CPEN400D Course In a Nutshell

Goal was to give you a strong understanding of the foundations of Deep Learning so that by the end of this course you can

- Learn high-level frameworks (e.g. TensorFlow, PyTorch, etc) and know the theory behind them to
- Build actual deep learning solutions and applications
- Continue your own studies of more advanced topics like being able to read research papers and follow along
- Quite possibly go on to invent something cool in Deep Learning!

Generative Models Reinforcement Learning

Deep Learning

Brad Quinton, Scott Chin

Oveerview

Super super lite intro (and not on the final exam):

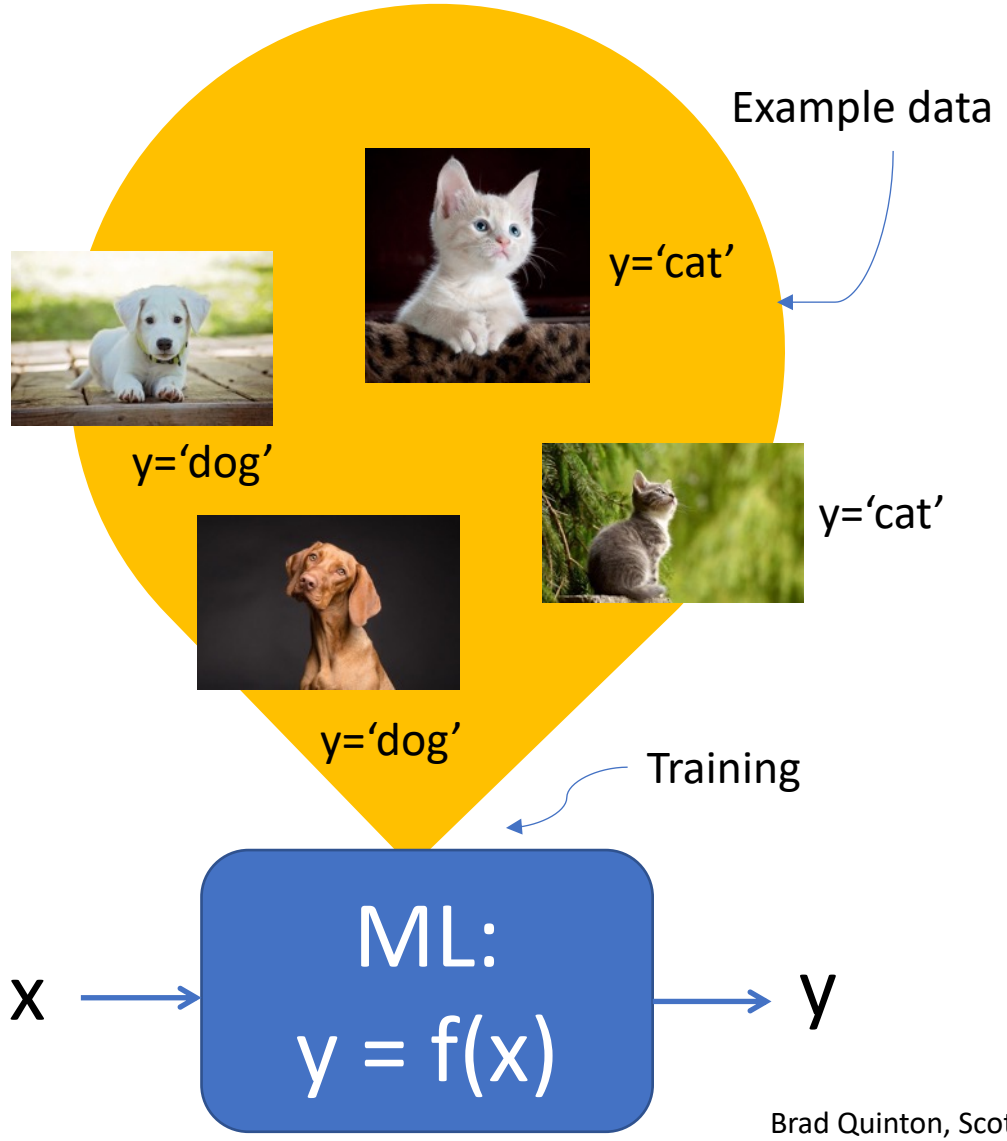
- Generative Models
- Reinforcement Learning

So far… only talked about Supervised Learning

- Data is labeled

Objective:

