DOES AGI = DNN + RL?

Scott O'Hara

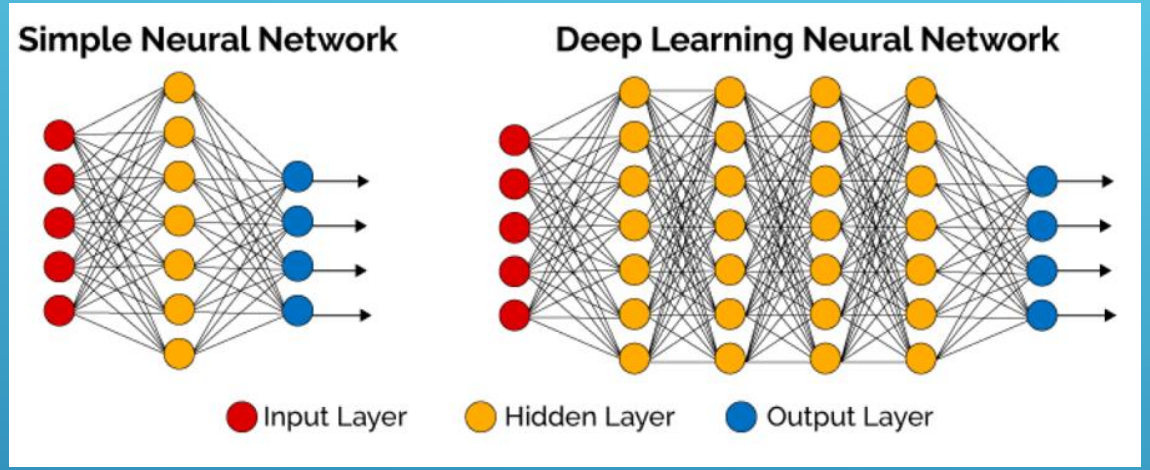
Metrowest Developers Machine Learning Group

5/27/2020

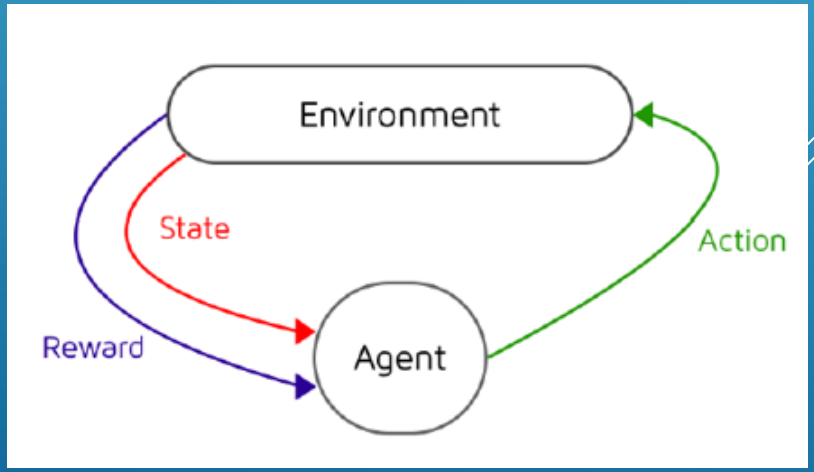
DOES AGI = DNN + RL?

AGI = Artificial General Intelligence

DNN = Deep Neural Networks



RL = Reinforcement Learning



REFERENCES

The material for this talk is primarily drawn from these sources.

MIT Deep Learning **, Prof Lex Fridman.**

- ***https://deeplearning.mit.edu/***

Prediction and Control with Function Approximation**, Profs. Martha White, Adam White. University of Alberta, Coursera.**

- **https://www.coursera.org/learn/prediction-control-function-approximation/**

"Deep Learning: A Critical Appraisal," 2018, Gary Marcus

- https://arxiv.org/abs/1801.00631

Reinforcement learning: an introduction R. S. Sutton and A. G. Barto, Second edition. Cambridge, Massachusetts: The MIT Press, 2018.

CS188 – Introduction to Artificial Intelligence**, Profs. Dan Klein, Pieter Abbeel, et al.**

- **http://ai.berkeley.edu/home.html**

Deep Reinforcement Learning

David Silver

Credit: Prediction and Control with Function Approximation, M. White & A. White, Univ of Alberta, Coursera, 2020

DEEP LEARNING

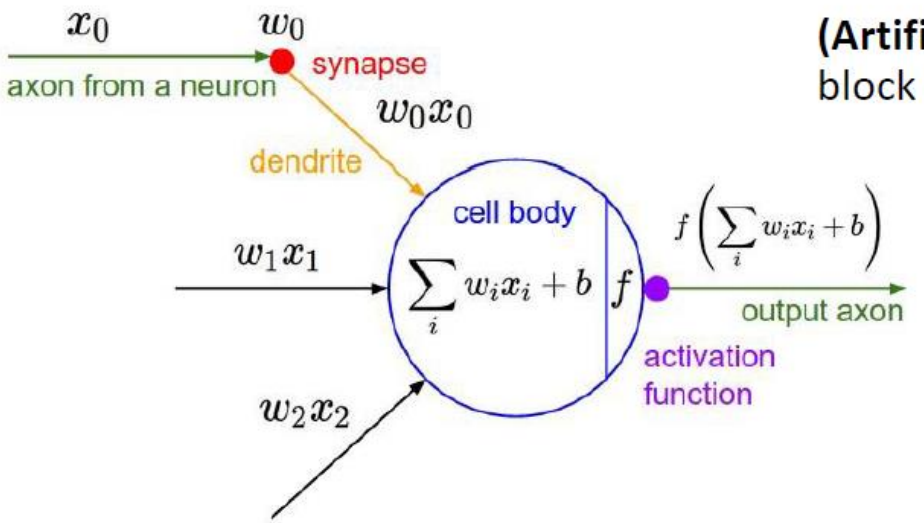
Deep Learning in **One** Slide

- **What is it:**
  Extract useful patterns from data.
- **How:**
  Neural network + optimization
- **How (Practical):**
  Python + TensorFlow & friends
- **Hard Part:**
  Good Questions + Good Data
- **Why now:**
  Data, hardware, community, tools, investment
- **Where do we stand?**
  Most big questions of intelligence have not been answered nor properly formulated

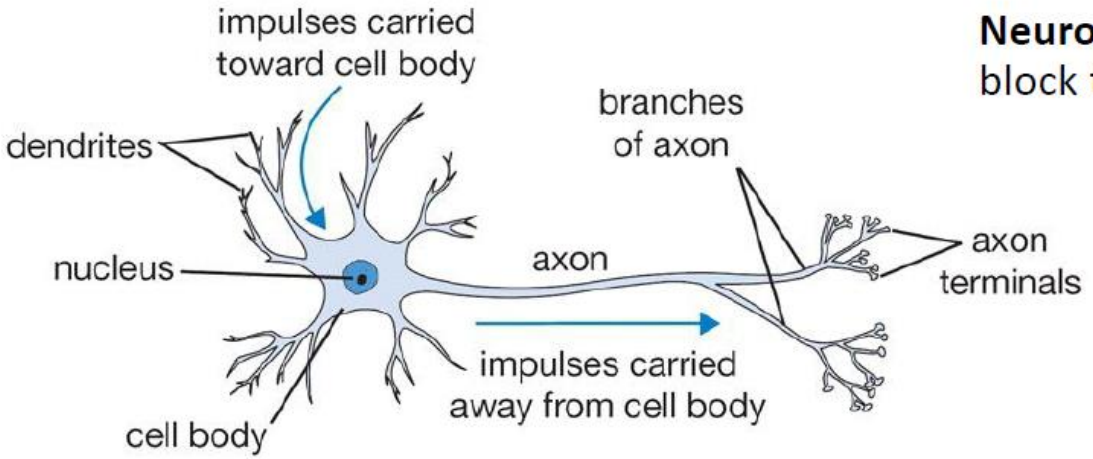
Exciting progress:

- Face recognition
- Image classification
- Speech recognition
- Text-to-speech generation
- Handwriting transcription
- Machine translation
- Medical diagnosis
- Cars: drivable area, lane keeping
- Digital assistants
- Ads, search, social recommendations
- Game playing with deep RL

Neuron: Biological Inspiration for Computation

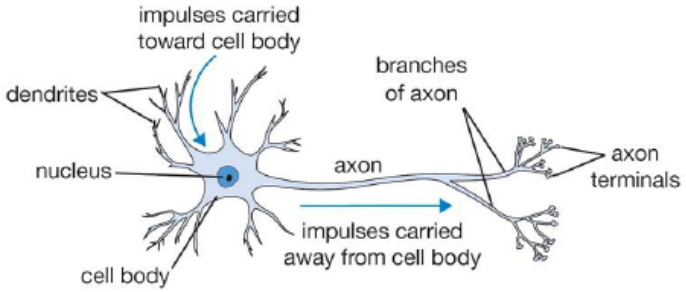


(Artificial) Neuron: computational building block for the "neural network"

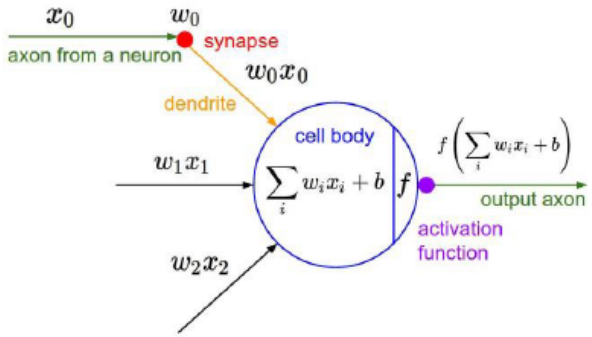


Neuron: computational building block for the brain

Neuron: Biological Inspiration for Computation



- **Neuron:** computational building block for the brain

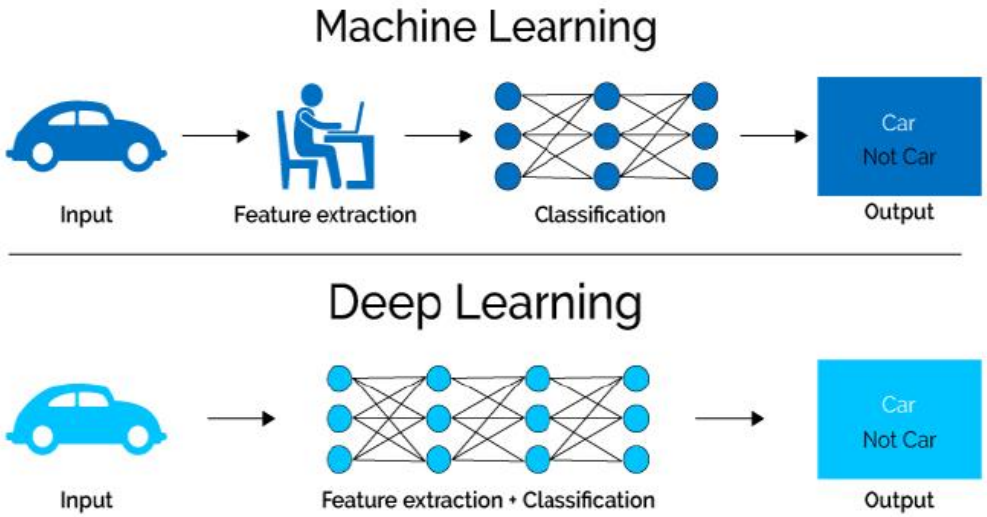


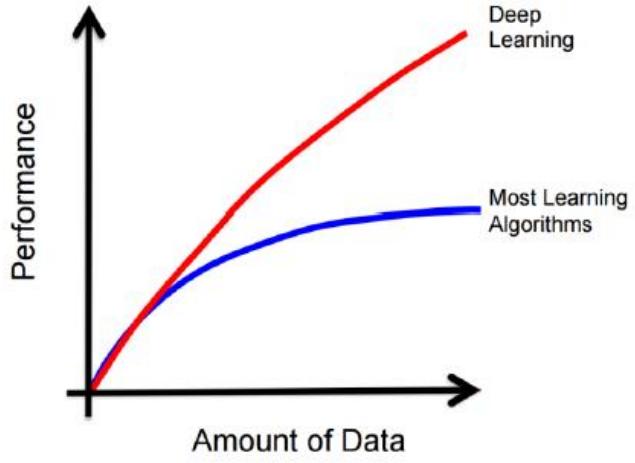
- **(Artificial) Neuron:** computational building block for the "neural network"

Key Difference:

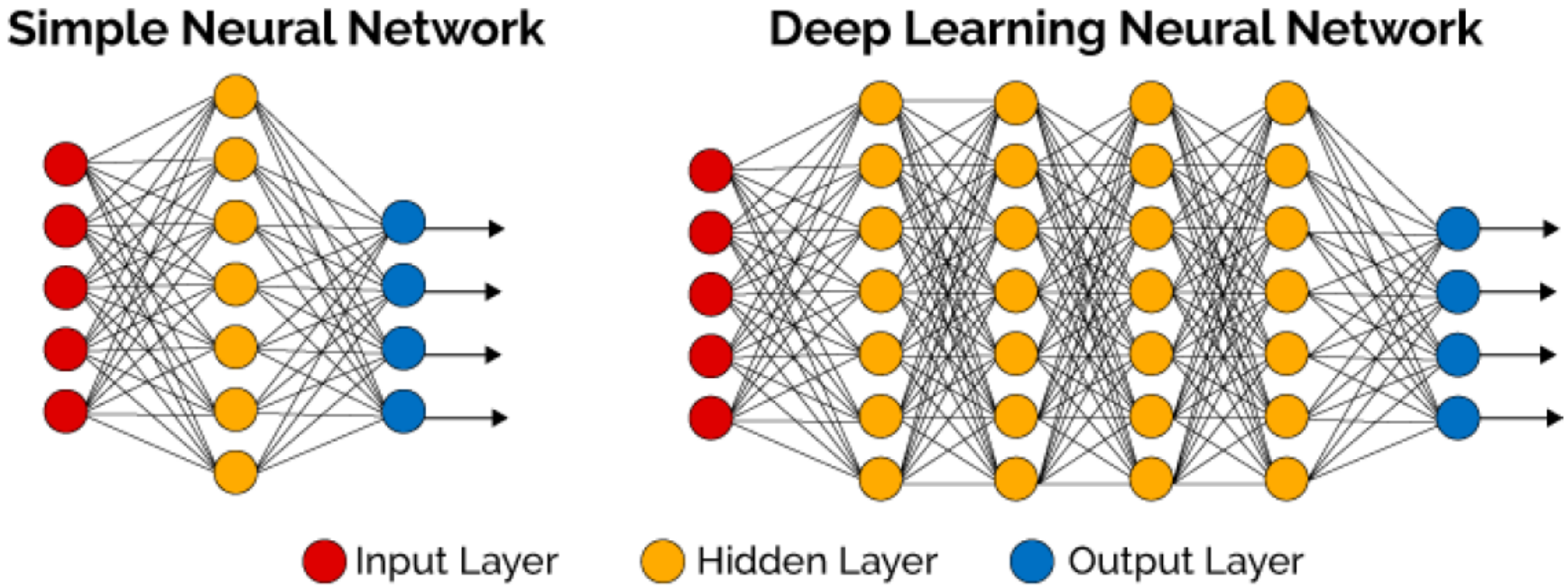
- **Parameters:** Human brains have ~10,000,000 times synapses than artificial neural networks.
- **Topology:** Human brains have no "layers". **Async:** The human brain works asynchronously, ANNs work synchronously.
- **Learning algorithm**: ANNs use gradient descent for learning. We don't know what human brains use
- **Power consumption:** Biological neural networks use very little power compared to artificial networks
- **Stages:** Biological networks usually never stop learning. ANNs first train then test.

