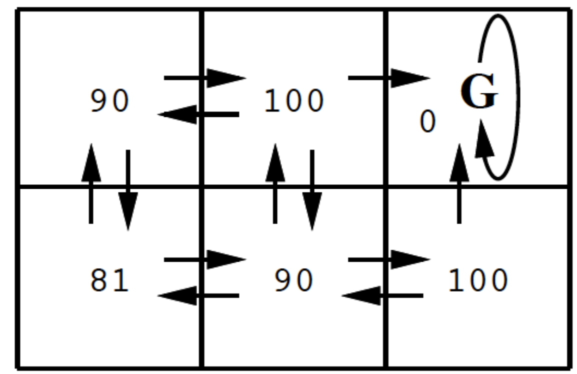
- One approach to RL to estimate V*(s). $V^*(s) \leftarrow \max_a \left[r(s,a) + \gamma V^*(\delta(s,a)) \right]$
- However, this approach requires you to know r(s,a) and δ(s,a).
- We need a function that directly learns good state-action pairs,
- i.e. what action should I take in this state. We call this Q(s,a).
- Given Q(s,a) it is now trivial to execute the optimal policy, *without knowing r(s,a) and δ(s,a).* We have:

$$\pi^*(s) = \arg\max_a Q(s,a)$$

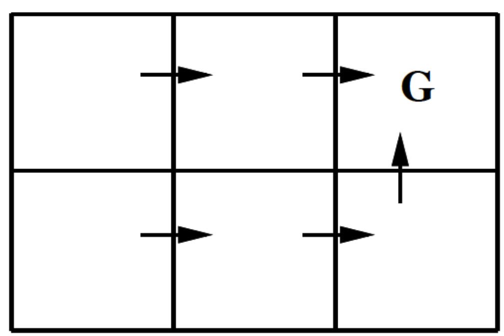
$$V^*(s) = \max_a Q(s,a)$$



$Q(s,a)$ values



$V^*(s)$ values



One optimal policy

$$\pi^*(s) = \arg\max_a Q(s,a)$$

$$V^*(s) = \max_a Q(s,a)$$

Q-Learning

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$
$$= r(s,a) + \gamma \max_{a'} Q(\delta(s,a),a')$$

- This still depends on r(s,a) and δ(s,a).
- However, imagine the robot is exploring its environment, trying new actions as it goes.
- At every step it receives some reward "r", and it observes the environment change into a new state s' for action a.
  How can we use these observations, (s,a,s',r) to learn a model?

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a') \quad s' = s_{t+1}$$

- This equation continually estimates Q at state s consistent with an estimate of Q at state s'**: temporal difference (TD-Q) learning**.
- Do an update after each state-action pair.

3 steps in Q-learning

1. Agent starts in a state (s1) takes an action (a1) and receives a reward (r1)
2. Agent selects action by referencing Q-table with highest value (max) OR by random (epsilon, ε)
3. Update q-values