- Learn a mapping function between input and label



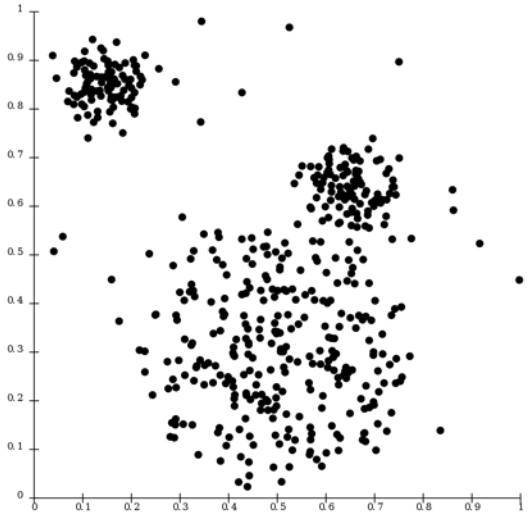
Unsupervised Learning

- Data has no labels

Objective

- Learn an underlying relationship

- Examples
- Clustering, dimensionality reduction, outlier detection, autoencoders, generative models



Clustering:
Find groupings of similar data

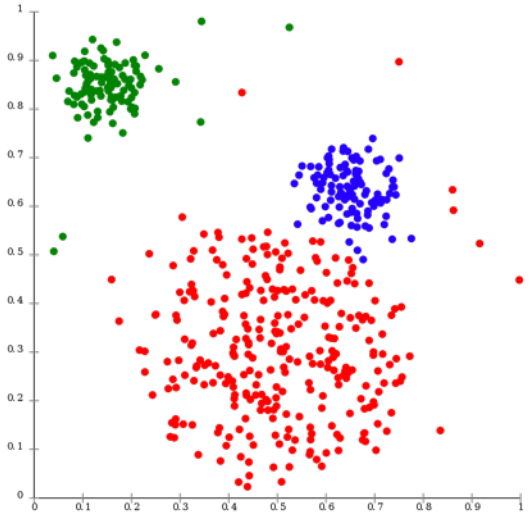


 Image source: https://en.wikipedia.org/wiki/Cluster_analysis#/media/File:OPTICS-Gaussian-data.svg

Generative Models in Deep Learning

- So far, we have talked mostly about discriminative/predictive models i.e. given an input, predict from which distribution (i.e. class) the input came
- Can think of each class as having some hidden underlying function that can output all possible members (e.g. images) of its class

Data as output of unknown functions

Example: Handwritten Digit Classification

- Consider all the ways that the number 1 can be written in a 28x28 grayscale image.
- Now consider that there is an unknown function that can produce all of these as its outputs and nothing else.
- Now consider that each of the other 9 digits has its own unknown function that can only output examples from its own class
- Therefore, you can consider your training data as samples of the output of their respective functions

Data as output of unknown functions

- Classification's goal is to discriminate between these functions
  i.e. given an input, predict from which function it was generated
- Your classifier needs to learn the nuances of these functions
- The more your training data can sample these nuances/variations, the more generalizable your model will be
- Under this view, you can imagine that some classes have more variation than others (i.e. more complex function).
  - E.g. consider the class of cat images versus class of handwritten digit 2.
- The neural network needs enough capacity to learn all the important nuances

Discriminative Models vs Generative Models

Discriminative/Predictive Models (e.g. a classifier)

- Learns to discriminate between the underlying function of each output class, and then be able to predict from which class's function it would have been generated.
- Learns an (approx.) discriminating function

Discriminative Models vs Generative Models

Discriminative/Predictive Models (e.g. a classifier)

- Learns to discriminate between the underlying function of each output class, and then be able to predict from which class's function it would have been generated.
- Learns an (approx.) discriminating function

Generative Models

- Learns the (approx.) underlying generating function directly.
- Can then use the model to generate new samples

Discriminative Models vs Generative Models

- Discriminative models try to learn boundaries in the feature space
- Generative models try to learn where data lies in the feature space