Why Deep Learning? **Scalable** Machine Learning





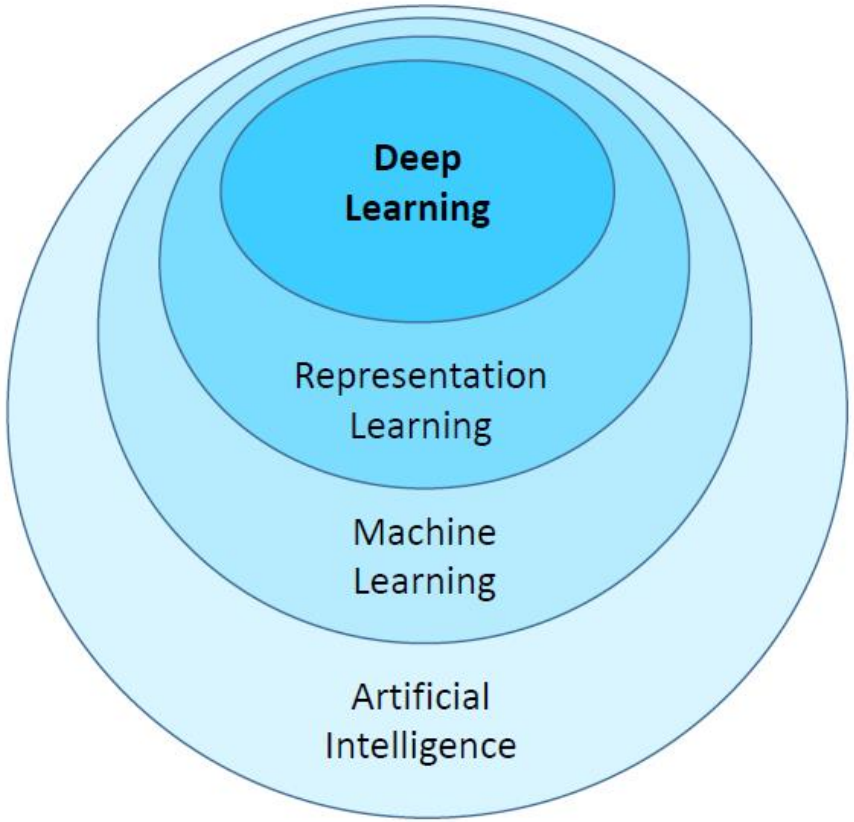
Combing Neurons in Hidden Layers:
The "Emergent" Power to Approximate

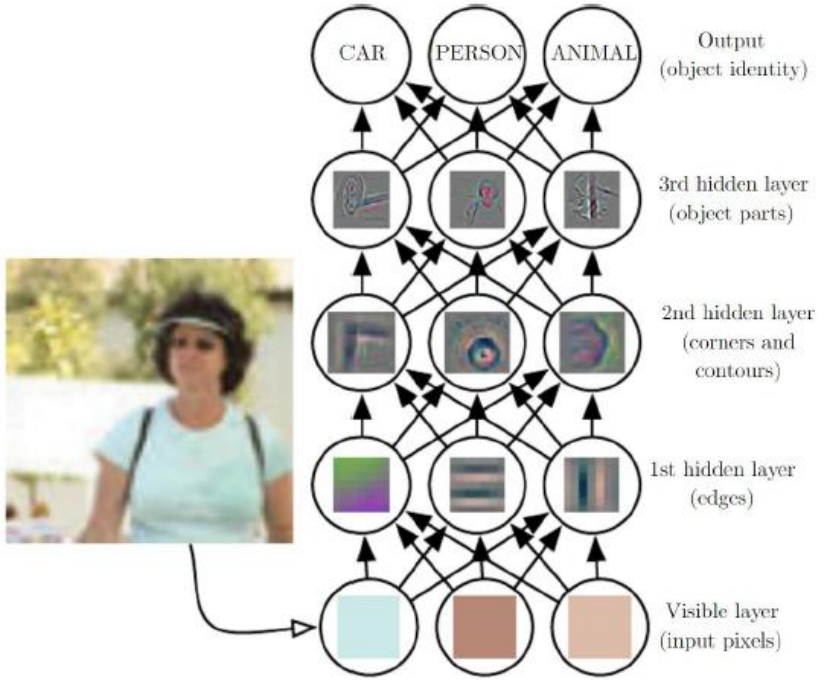


Universality: For any arbitrary function f(x), there exists a neural network that closely approximate it for any input x

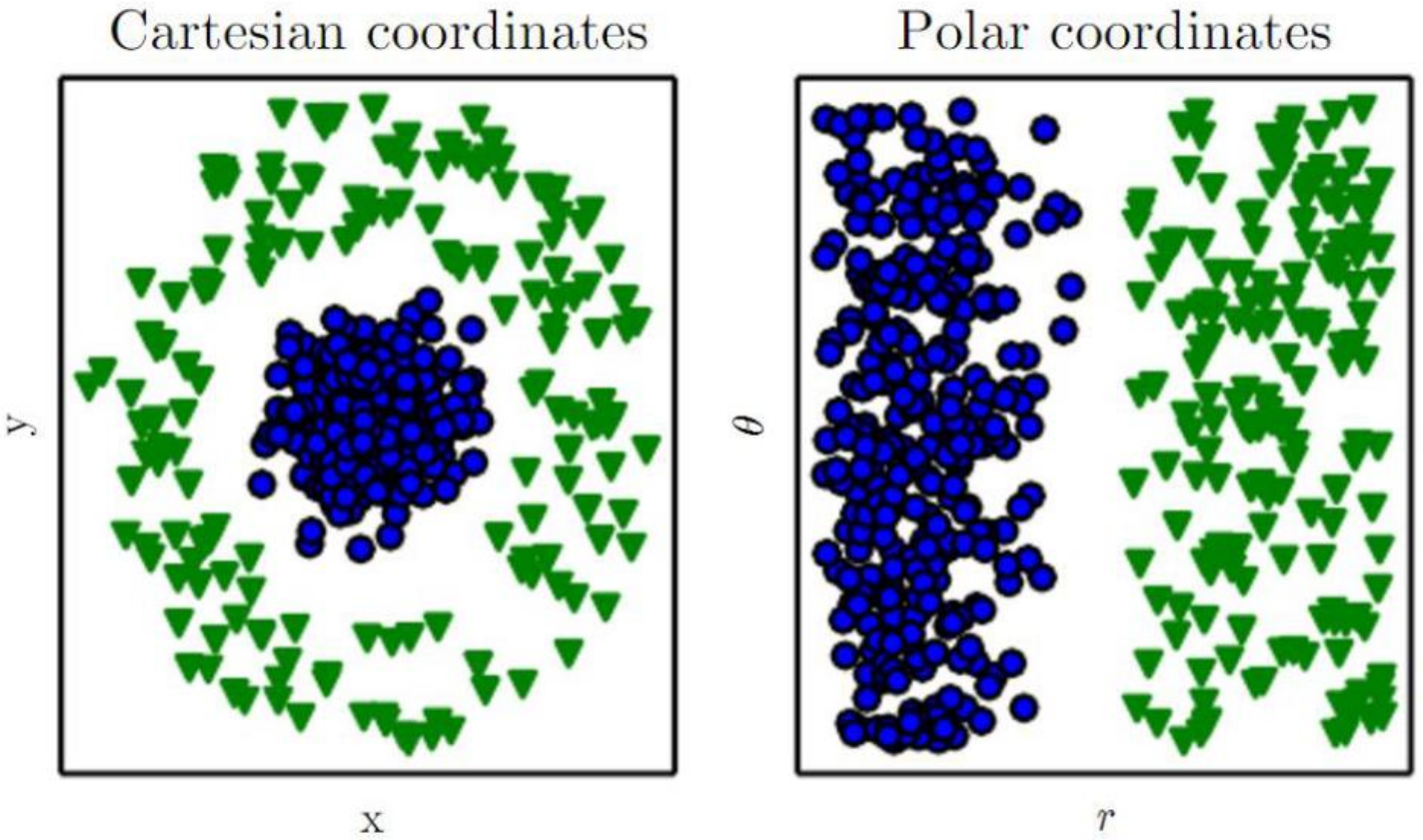
Deep Learning is **Representation Learning**

(aka Feature Learning)





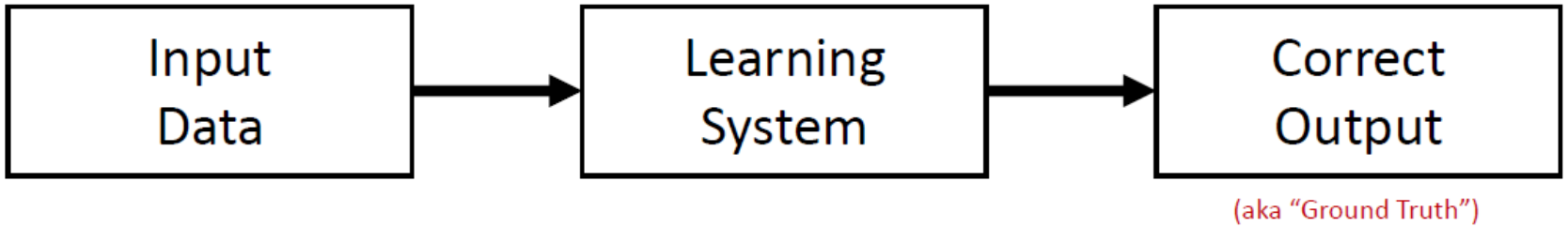
Representation Matters



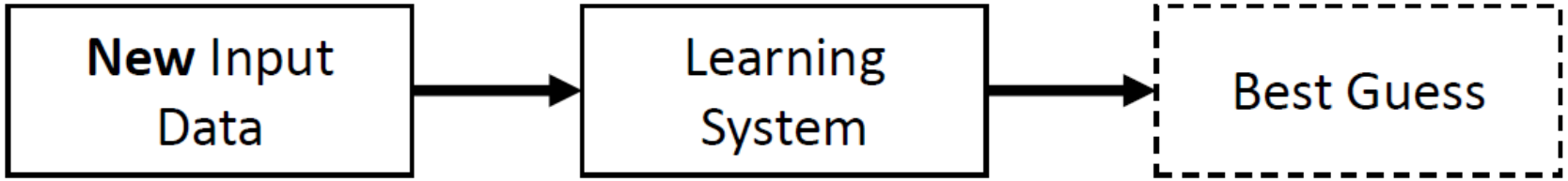
Task: Draw a line to separate the **green triangles** and **blue circles**.

Deep Learning: Training and Testing

Training Stage:

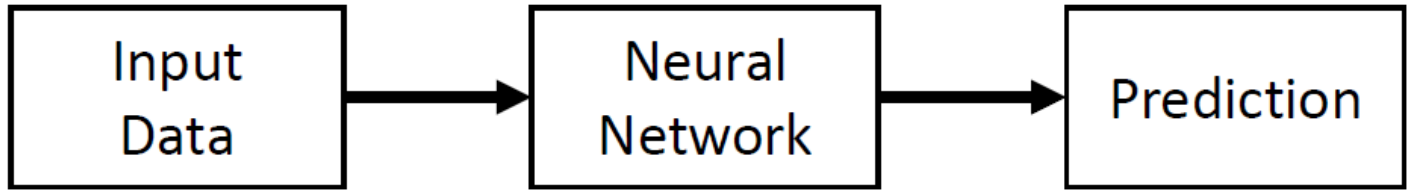


Testing Stage:

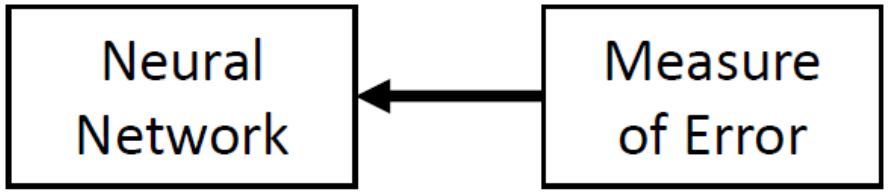


How Neural Networks Learn: Backpropagation

Forward Pass:

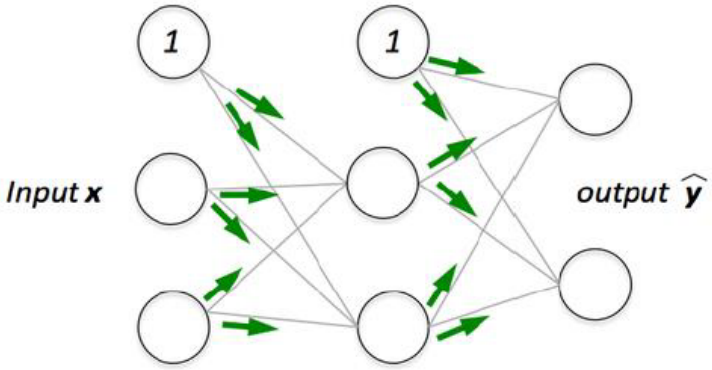


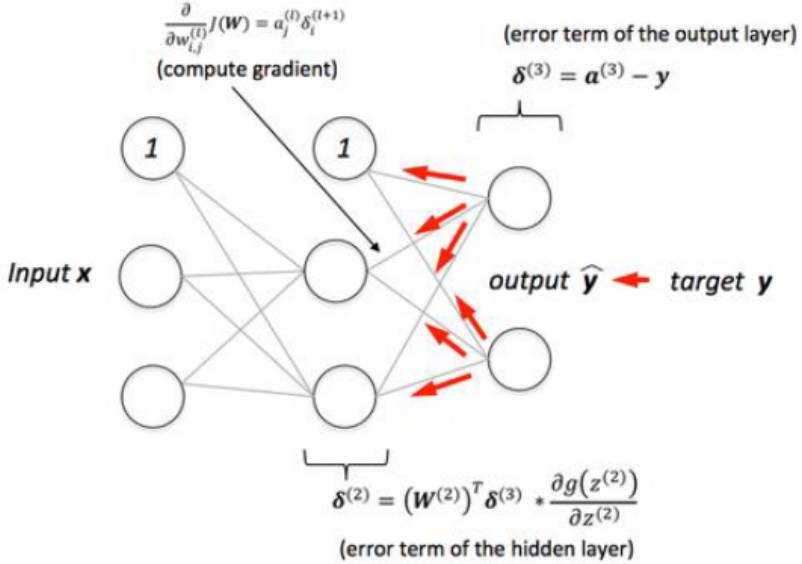
Backward Pass (aka Backpropagation):



Adjust to Reduce Error

Backpropagation





Task: Update the **weights** and **biases** to decrease **loss function**

Subtasks:

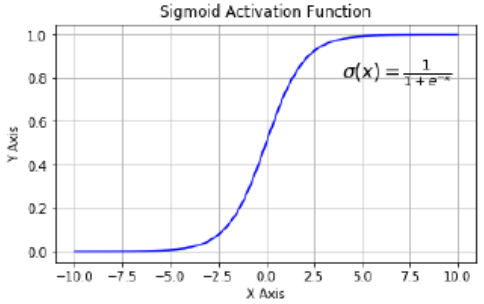
1. Forward pass to compute network output and "error"
2. Backward pass to compute gradients
3. A fraction of the weight's gradient is subtracted from the weight.

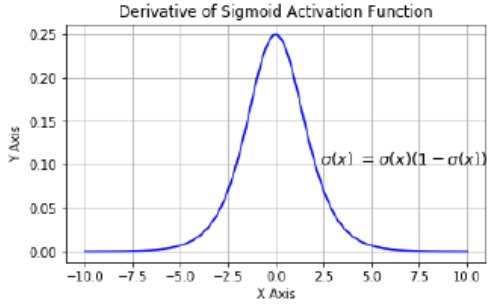
Learning Rate

Numerical Method: Automatic Differentiation

Key Concepts:

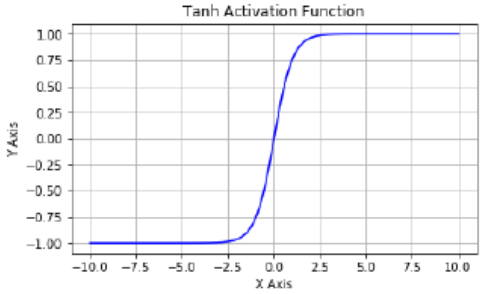
Activation Functions

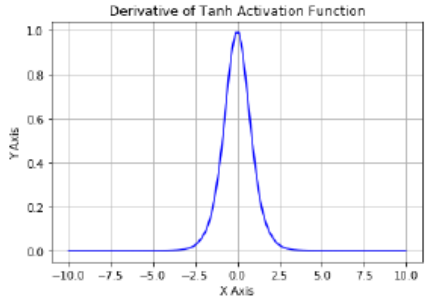




Sigmoid

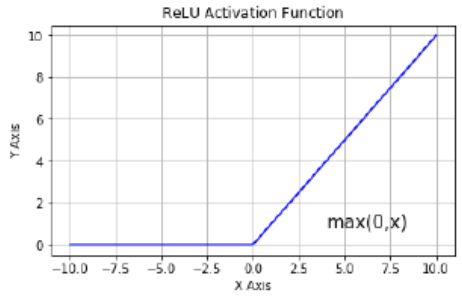
- Vanishing gradients
- Not zero centered

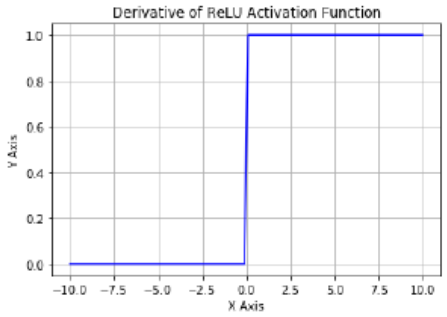




Tanh

- Vanishing gradients

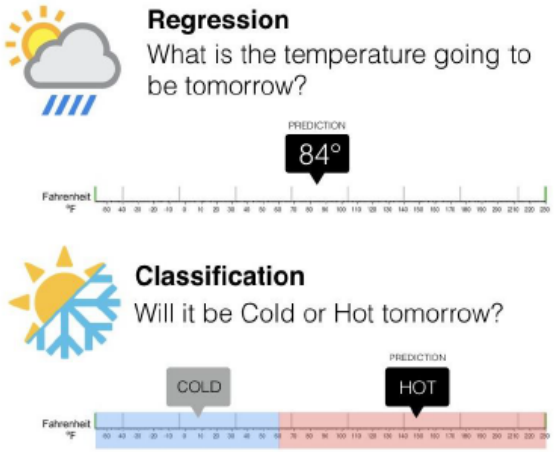




ReLU

- Not zero centered

Loss Functions



- Loss function quantifies gap between prediction and ground truth
- For regression:
  - Mean Squared Error (MSE)
- For classification:
  - Cross Entropy Loss