Initialized

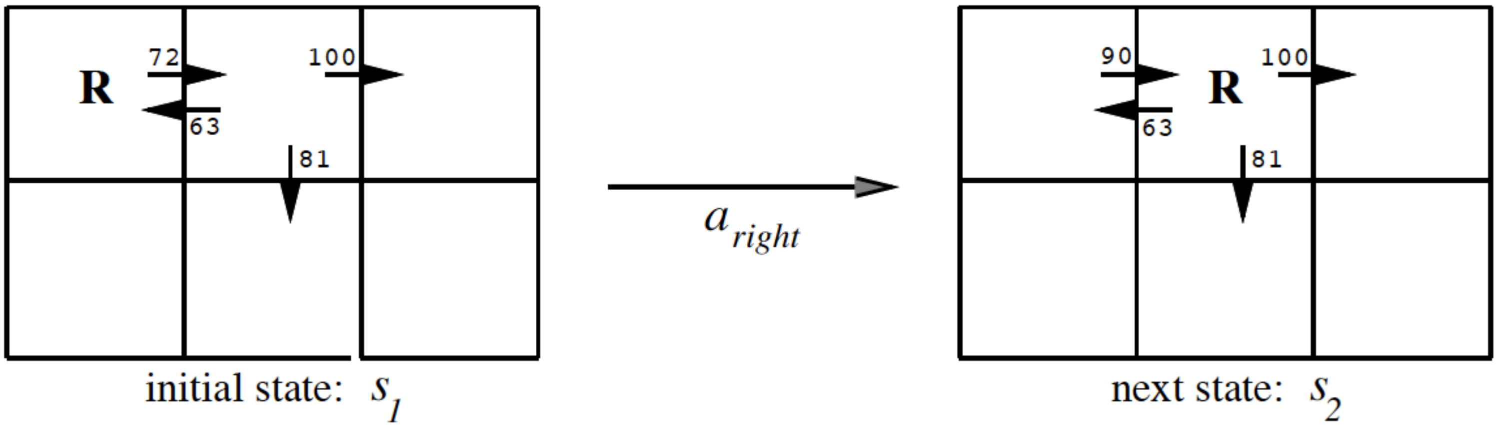
Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	327	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	0	0	0	0	0	0

Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

https://en.wikipedia.org/wiki/Q-learning

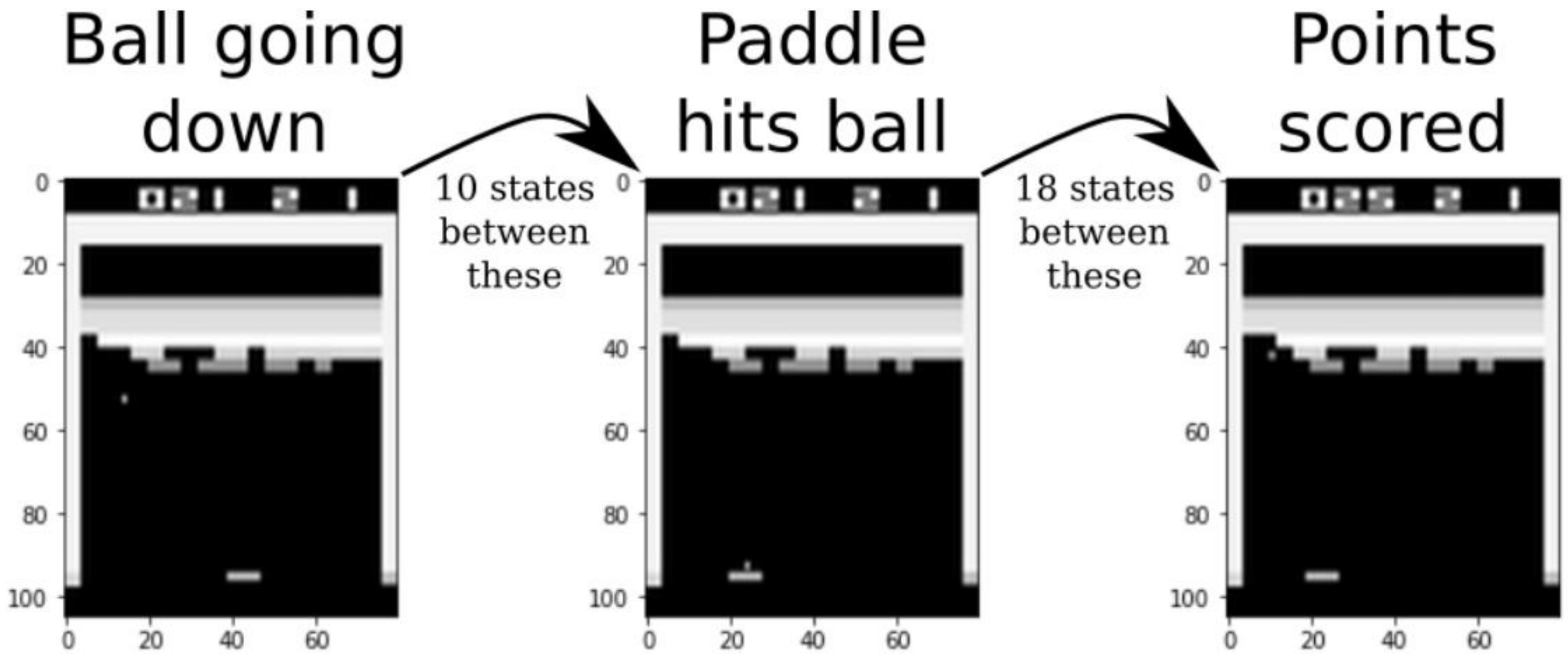
Example Q-Learning



$$
\begin{aligned}
\hat{Q}(S_1, a_{right}) &\leftarrow r + \gamma \max_{a'} \hat{Q}(S_2, a') \\
&\leftarrow 0 + 0.9 \max(63, 81, 100) \\
&\leftarrow 90
\end{aligned}
$$

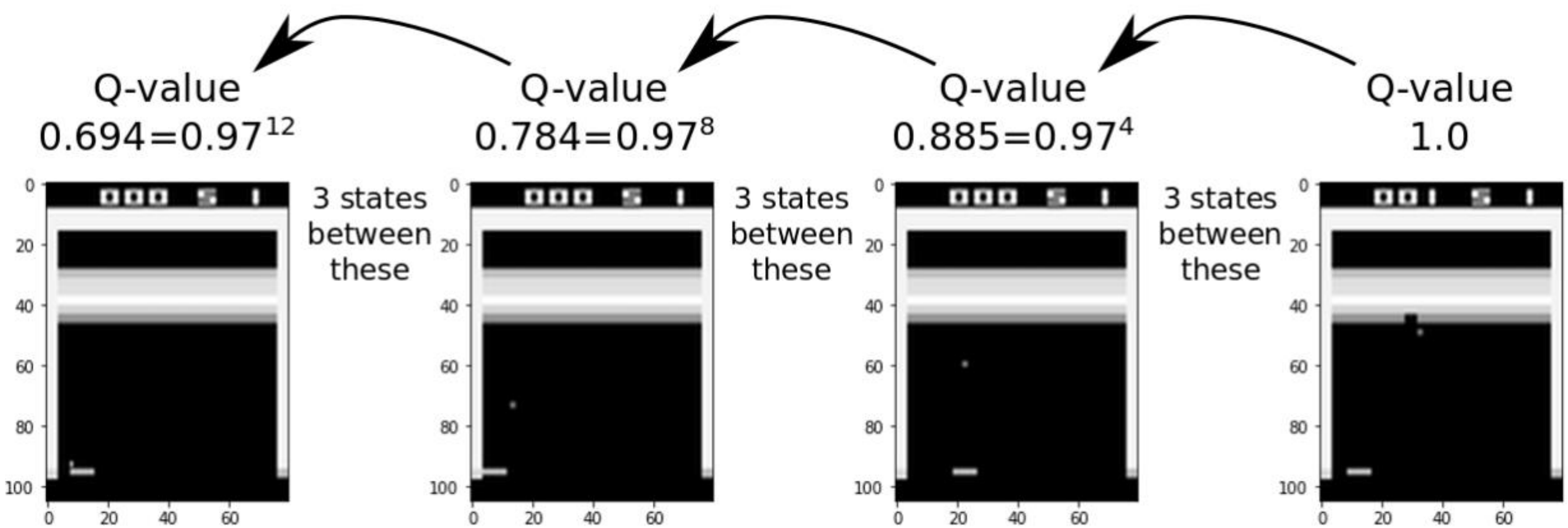
Q-learning propagates Q-estimates 1-step backwards