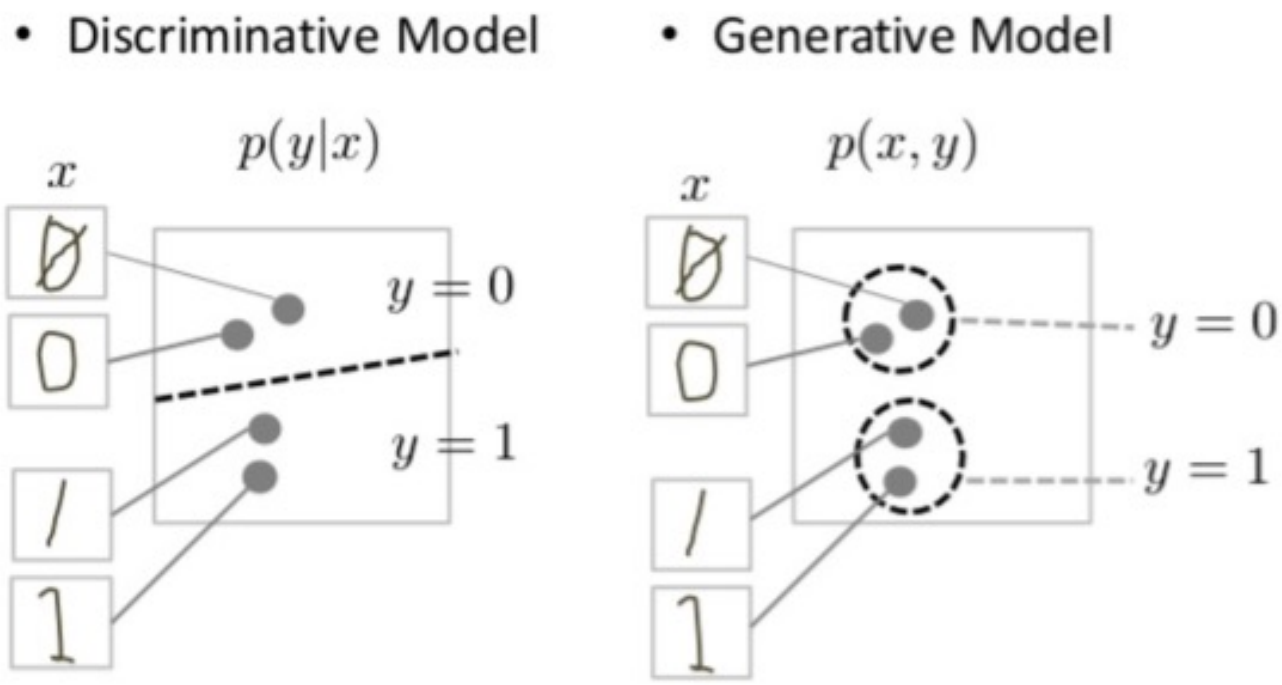


Image Source: https://developers.google.com/machine-learning/gan/generative

Generative Models are Harder to Learn

- A generative model for images might capture correlations like "things that look like boats are probably going to appear near things that look like water" and "eyes are unlikely to appear on foreheads." These are very complicated distributions.
- A discriminative model might learn the difference between "sailboat" or "not sailboat" by just looking for a few tell-tale patterns. It could ignore many of the correlations that the generative model must get right.

https://developers.google.com/machine-learning/gan/generative

Generative Adversarial Networks (GANs)