Mean Squared Error

$$MSE = \frac{1}{N}\sum(t_i - s_i)^2$$

Prediction

Ground Truth

Cross Entropy Loss

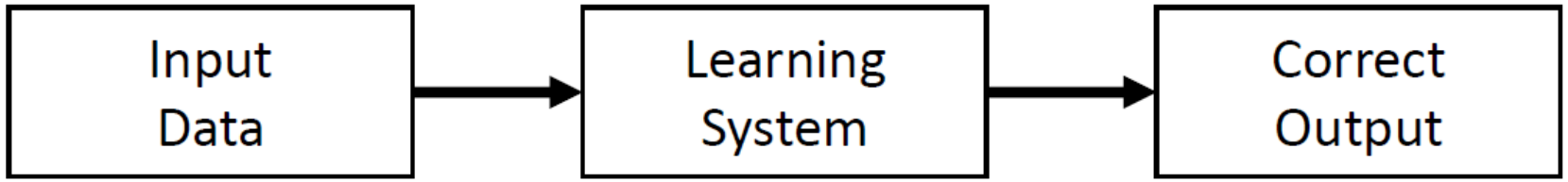
$$CE = -\sum_{i}^{C} t_i log(s_i)$$

Classes

Prediction

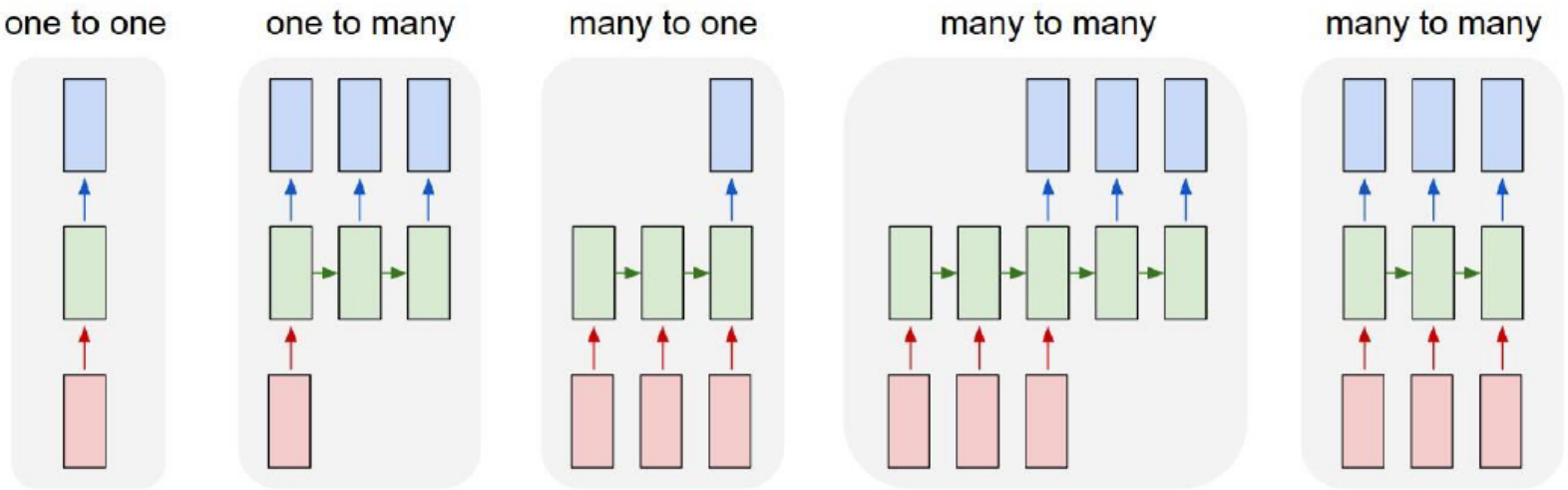
Ground Truth {0,1}

What can we do with Deep Learning?

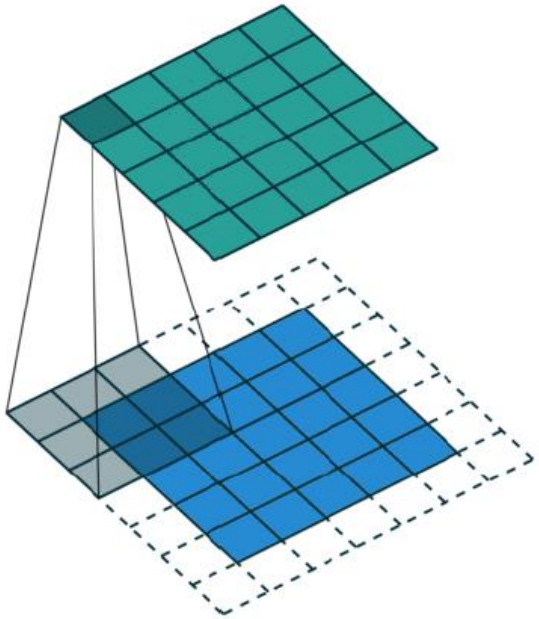


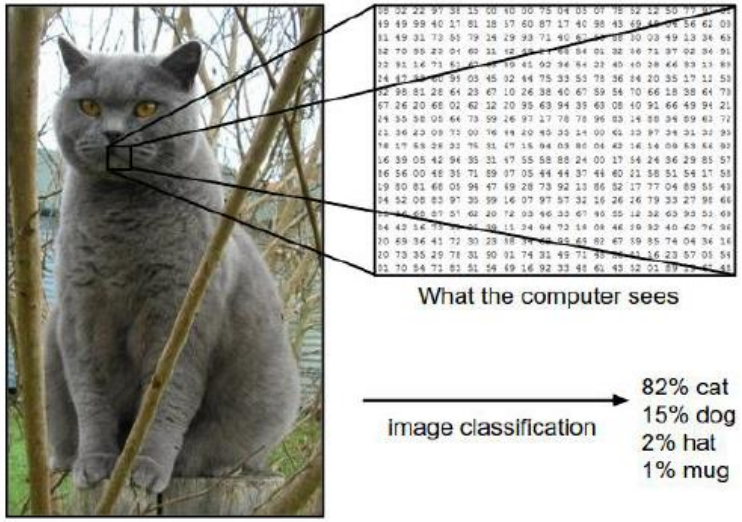
- Number
- Vector of numbers
- Sequence of numbers
- Sequence of vectors of numbers

- Number
- Vector of numbers
- Sequence of numbers
- Sequence of vectors of numbers

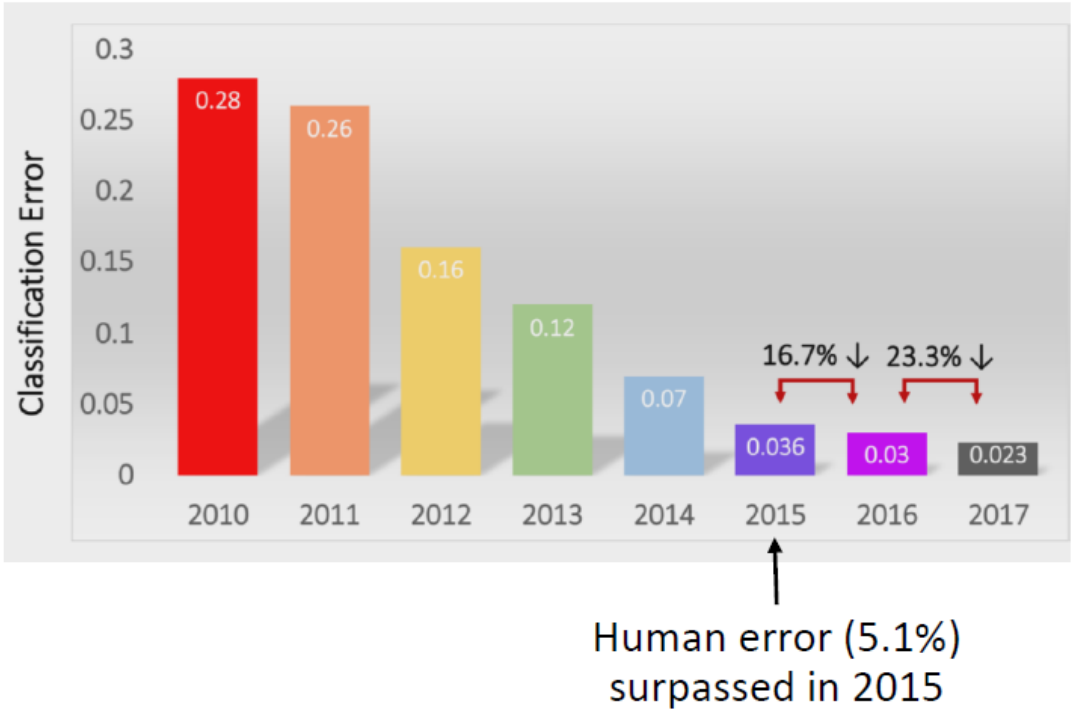


Convolutional Neural Networks: Image Classification





- Convolutional filters: take advantage of spatial invariance

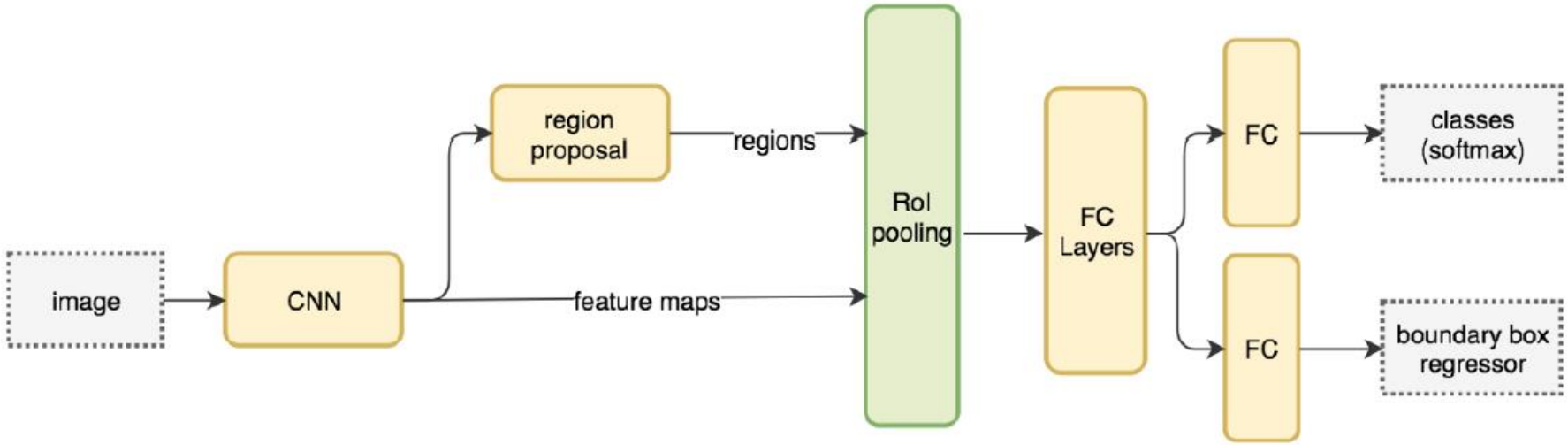


- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models
- **SENet (2017): 2.99% to 2.251%**
  - Squeeze and excitation block: network is allowed to adaptively adjust the weighting of each feature map in the convolutional block.

References: [90]

Object Detection / Localization

Region-Based Methods | Shown: Faster R-CNN




```
ROIs = region_proposal(image)
for ROI in ROIs
    patch = get_patch(image, ROI)
    results = detector(patch)
```

MANY OTHER DL ARCHITECTURES

- **Transfer Learning**

  Pretrained sub-models for e.g., language, object recognition etc.
- **Autoencoders**

  Data compression, unsupervised learning
- **Generative Adversarial Network (GAN)**

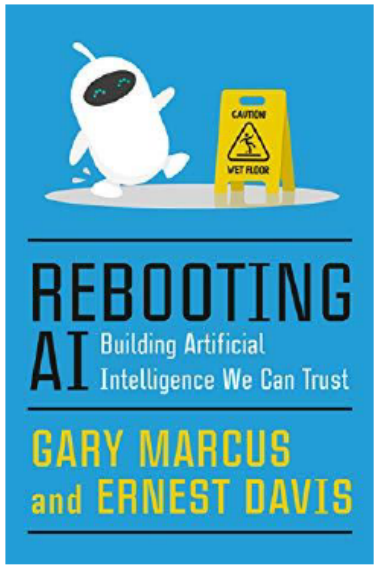
  Synthesize “fake” pictures
- **Recurrent Neural Networks (RNNs)**

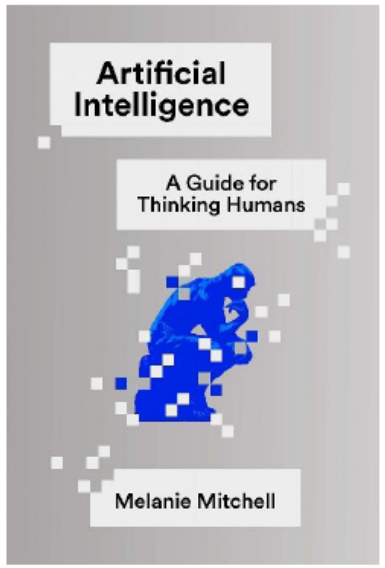
  Sequence data, text, speech, audio, video,
- **Transformers**

  Current go to architecture for language translation.

THE LIMITATIONS OF DEEP LEARNING

Limitations of Deep Learning





- 2019 is the year it became cool to say that "deep learning" has limitations.
- Books, articles, lectures, debates, videos were released that learning-based methods cannot do commonsense reasoning.

Prediction from Rodney Brooks:

"By 2020, the popular press starts having stories that the era of Deep Learning is over."

http://rodneybrooks.com/predictions-scorecard-2019-january-01/

DEEP LEARNING: A CRITICAL APPRAISAL

Author: Gary Marcus
Published: 2018
Link: https://arxiv.org/abs/1801.00631

- Gary Marcus is also the author of **Rebooting AI**, a well-known book discussing the limitations of deep learning and making recommendations about the future of AI research.

- In *Deep Learning: A Critical Appraisal,* Marcus lists 10 limitations on the scope of deep learning.

Credit: Deep Learning: A Critical Appraisal, Gary Marcus, 2018

10 LIMITATIONS OF DEEP LEARNING

1. DL is data hungry.
2. DL representations are shallow and don't transfer well.
3. DL has no natural way to deal with hierarchical structure.
4. DL struggles with open-ended inference.
5. DL is not sufficiently transparent (DL systems are black boxes)
6. DL has not been well integrated with prior knowledge.
7. DL cannot inherently distinguish between causation and correlation.
8. DL presumes a largely stable world, in ways that may be problematic.
9. DL works well as an approximation, but its answers often cannot be fully trusted.
10. DL is difficult to engineer.

Credit: Deep Learning: A Critical Appraisal, **Gary Marcus, 2018**

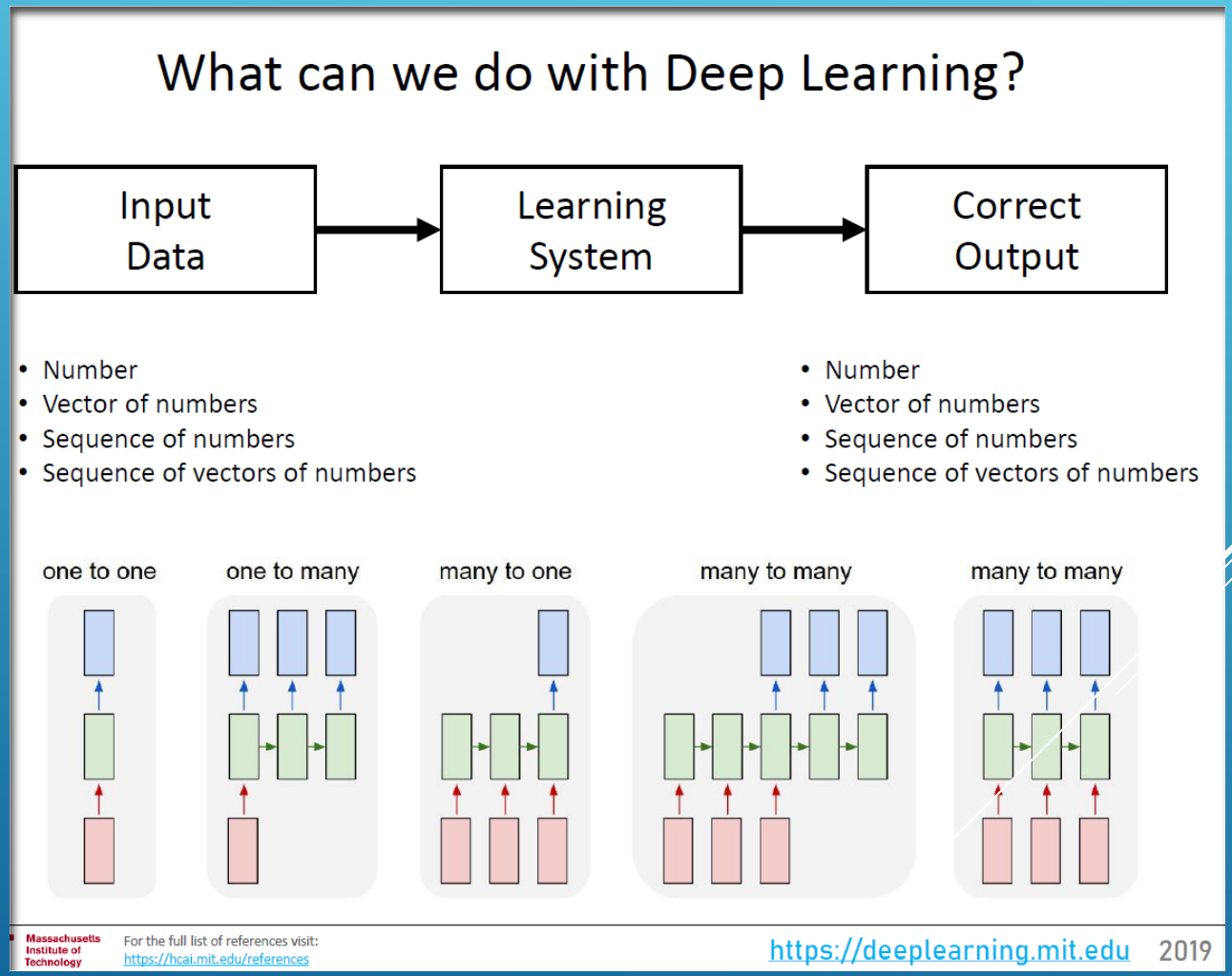
DEEP LEARNING IS DATA HUNGRY

- Humans can learn abstract relationships in a few trials.
- If I told you that a *schmister* was a sister over the age of 10 but under the age of 21, perhaps giving you a single example, you could immediately infer whether you had any schmisters…
- Deep learning currently lacks a mechanism for learning abstractions through explicit verbal definition and works best when there are thousands, millions, or even billions of training examples.