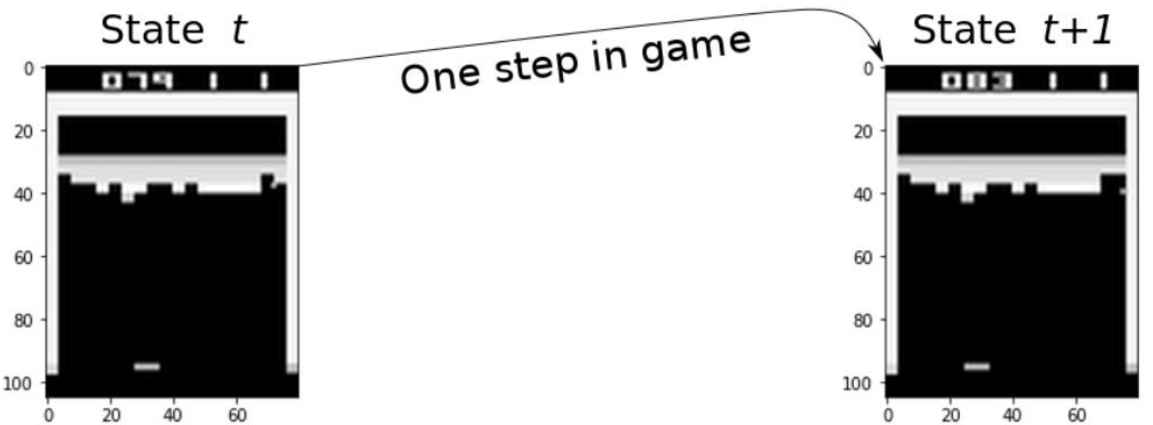
Example: Atari Breakout



https://colab.research.google.com/github/Hvass-Labs/TensorFlow-Tutorials/blob/master/16_Reinforcement_Learning.ipynb#scrollTo=CT6gxbxIJ-V_

$$Q(s_t, a_t) \leftarrow \underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of future rewards}}$$





Action:	Q-Values:
====	======
NOOP	**2.900** (Action Taken)
FIRE	2.622
RIGHT	2.078
LEFT	2.266
RIGHTFIRE	2.183
LEFTFIRE	2.780

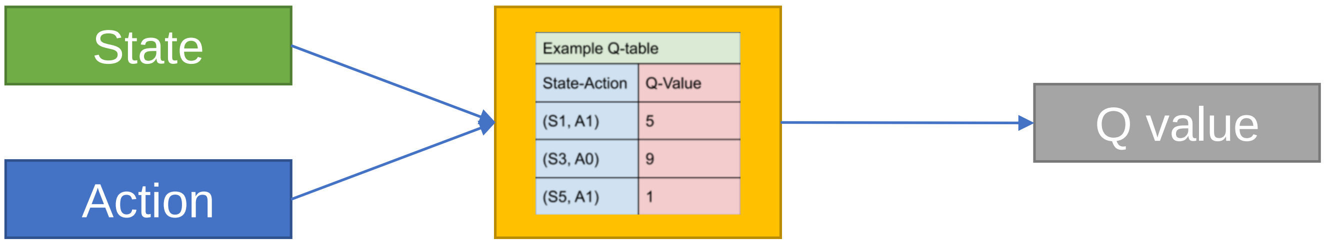
Update Q-Value
1+0.97x1.830=2.775

Action:	Q-Values:
====	======
NOOP	1.814
FIRE	1.777
RIGHT	1.825
LEFT	1.697
RIGHTFIRE	**1.830** (Max Value)
LEFTFIRE	1.740