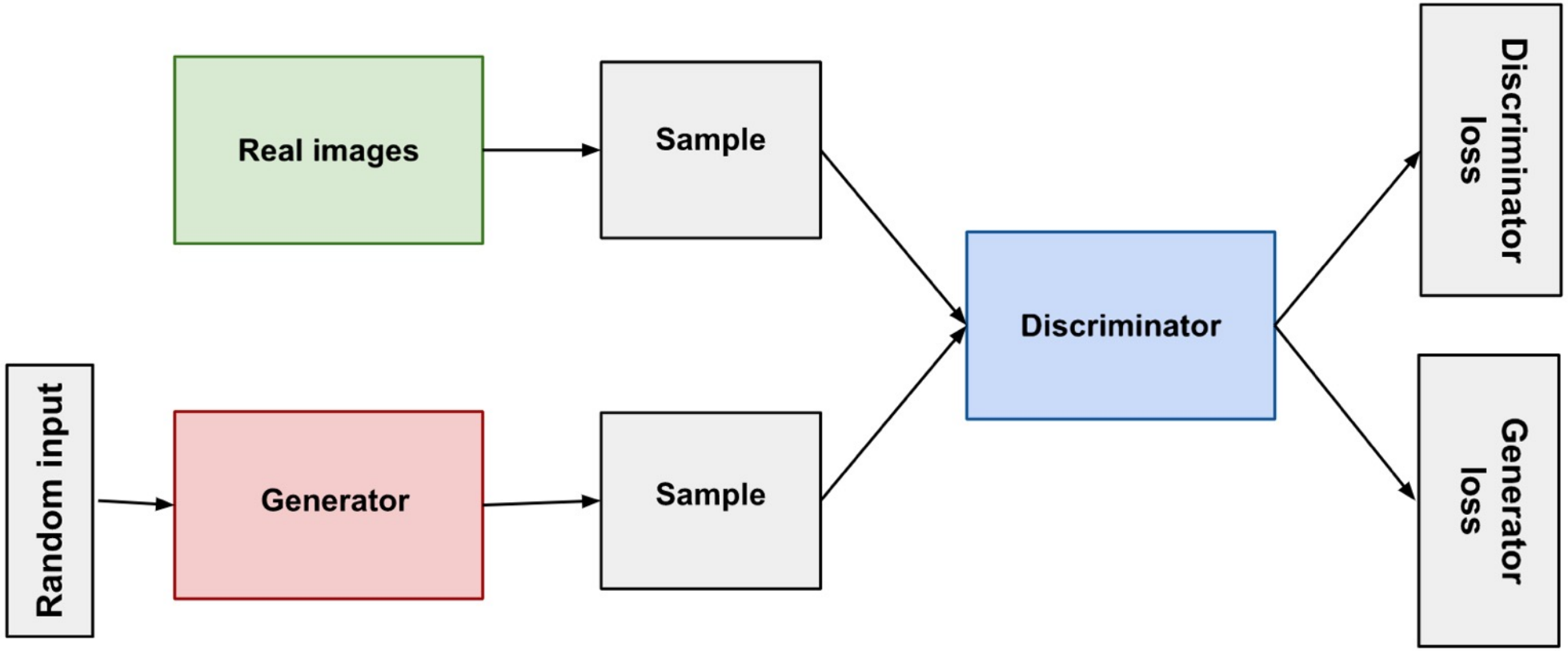
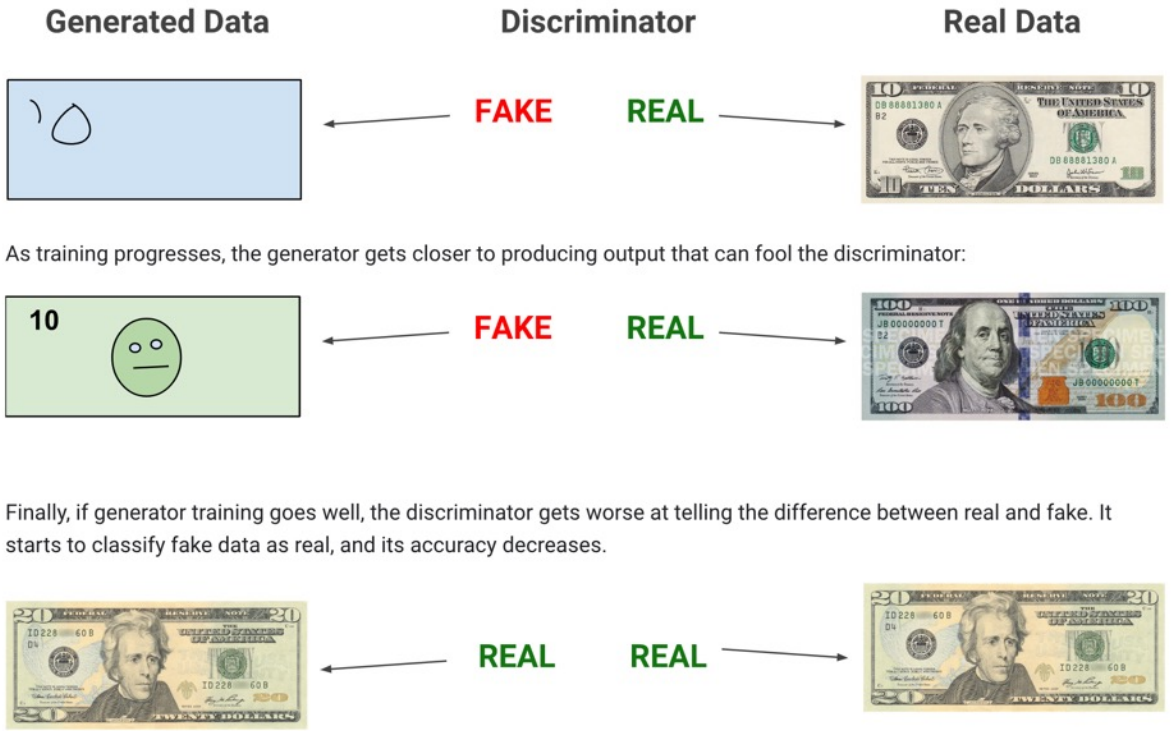