Credit: Deep Learning: A Critical Appraisal, **Gary Marcus, 2018**

DL HAS NO NATURAL WAY TO DEAL WITH HIERARCHICAL STRUCTURE.

DL inputs and outputs are naturally flat. To create structure in DL representations requires a hack. (my opinion)



Credit: Deep Learning: A Critical Appraisal, Gary Marcus, 2018

DEEP LEARNING THUS FAR HAS NOT BEEN WELL INTEGRATED WITH PRIOR KNOWLEDGE

- Researchers in deep learning appear to have a very strong bias against including prior knowledge even when (as in the case of physics) that prior knowledge is well known.
- It is not straightforward in general to integrate prior knowledge into a deep learning system in part because the knowledge represented in deep learning systems pertains mainly to (largely opaque) correlations between features, rather than to abstractions like quantified statements (e.g. *all men are mortal*)..

Credit: Deep Learning: A Critical Appraisal, **Gary Marcus, 2018**

EASILY-DRAWN INFERENCES THAT PEOPLE CAN ANSWER WITHOUT ANYTHING LIKE DIRECT TRAINING:

- *Who is taller, Prince William or his baby son Prince George?*
- *Can you make a salad out of a polyester shirt?*
- *If you stick a pin into a carrot, does it make a hole in the carrot or in the pin?*

Credit: Deep Learning: A Critical Appraisal, **Gary Marcus, 2018**

DEEP REINFORCEMENT LEARNING

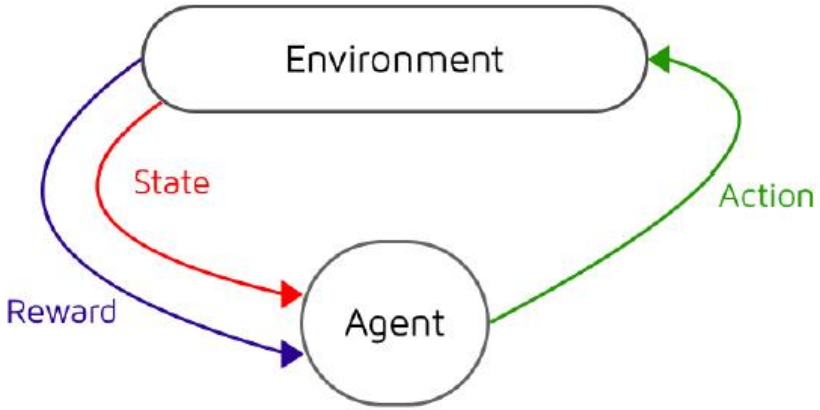
Reinforcement Learning Framework

At each step, the agent:

- Executes action
- Observe new state
- Receive reward

Open Questions:

- What cannot be modeled in this way?
- What are the challenges of learning in this framework?



THE MARKOV DECISION PROCESS

- The ***Markov Decision Process*** (MDP) provides a mathematical framework for reinforcement learning.
- Markov decision processes use **probability** to model **uncertainty** about the domain.
- Markov decision use **utility** to model an agent's **objectives**. The higher the utility, the "happier" your agent is.
- MDP algorithms discover an **optimal decision policy (π)** specifying how the agent should act in all possible states in order to maximize its expected utility.

MARKOV DECISION PROCESSES

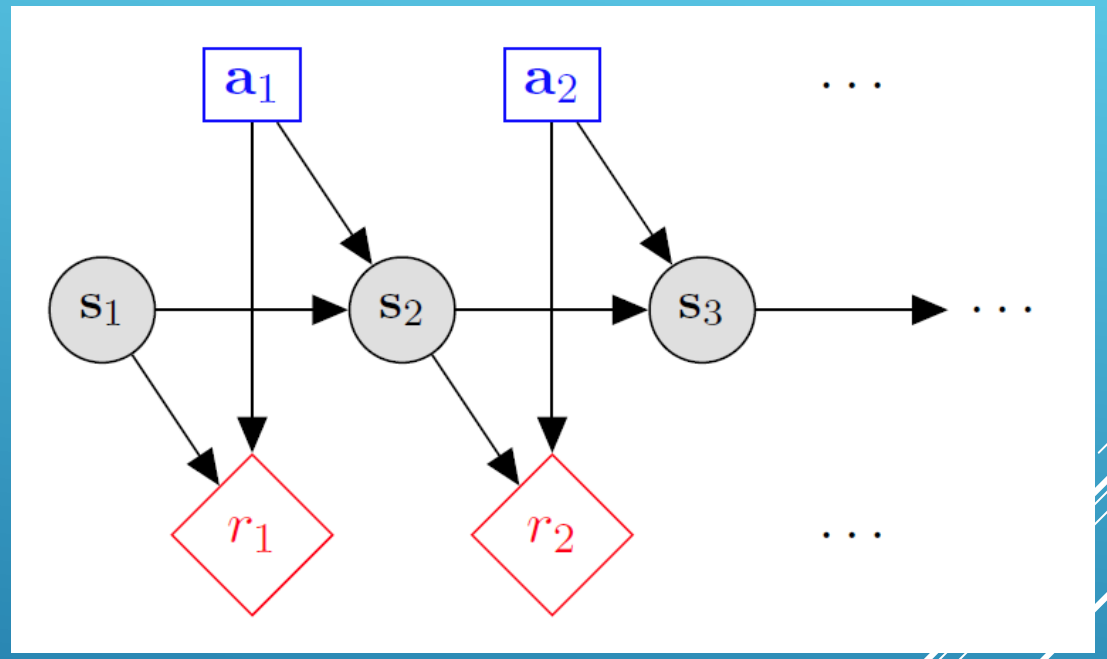
- **States:** $s_1, \ldots, s_n$
- **Actions:** $a_1, \ldots, a_m$
- **Reward model:**

$$\mathrm{R}(s, a, s') \in R$$

- **Transition model:**

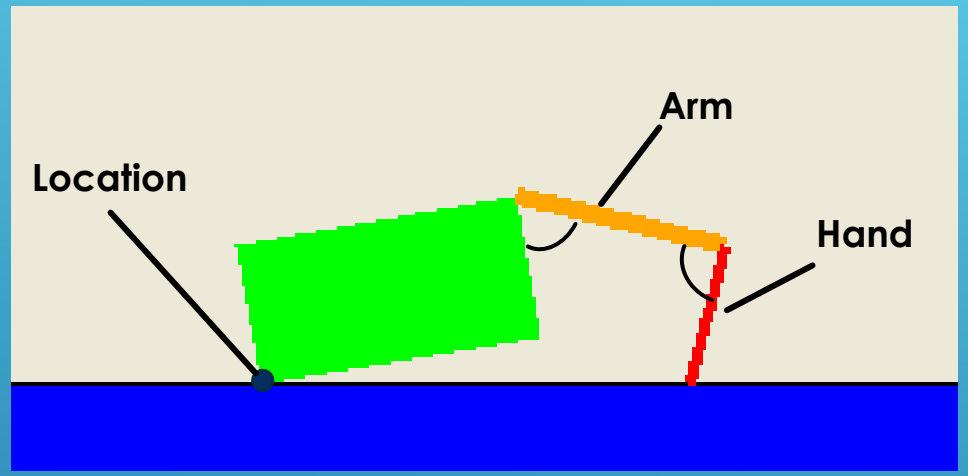
$$\boldsymbol{T}(s, a, s') = \mathrm{P}(s'|s, a)$$

- **Discount factor:** $\gamma \in [0, 1]$



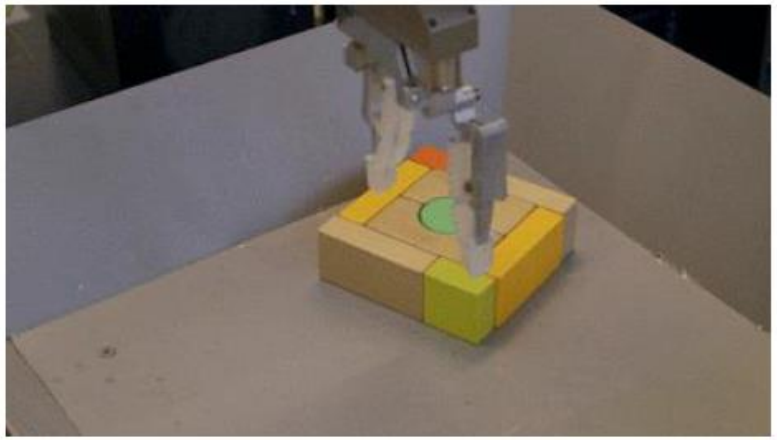
<https://en.wikipedia.org/w/index.php?title=Markov_decision_process&oldid=855934986> [accessed 11 September 2018]

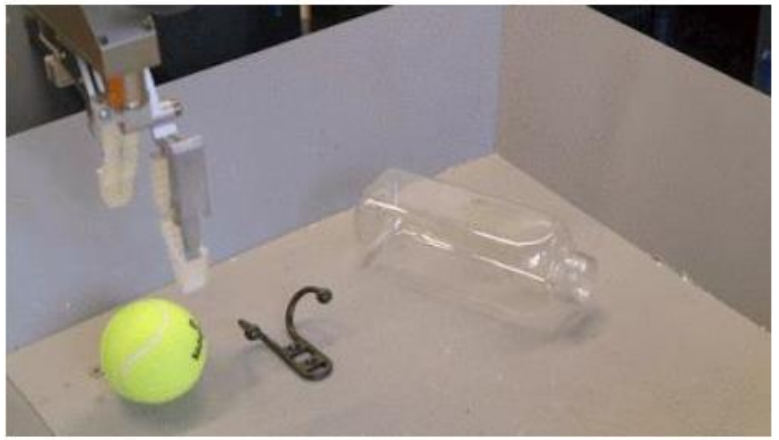
APPLICATION: CRAWLER ROBOT



- **States:** <Location, Arm angle, Hand angle>
- **Actions:** increase Arm angle, decrease Arm angle,
  increase Hand angle, decrease Hand angle.
- **Reward model:** +1 if robot moves right, -1 if robot moves left.
- **Transition model:** model of box movement caused by arm movements.

Examples of Reinforcement Learning





Grasping Objects with Robotic Arm

- **Goal** - Pick an object of different shapes
- **State** - Raw pixels from camera
- **Actions** – Move arm. Grasp.
- **Reward** - Positive when pickup is successful

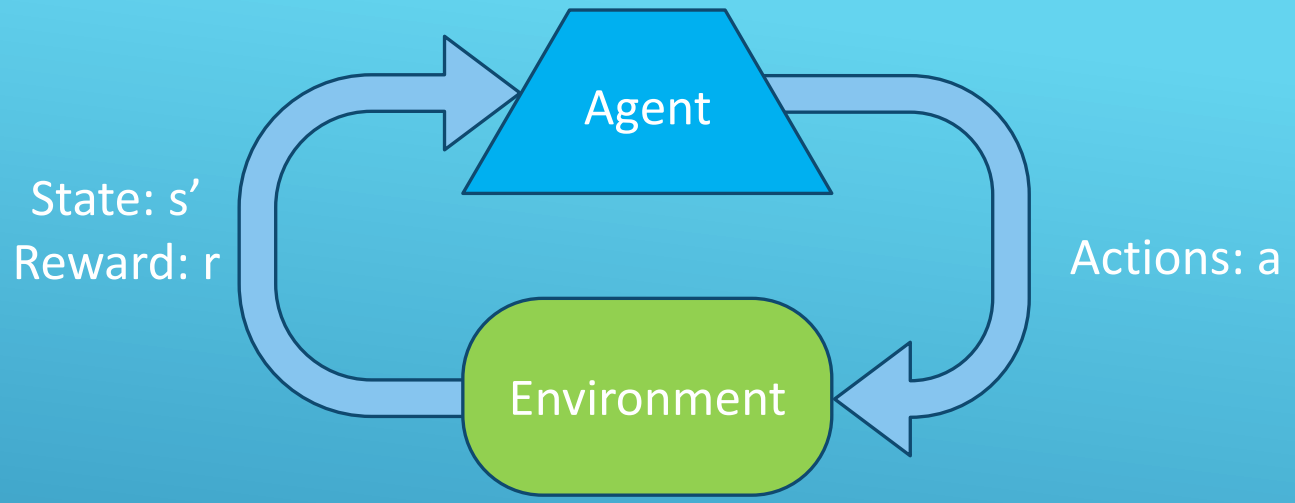
Examples of Reinforcement Learning

Doom*

- **Goal**:
  Eliminate all opponents
- **State:**
  Raw game pixels of the game
- **Actions:**
  Up, Down, Left, Right, Shoot, etc.
- **Reward:**
- Positive when eliminating an opponent,
  negative when the agent is eliminated

** Added for important thought-provoking considerations of AI safety in the context of autonomous weapons systems (see AGI lectures on the topic).*

THE REINFORCEMENT LEARNING PROBLEM

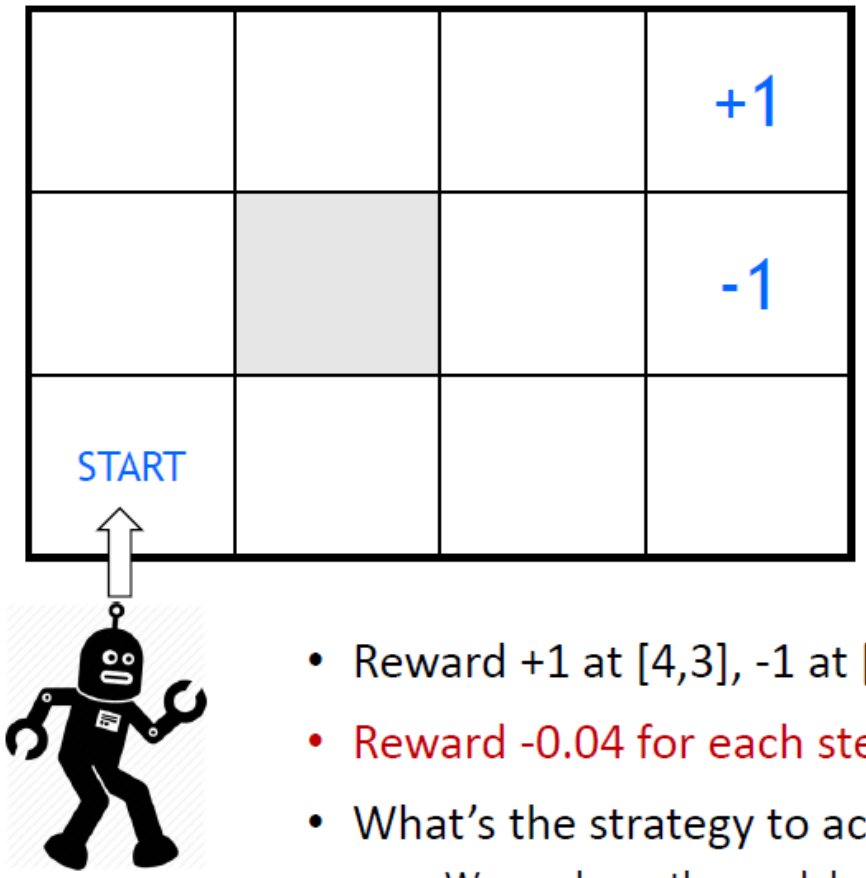


- Agent must learn to act to maximize expected rewards.
- Agent knows the current state s, takes an action ***a***, receives a reward ***r*** and observes the next state ***s'***.

$$S_0, A_0, R_0, S_1, A_1, R_2, S_2, A_2, R_2, \dots, S_n, A_n, R_n, S_T$$

- Agent has **no access** to the reward model ***r(s,a,s')*** or the transition model ***T(s,a,s')***.