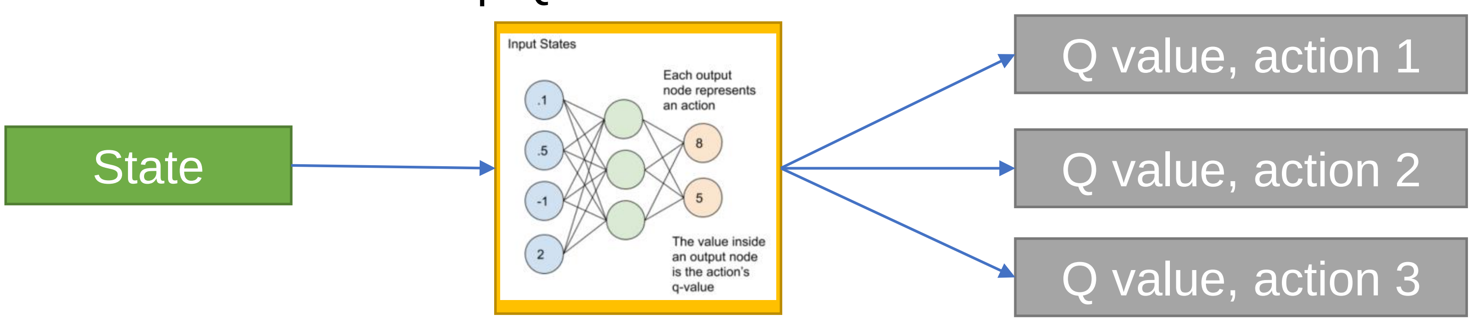
$$Q(state_t, NOOP) \leftarrow \underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount}} \cdot \underbrace{\max_a Q(state_{t+1}, a)}_{\text{estimate of future rewards}} = 1.0 + 0.97 \cdot 1.830 \simeq 2.775$$

RL + DL = Deep Q-Network (DQN)

Qtable

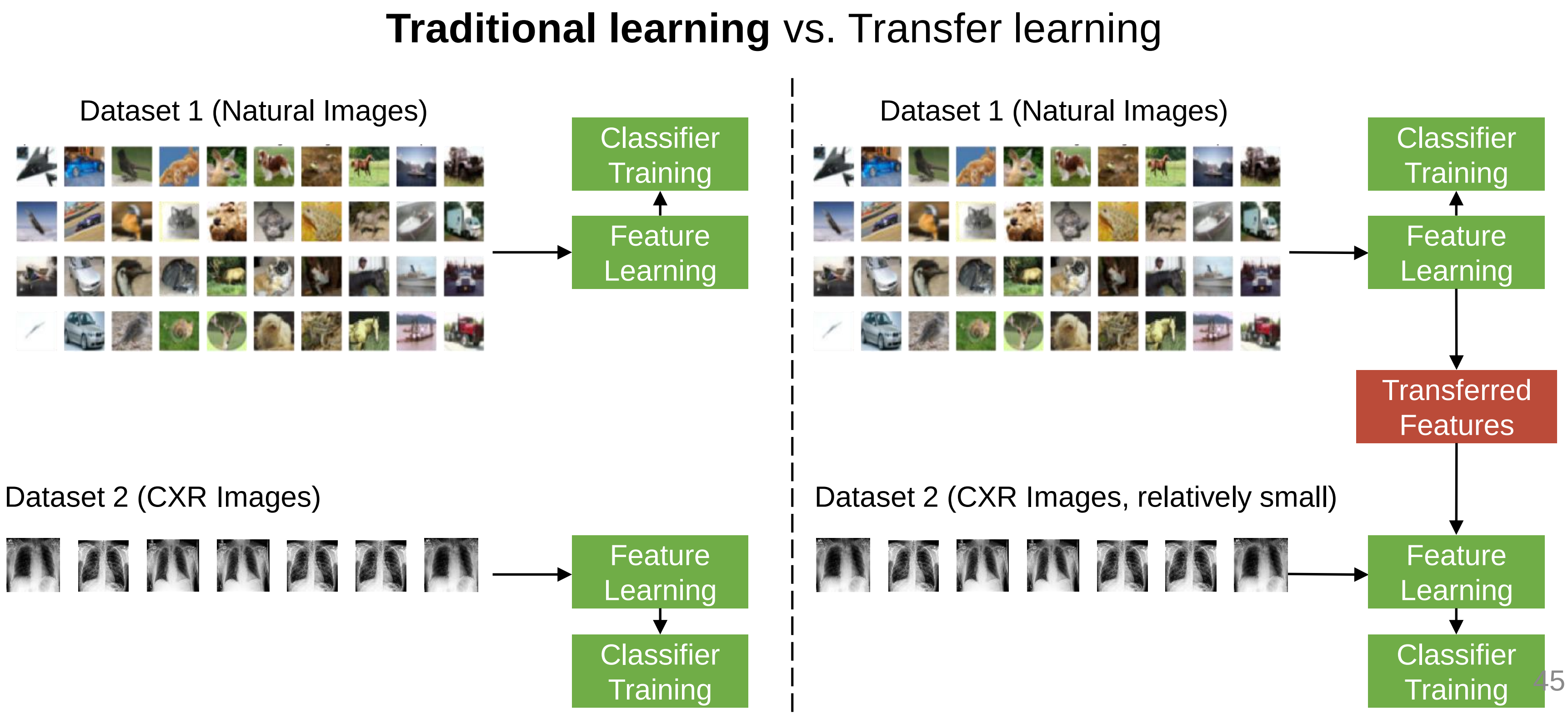


Deep Q Neural Network

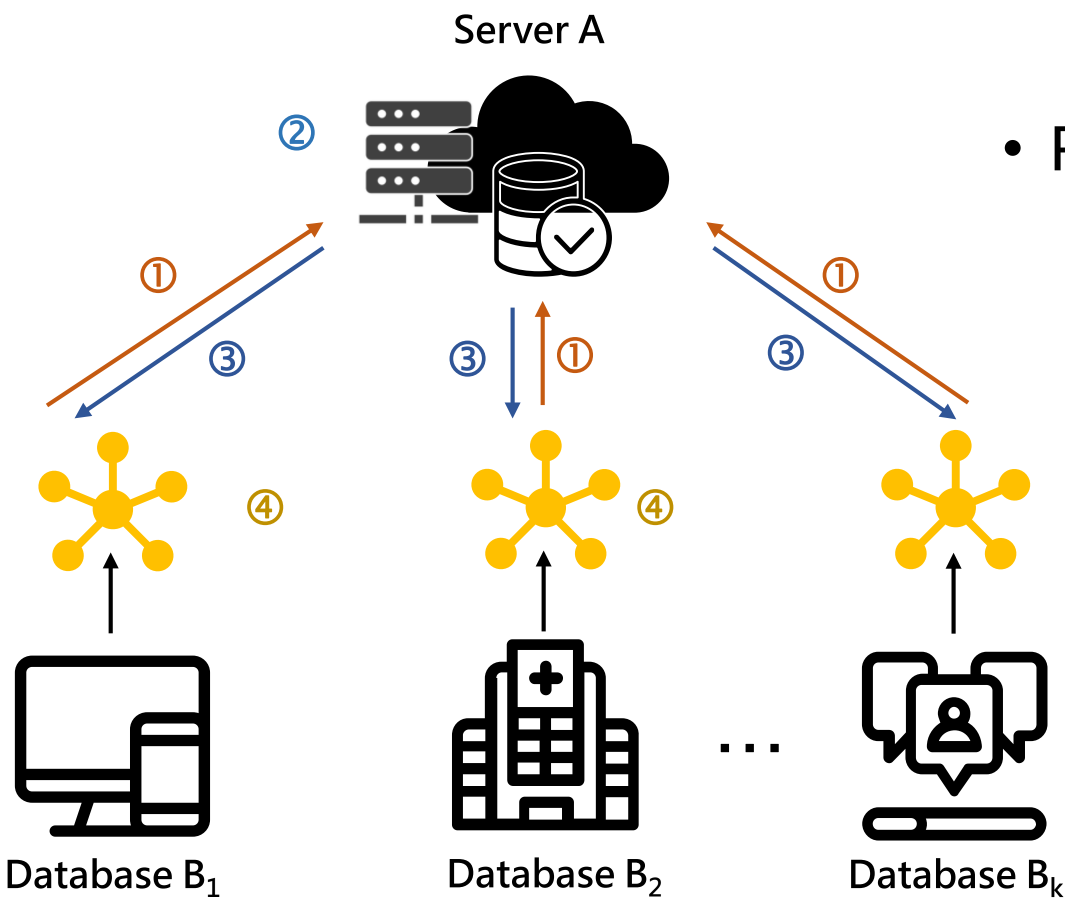


Transfer learning

- Improvement of learning for new task through the transfer of knowledge from similar task that has been trained.