Robot in a Room



actions: UP, DOWN, LEFT, RIGHT

(Stochastic) model of the world:

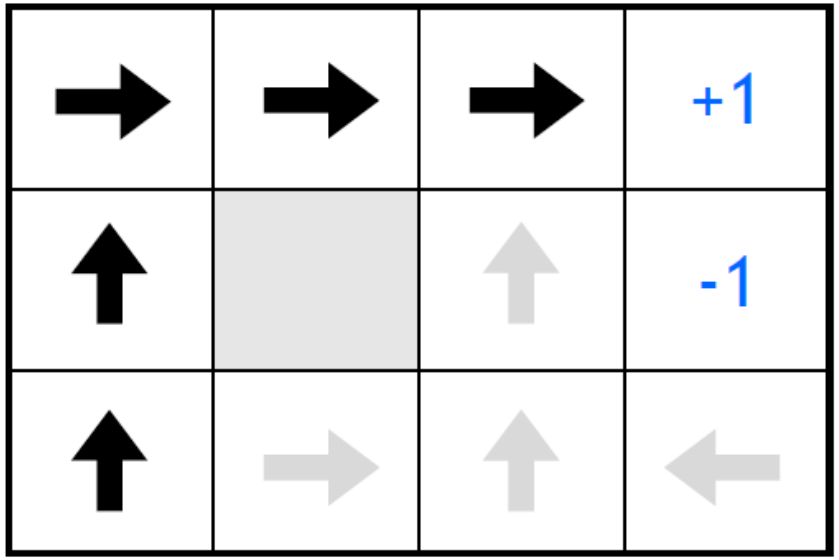
Action: **UP**

80% move UP
10% move LEFT
10% move RIGHT

- Reward +1 at [4,3], -1 at [4,2]
- Reward -0.04 for each step
- What's the strategy to achieve max reward?
  - We can learn the model and plan
  - We can learn the value of (action, state) pairs and act greed/non-greedy
  - We can learn the policy directly while sampling from it

Optimal Policy for a Deterministic World

Reward: **-0.04** for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are deterministic:

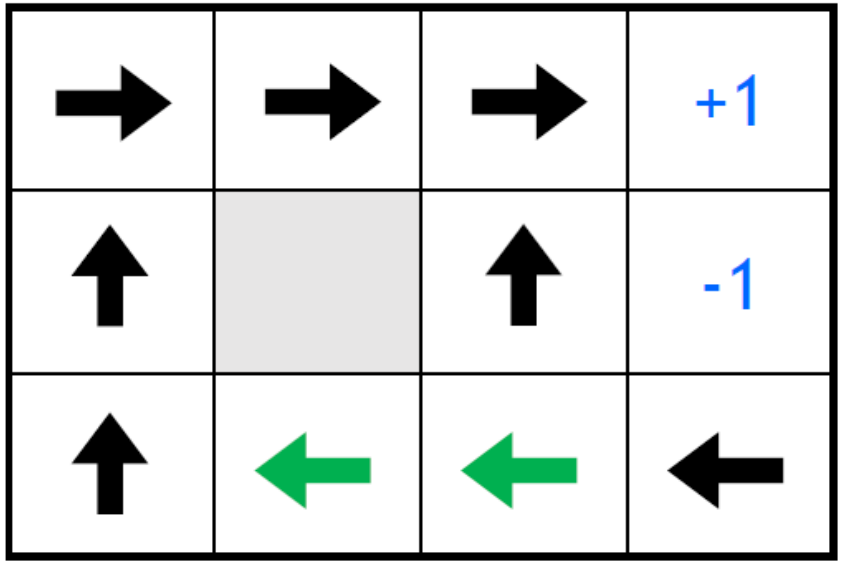
UP

100%	move UP
0%	move LEFT
0%	move RIGHT

Policy: Shortest path.

Optimal Policy for a Stochastic World

Reward: **-0.04** for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are stochastic:

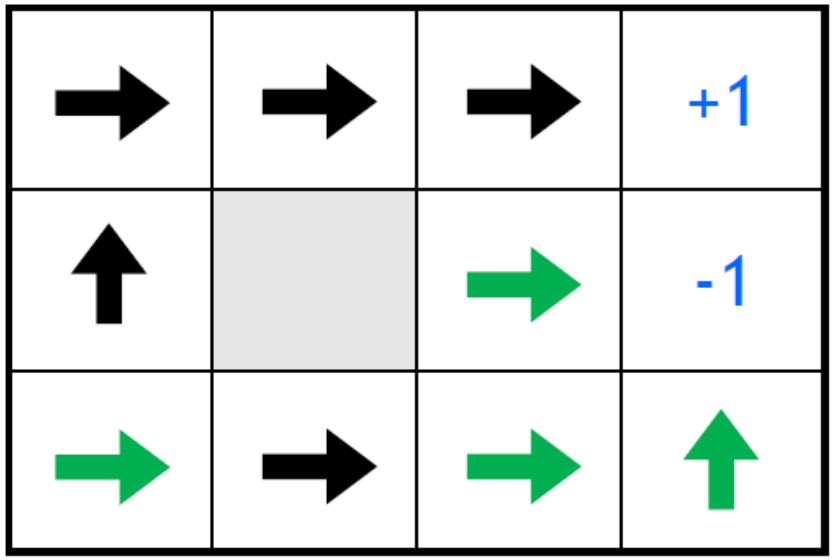
UP

80%	move UP
10%	move LEFT
10%	move RIGHT

Policy: Shortest path. Avoid -UP around -1 square.

Optimal Policy for a Stochastic World

Reward: **-2** for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are stochastic:

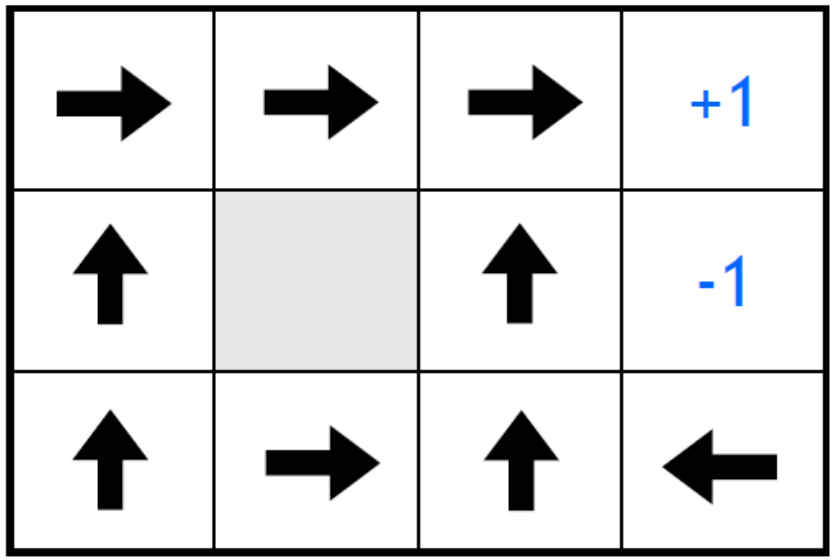
UP

80%	move UP
10%	move LEFT
10%	move RIGHT

Policy: Shortest path.

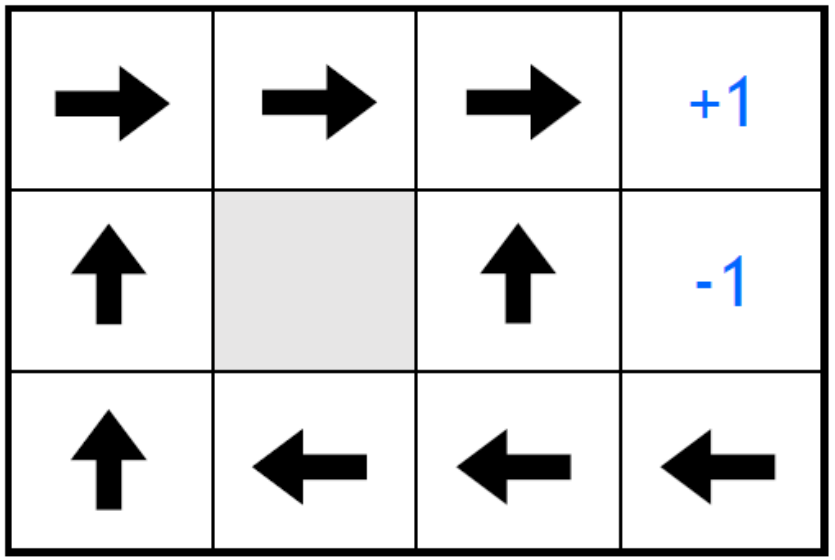
Optimal Policy for a Stochastic World

Reward: **-0.1** for each step



More urgent

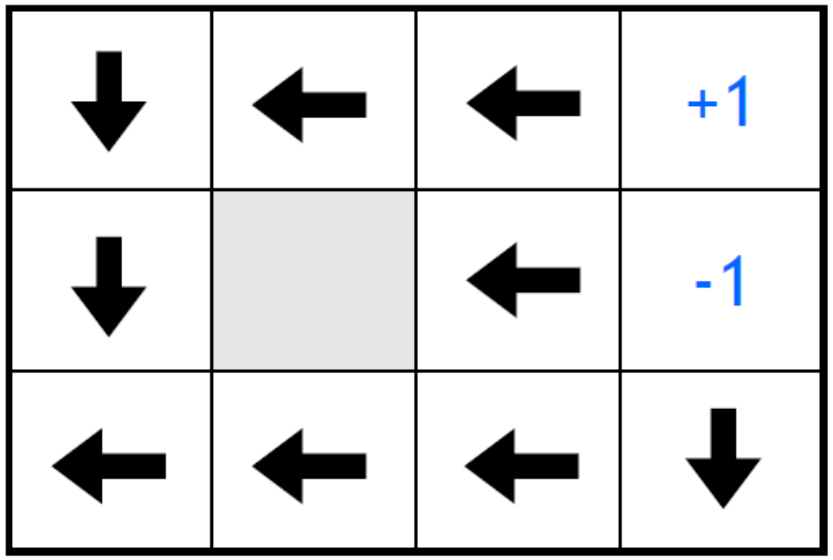
Reward: **-0.04** for each step



Less urgent

Optimal Policy for a Stochastic World

Reward: **+0.01** for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are stochastic:

UP

80%	move UP
10%	move LEFT
10%	move RIGHT

Policy: Longest path.

3 Types of Reinforcement Learning

Model-based

- Learn the model of the world, then plan using the model
- Update model often
- Re-plan often

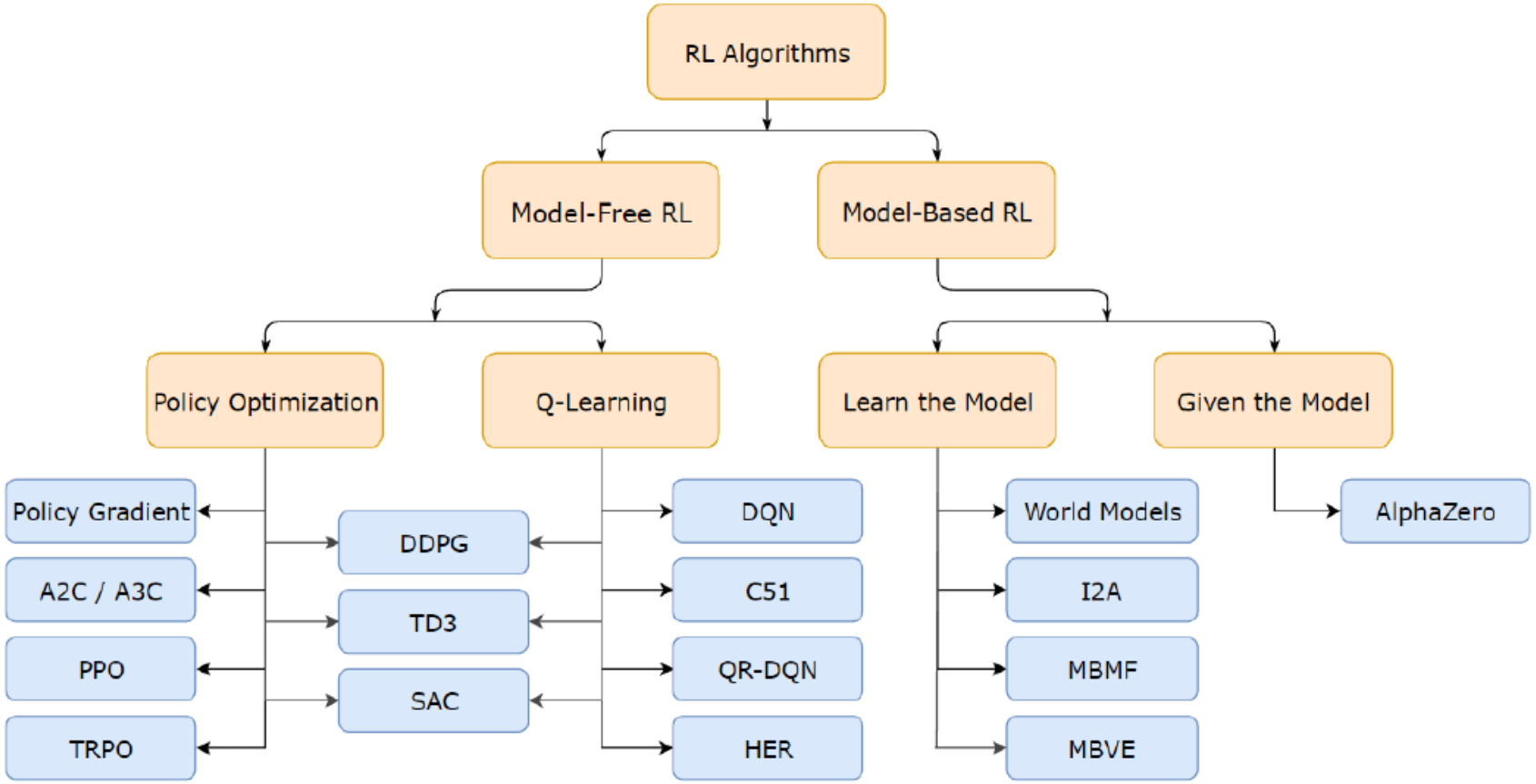
Value-based

- Learn the state or state-action value
- Act by choosing best action in state
- Exploration is a necessary add-on

Policy-based

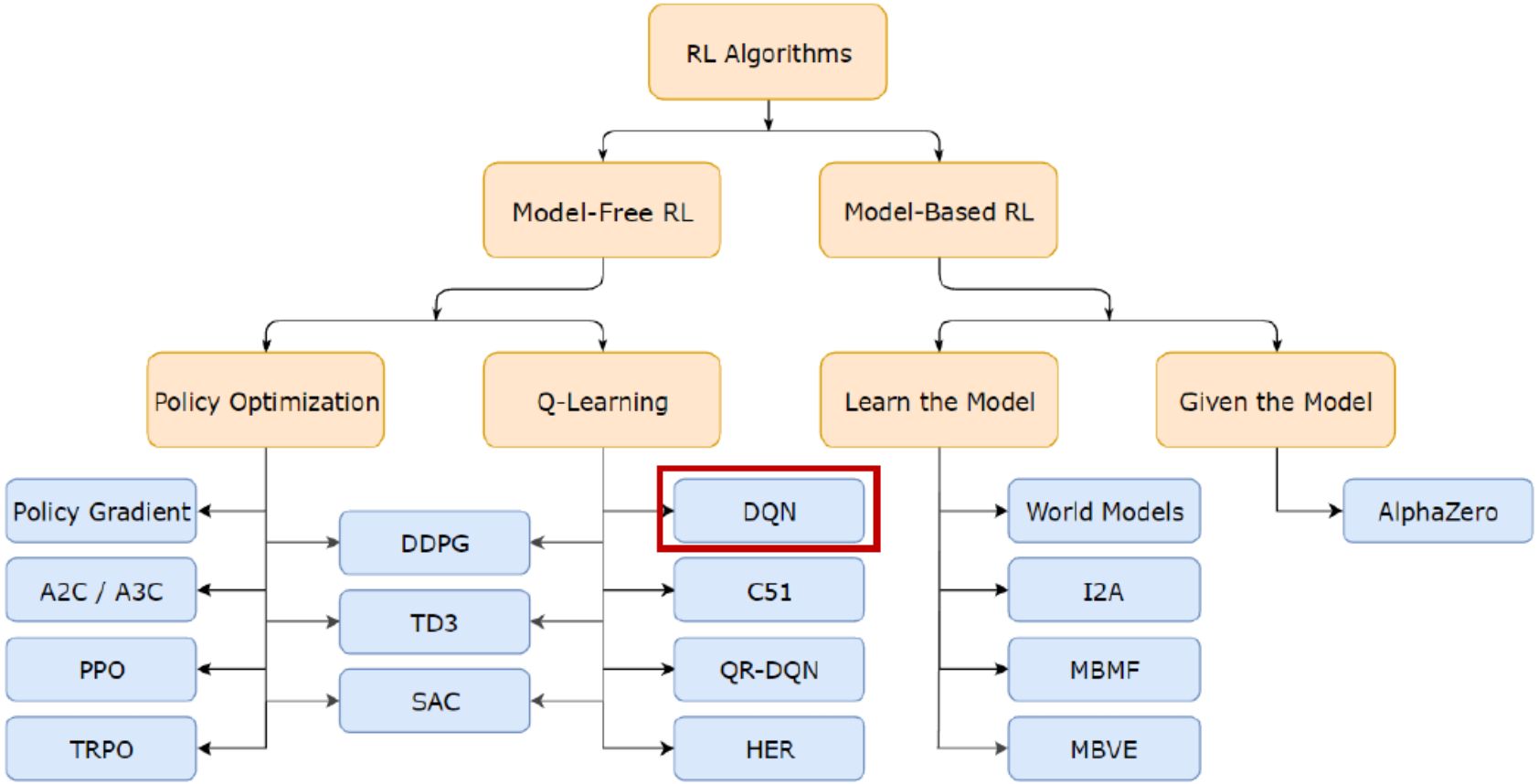
- Learn the stochastic policy function that maps state to action
- Act by sampling policy
- Exploration is baked in

Taxonomy of RL Methods

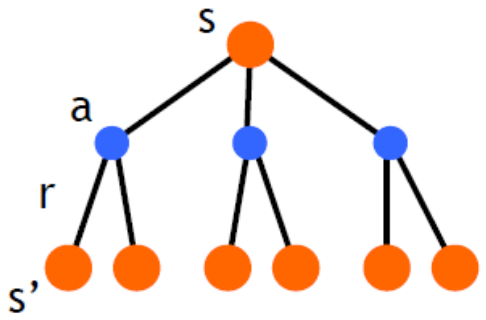


Link: https://spinningup.openai.com

Taxonomy of RL Methods



Q-Learning



- State-action value function: $Q^{\pi}(s,a)$
  - Expected return when starting in *s*, performing *a*, and following π

- Q-Learning: Use **any policy** to estimate Q that maximizes future reward:
  - Q directly approximates Q* (Bellman optimality equation)
  - Independent of the policy being followed
  - Only requirement: keep updating each (s,a) pair

Learning Rate

Discount Factor

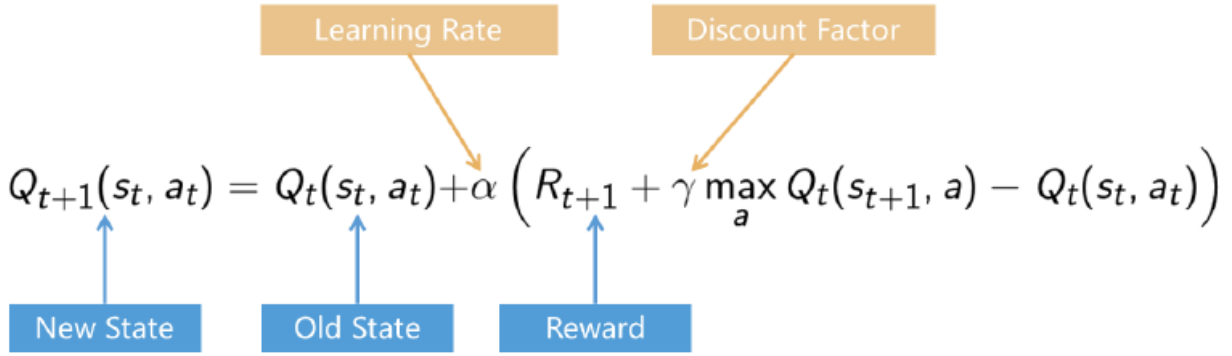
$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right)$$

New State

Old State

Reward

Tabular Q-Learning

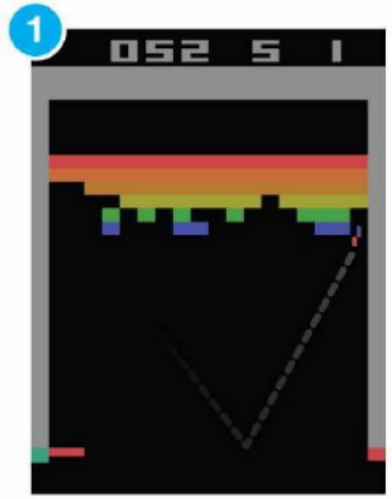


	A1	A2	A3	A4
S1	+1	+2	-1	0
S2	+2	0	+1	-2
S3	-1	+1	0	-2
S4	-2	0	+1	+1

```
initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
      select and carry out an action a
      observe reward r and new state s'
      Q[s,a] = Q[s,a] + α(r + γ max_a' Q[s',a'] - Q[s,a])
      s = s'
until terminated
```

Q-Learning: Representation Matters

- In practice, Tabular Q-Learning is impractical
  - Very limited states/actions
  - Cannot generalize to unobserved states



- Think about the **Breakout** game
  - State: screen pixels
    - Image size: 84×84 (resized)
    - Consecutive 4 images
    - Grayscale with 256 gray levels

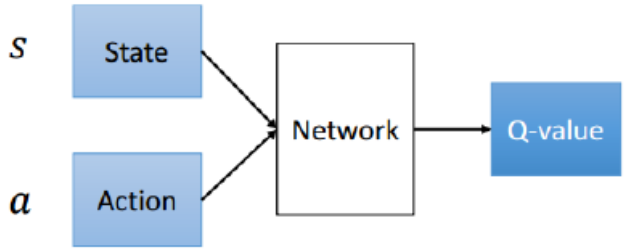
$256^{84\times84\times4}$ rows in the Q-table!

$= 10^{69,970} >> 10^{82}$ atoms in the universe

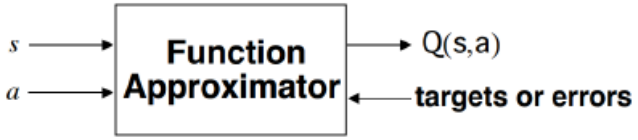
References: [83, 84]

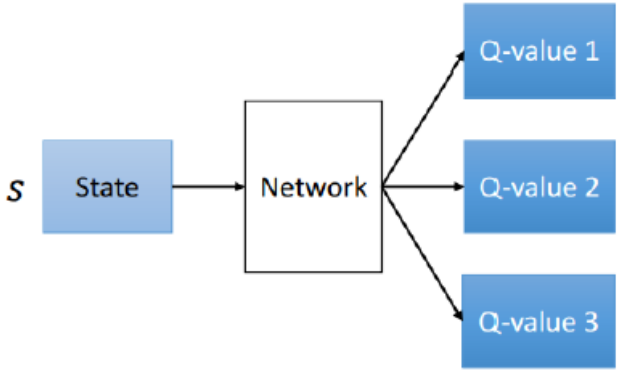
DQN: Deep Q-Learning

Use a neural network to approximate the Q-function:

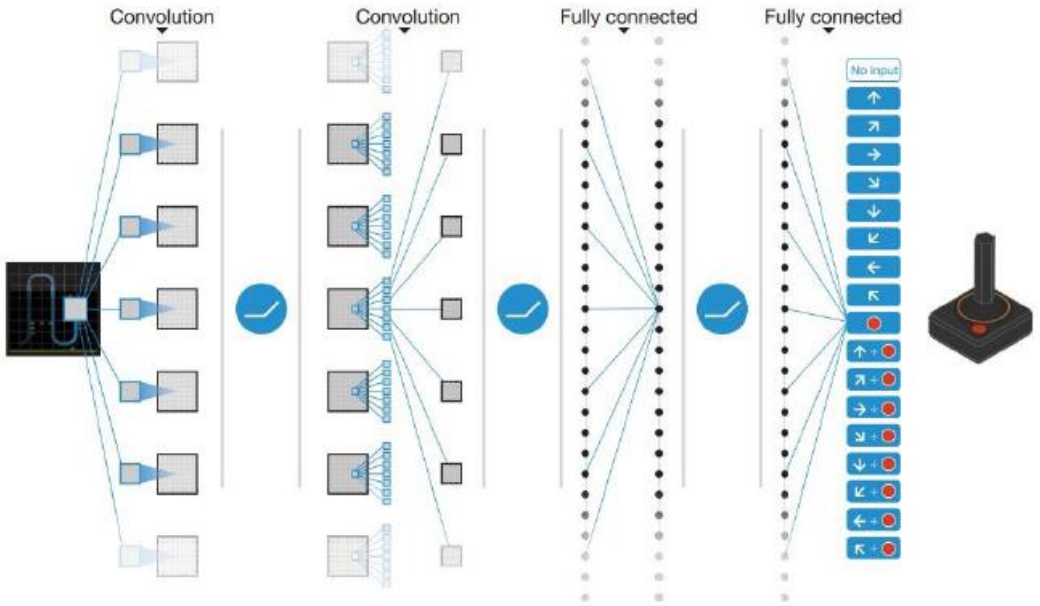


$$Q(s, a; \boldsymbol{\theta}) \approx Q^*(s, a)$$





Deep Q-Network (DQN): Atari



Layer	Input	Filter size	Stride	Num filters	Activation	Output
conv1	84x84x4	8x8	4	32	ReLU	20x20x32
conv2	20x20x32	4x4	2	64	ReLU	9x9x64
conv3	9x9x64	3x3	1	64	ReLU	7x7x64
fc4	7x7x64			512	ReLU	512
fc5	512			18	Linear	18

Mnih et al. "Playing atari with deep reinforcement learning." 2013.

DQN and Double DQN

- Loss function (squared error):

$$L = \mathbb{E}[(\underbrace{\boldsymbol{r + \gamma max_{a'} Q(s', a')}}_{\textbf{target}} - \underbrace{Q(s, a)}_{\textbf{prediction}})^2]$$

- DQN: same network for both Q
- Double DQN: separate network for each Q
  - Helps reduce bias introduced by the inaccuracies of Q network at the beginning of training

DQN Tricks

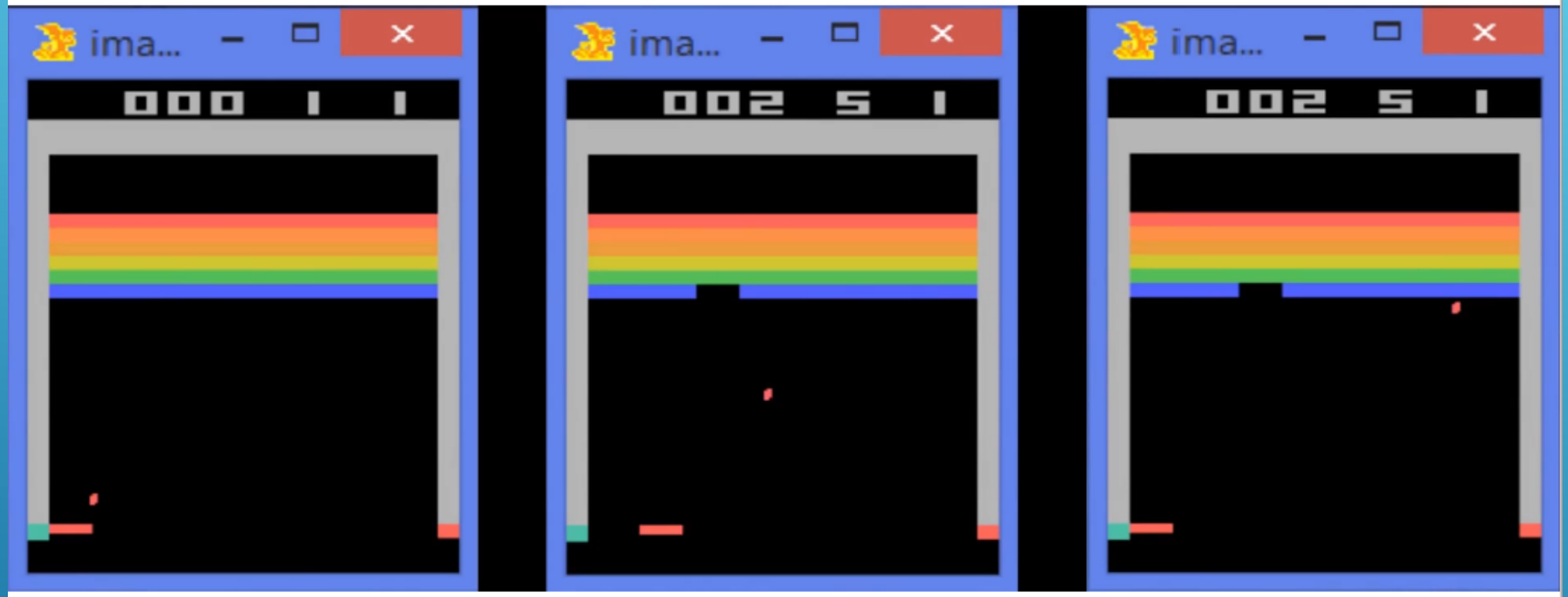
- Experience Replay
  - Stores experiences (actions, state transitions, and rewards) and creates mini-batches from them for the training process
- Fixed Target Network
  - Error calculation includes the target function depends on network parameters and thus changes quickly. Updating it only every 1,000 steps increases stability of training process.

$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha \left[r_{t+1} + \gamma \max_p Q(s_{t+1}, p) - Q(s_t, a) \right]$$

target Q function in the red rectangular is fixed

Replay	○	○	×	×
Target	○	×	○	×
Breakout	**316.8**	240.7	10.2	3.2
River Raid	**7446.6**	4102.8	2867.7	1453.0
Seaquest	**2894.4**	822.6	1003.0	275.8
Space Invaders	**1088.9**	826.3	373.2	302.0