Federated learning



- Federated Learning Steps

① Sending encrypted gradients

② Secure aggregation

③ Sending back model updates

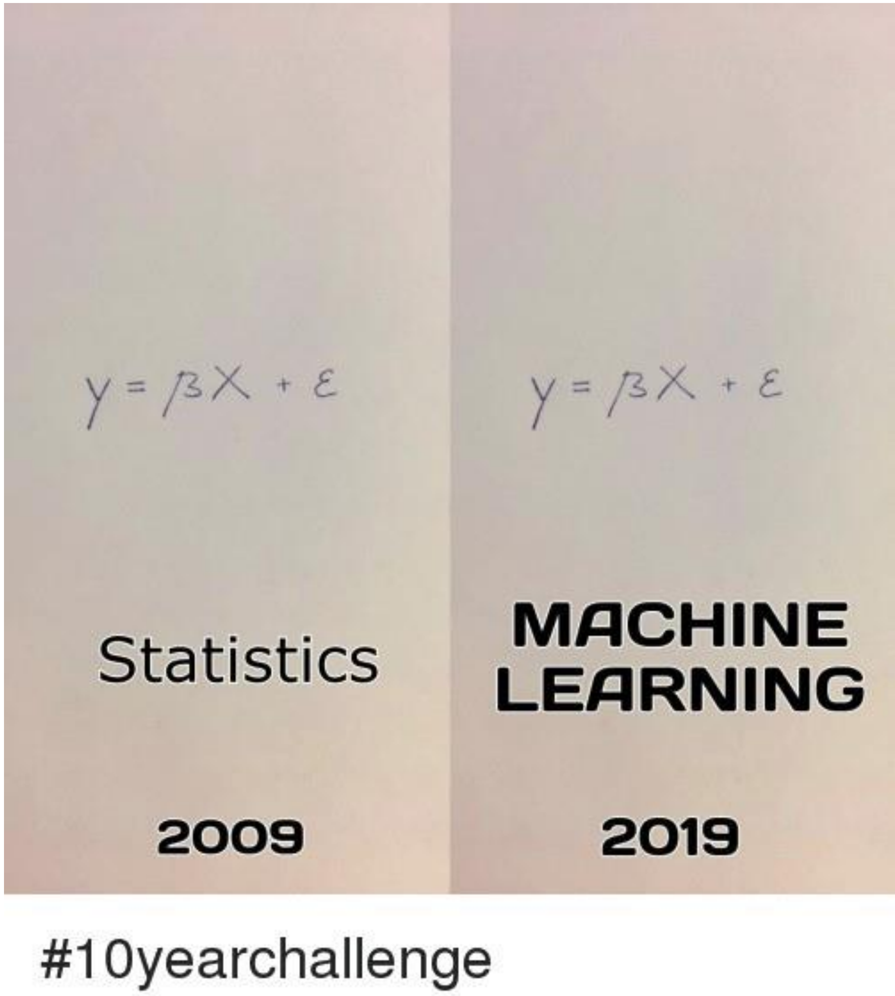
④ Updating models

Konečný, Jakub, et al. "Federated learning: Strategies for improving communication efficiency." *arXiv preprint arXiv:1610.05492* (2016).

Learning, learning, learning

- Supervised learning
  - **FFNNs**
  - **CNNs**
  - **RNNs**
  - **Encoder Decoder**
- Unsupervised learning
  - **Autoencoder**
  - **GAN**
- Self-Supervised learning
- Reinforcement learning
- Transfer learning
- Federated learning

Questions?



$y = \beta X + \varepsilon$
Statistics
2009
$y = \beta X + \varepsilon$
MACHINE LEARNING
2019
#10yearchallenge