Atari Breakout

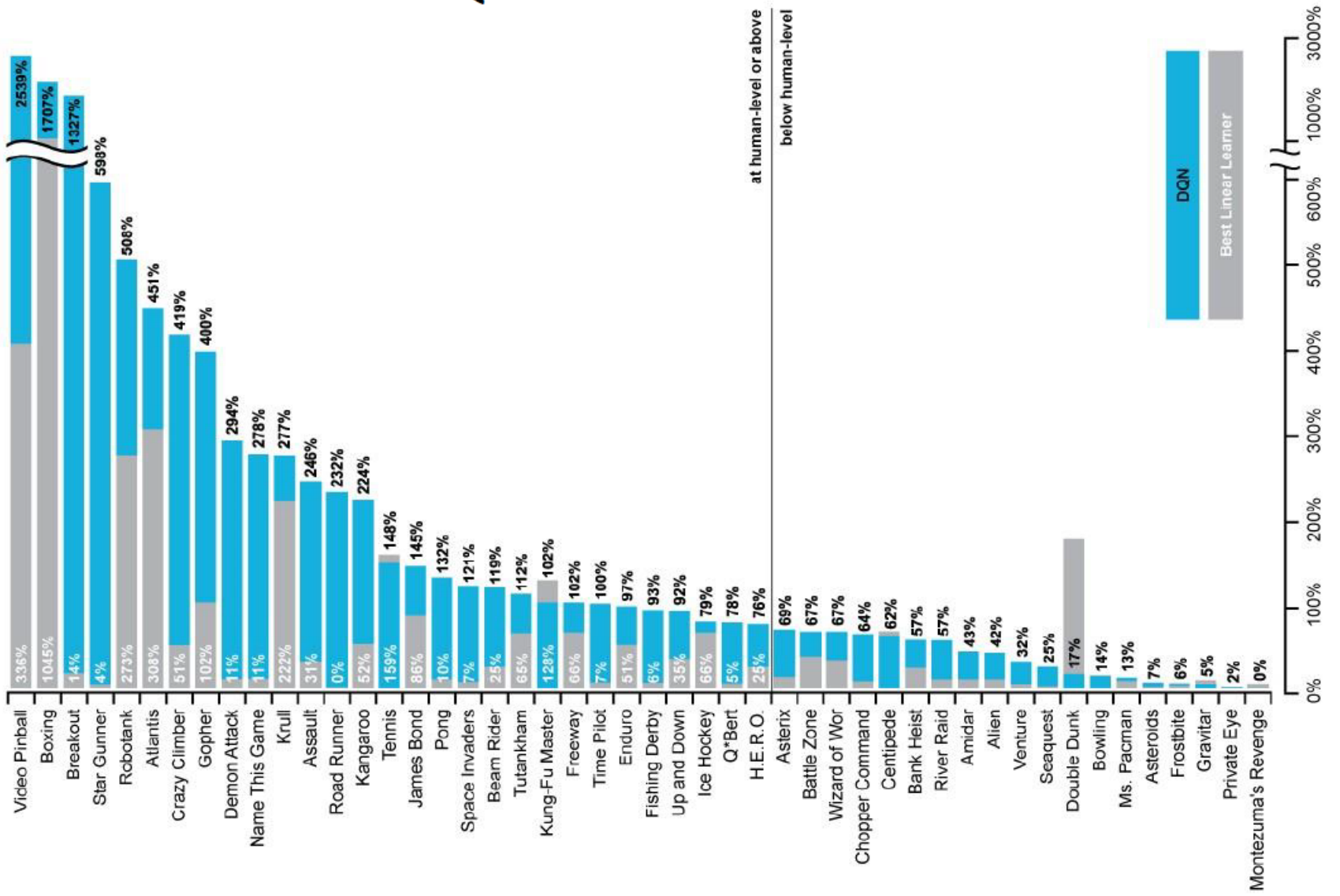


After
10 Minutes
of Training

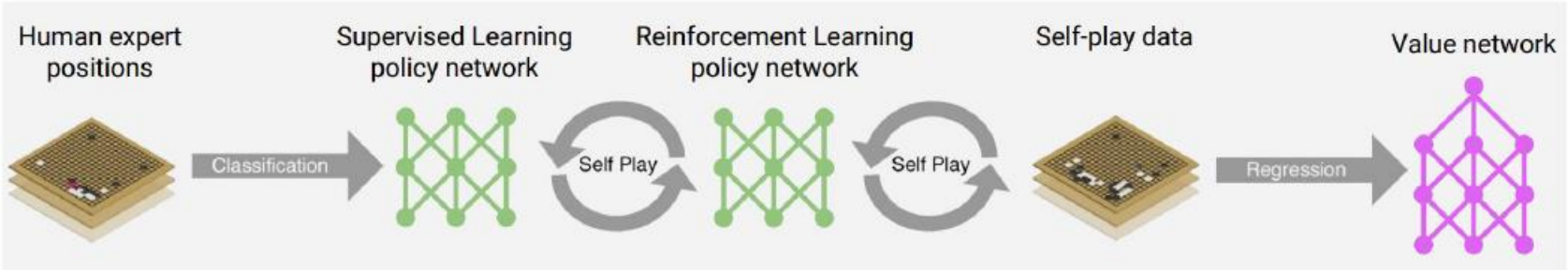
After
120 Minutes
of Training

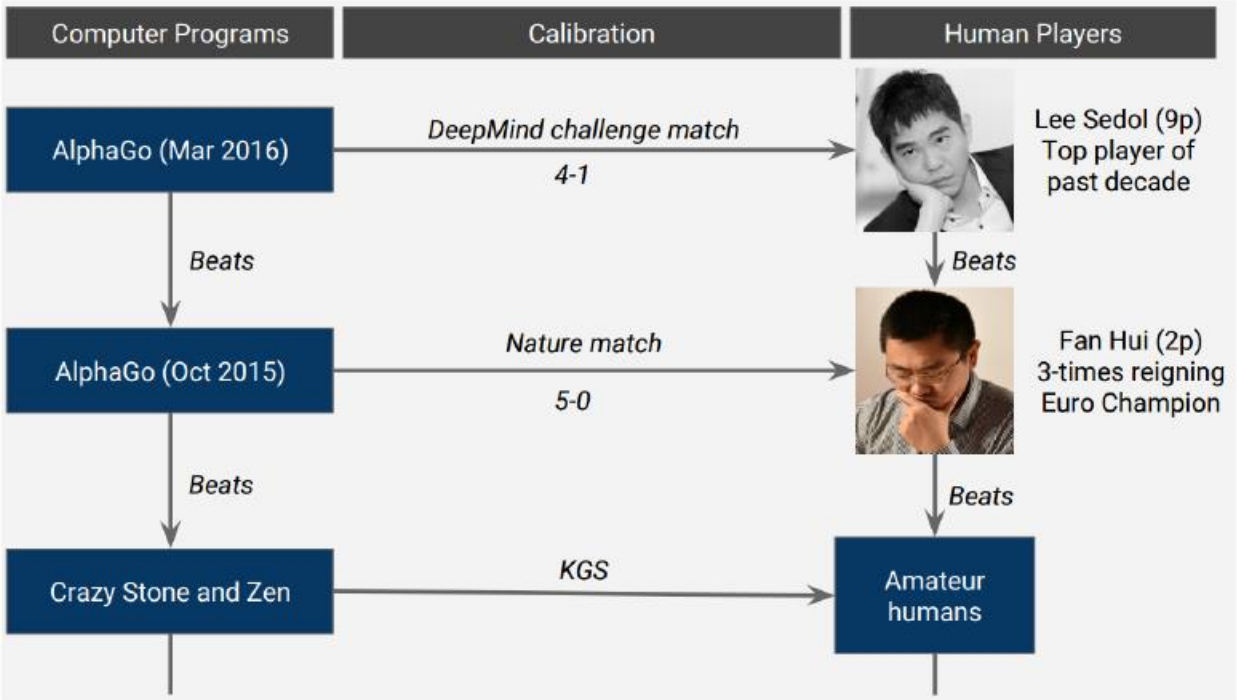
After
240 Minutes
of Training

DQN Results in Atari



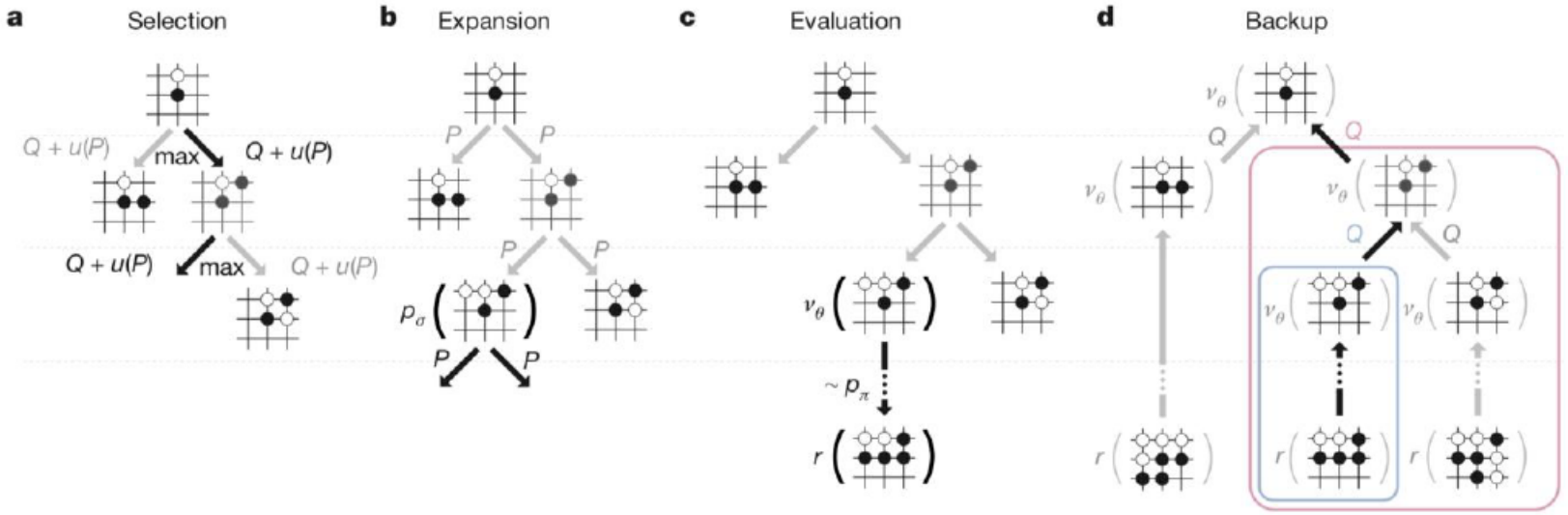
AlphaGo (2016) Beat Top Human at **Go**



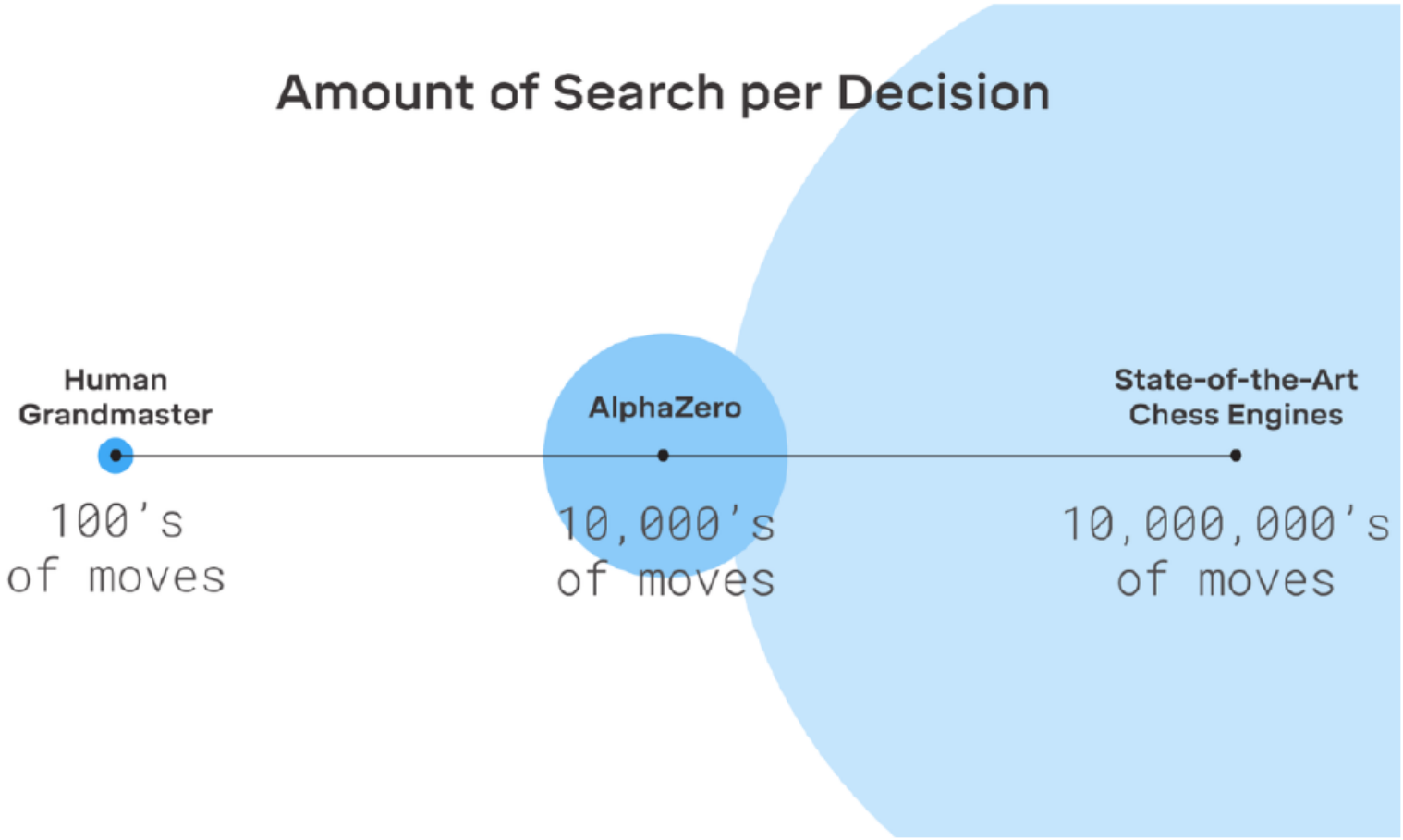


AlphaGo Zero Approach

- Same as the best before: Monte Carlo Tree Search (MCTS)
  - Balance exploitation/exploration (going deep on promising positions or exploring new underplayed positions)
- Use a neural network as "intuition" for which positions to expand as part of MCTS (same as AlphaGo)



AlphaZero vs Chess, Shogi, Go



AlphaZero vs Chess, Shogi, Go

Chess

AlphaZero vs. Stockfish

W:29.0% D:70.6% L:0.4%

W:2.0% D:97.2% L:0.8%

Shogi

AlphaZero vs. Elmo

W:84.2% D:2.2% L:13.6%

W:98.2% D:0.0% L:1.8%

Go

AlphaZero vs. AG0

W:86.9% L:31.1%

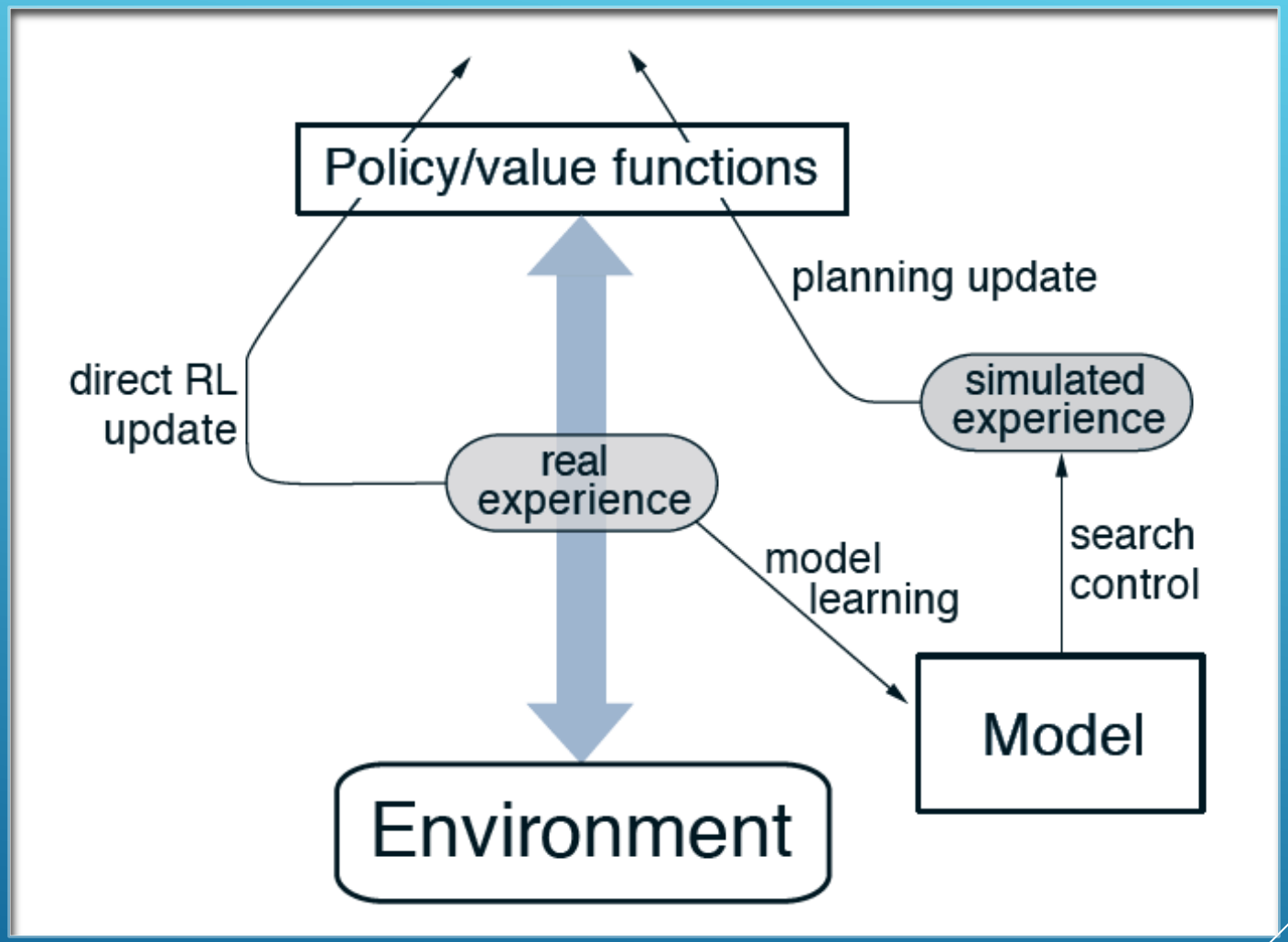
W:53.7% L:46.3%

AZ wins AZ draws AZ loses AZ white AZ black

DOES AGI = DNN + RL?

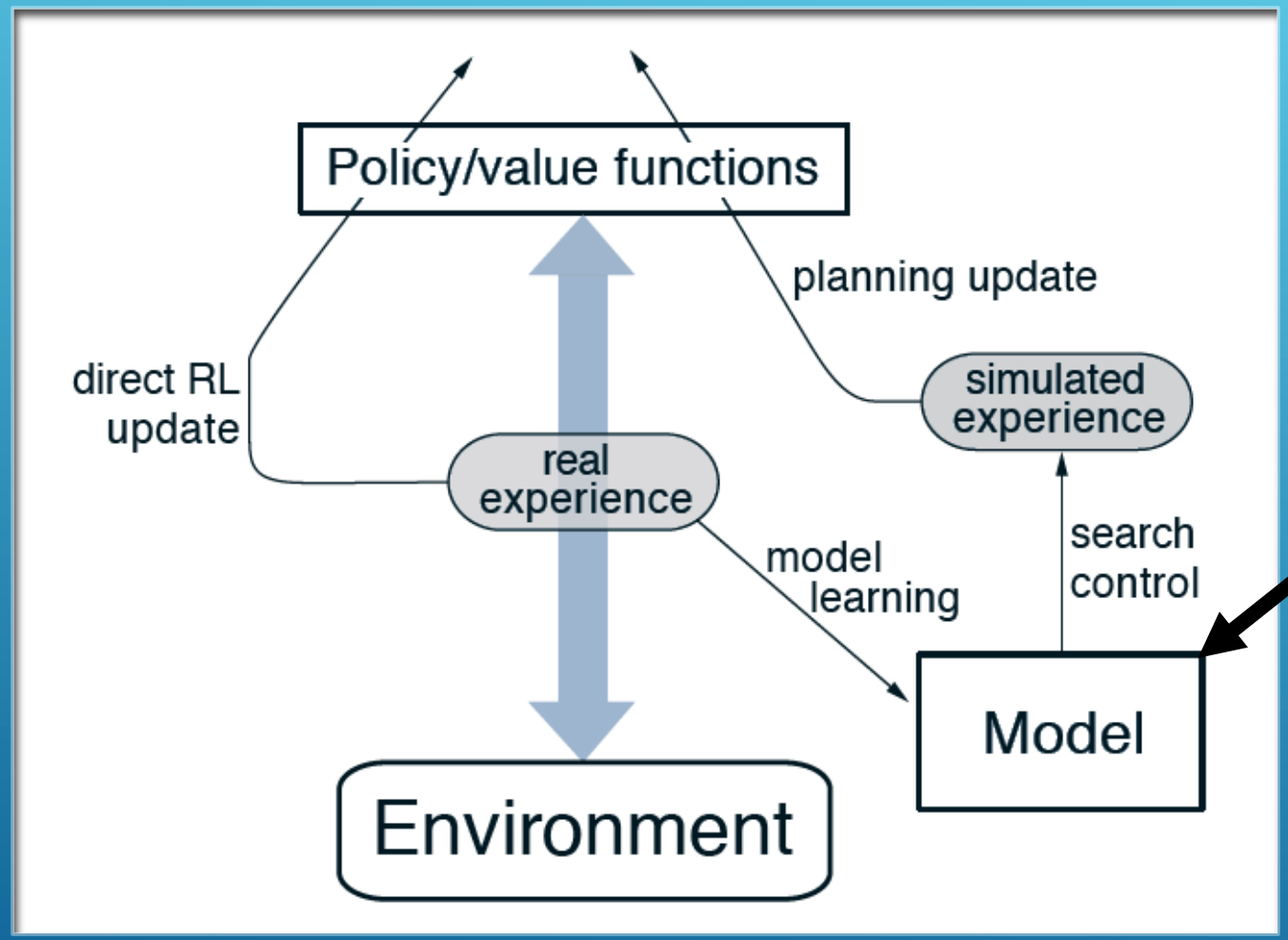
Scott O'Hara
Metrowest Developers Machine Learning Group
5/27/2020

MODEL-BASED REINFORCEMENT LEARNING



Credit: image borrowed from lecture slides David Silver, DeepMind, "Introduction to Reinforcement Learning

WHAT KIND OF MODEL TO USE?



Current Approaches

- Tables
- Linear Functions
- Deep Neural Networks

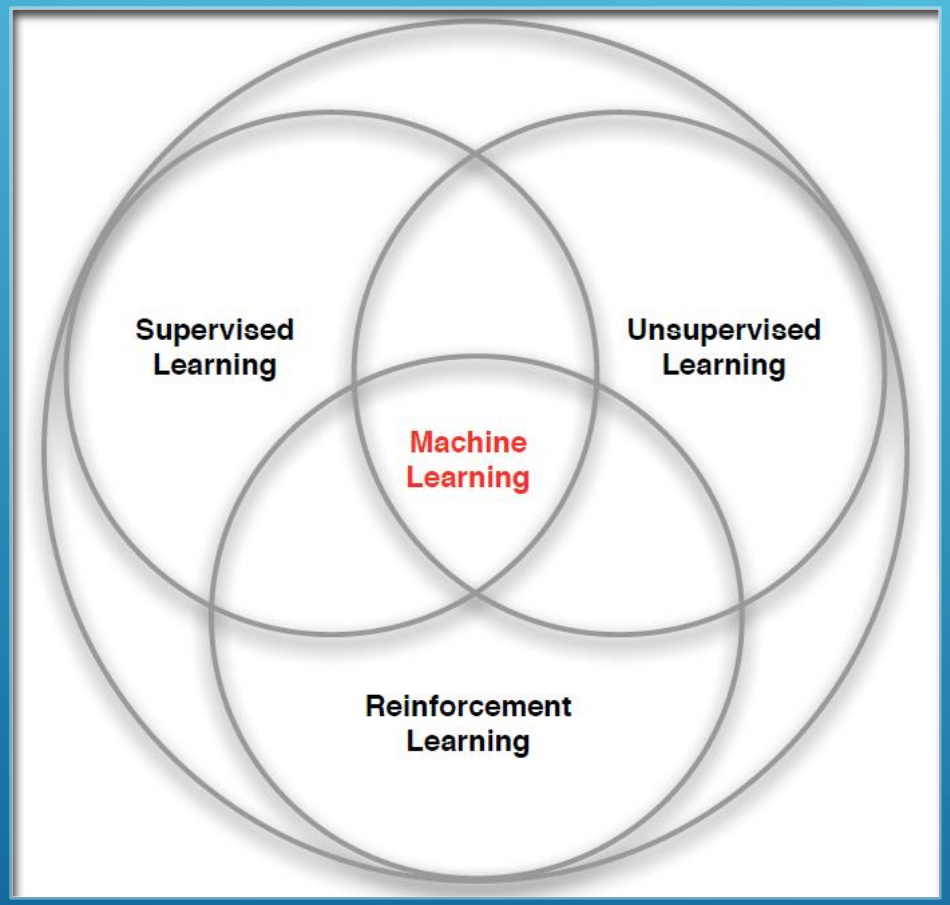
But also:

- Bayesian Networks
- Scripts (a'la Schank)
- Frames (a'la Minsky)
- Cognitive Models
- Long-term Memory
- Innate Knowledge

Credit: image borrowed from lecture slides David Silver, DeepMind, "Introduction to Reinforcement Learning

EXTRA SLIDES

3 TYPES OF MACHINE LEARNING



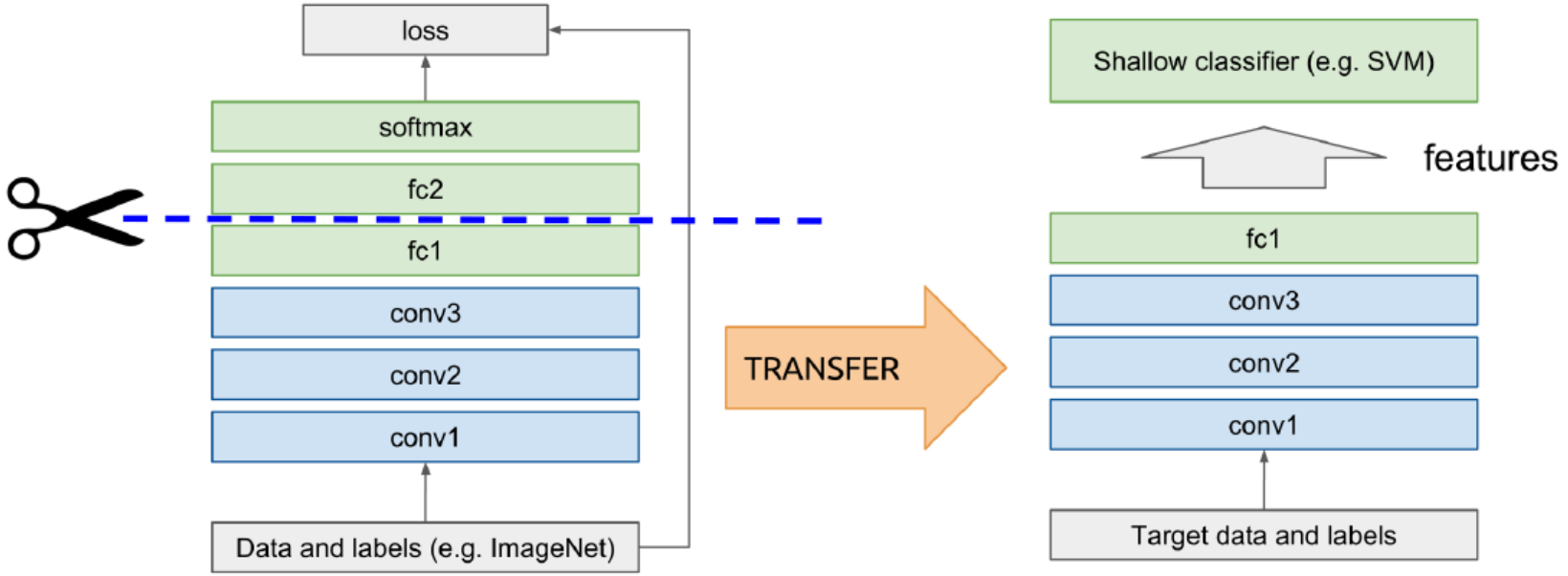
Supervised Learning – Learn a function from labeled data that maps input attributes to an output label e.g., linear regression, decision trees, SVMs.

Unsupervised Learning – Learn patterns in unlabeled data e.g., principle component analysis or clustering algorithms such as K-means, HAC, or Gaussian mixture models.

Reinforcement Learning – An agent learns to maximize rewards while acting in an uncertain environment.

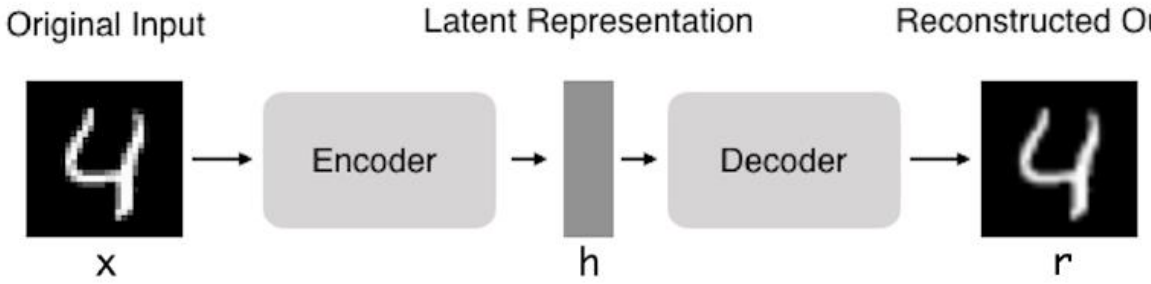
Credit: image borrowed from lecture slides David Silver, DeepMind, "Introduction to Reinforcement Learning

Transfer Learning

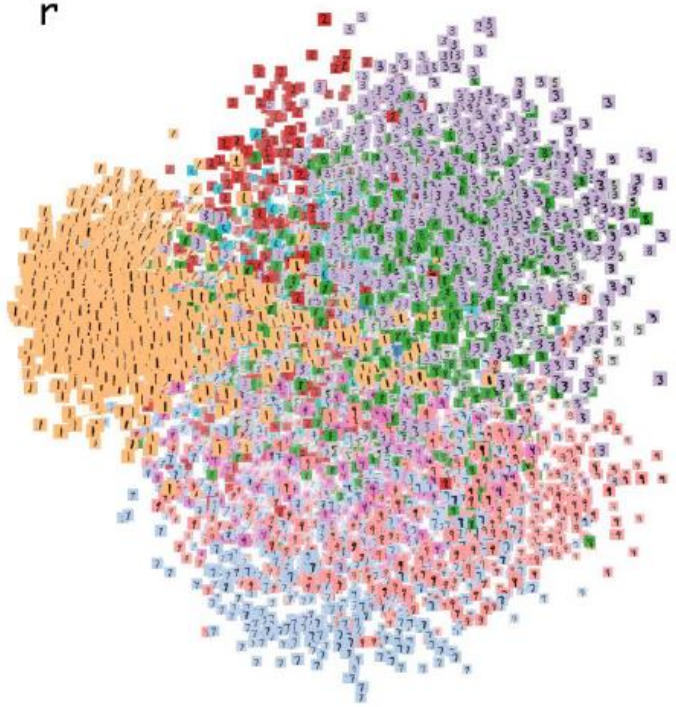


- Fine-tune a pre-trained model
- Effective in many applications: computer vision, audio, speech, natural language processing

Autoencoders



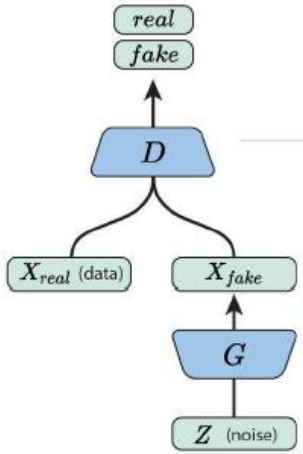
- Unsupervised learning
- Gives embedding
  - Typically better embeddings come from discriminative task



http://projector.tensorflow.org/

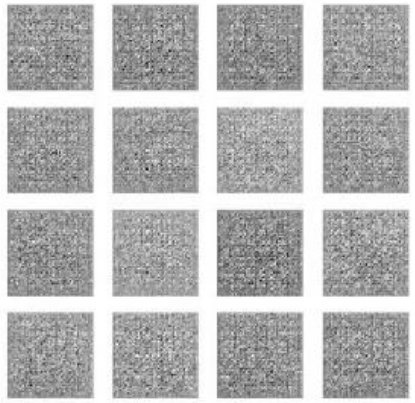
Generative Adversarial Network (GANs)

Generative Adversarial Networks (GANs) are a way to make a generative model by having two neural networks compete with each other.



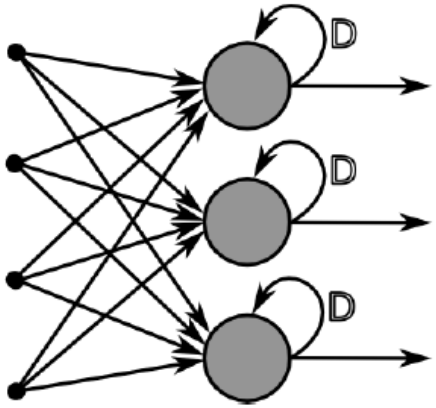
The **discriminator** tries to distinguish genuine data from forgeries created by the generator.

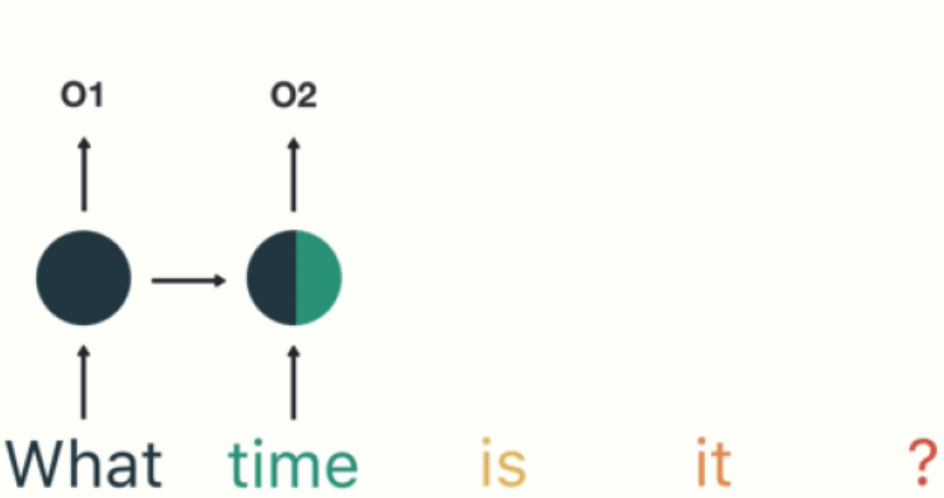
The **generator** turns random noise into immitations of the data, in an attempt to fool the discriminator.



Progressive GAN
10/2017
1024 x 1024

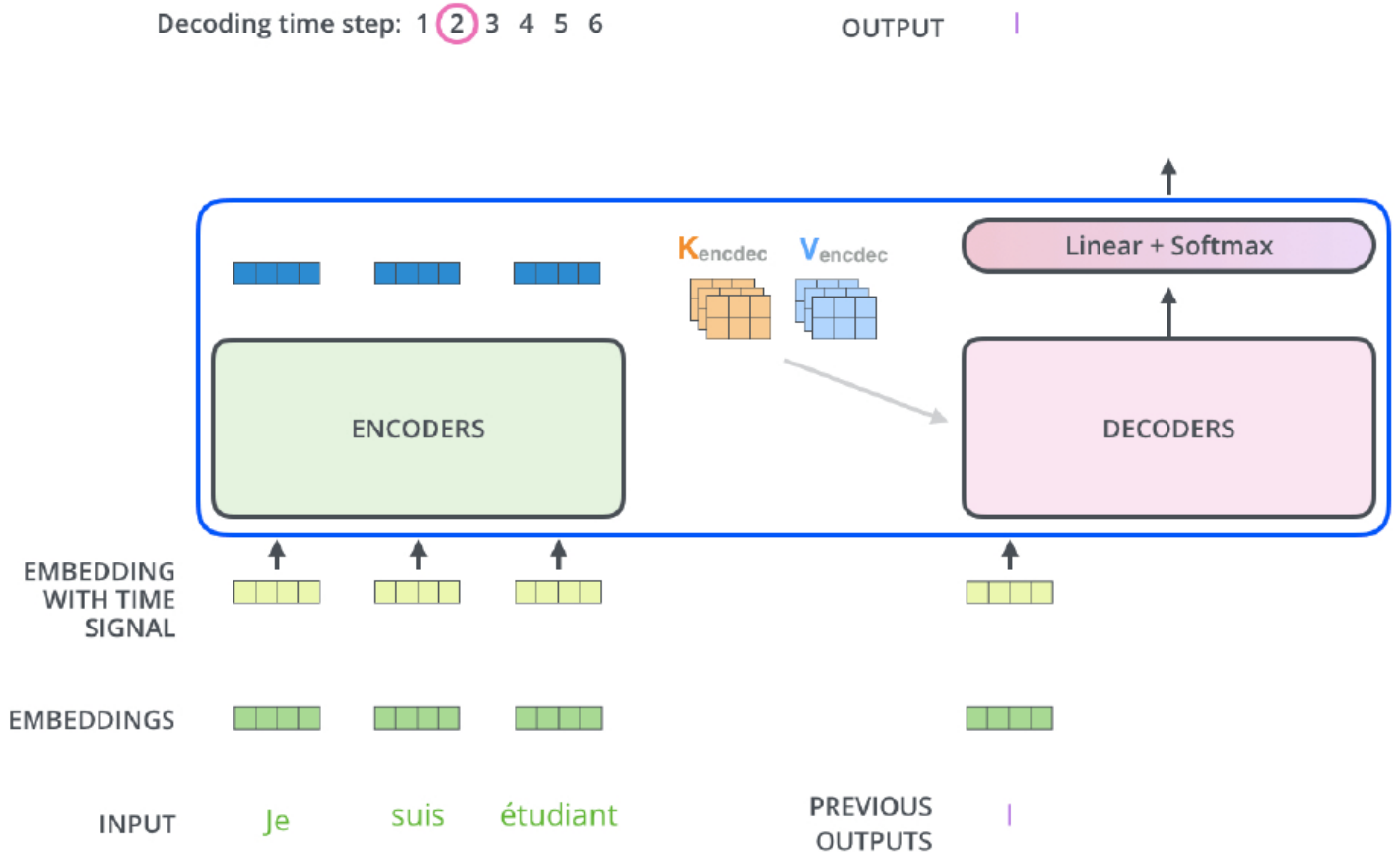
Recurrent Neural Networks





- Applications
  - Sequence Data
  - Text
  - Speech
  - Audio
  - Video
  - Generation

Transformer



Vaswani et al. "Attention is all you need." *Advances in Neural Information Processing Systems*. 2017.