Image Credit: https://developers.google.com/machine-learning/gan

Generative Adversarial Networks (GANs)

- Generator Model: Learns to generate realistic data
- Discriminator Model: Learns to predict whether input is real or fake



"Generative Adversarial Networks", Goodfellow et al, 2014, https://arxiv.org/abs/1406.2661

Image Credit: https://developers.google.com/machine-learning/gan

Training the Discriminator

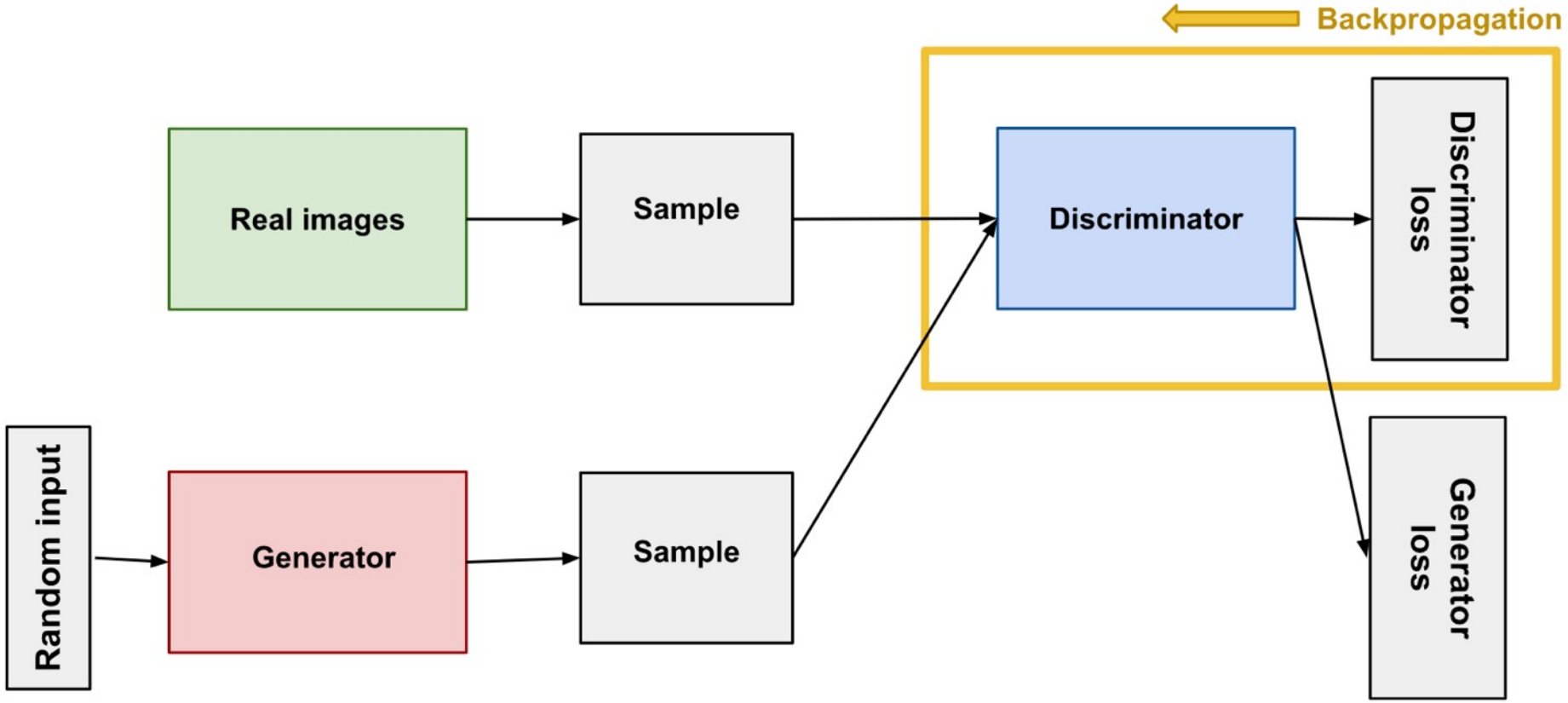


Image Credit: https://developers.google.com/machine-learning/gan

Training the Generator

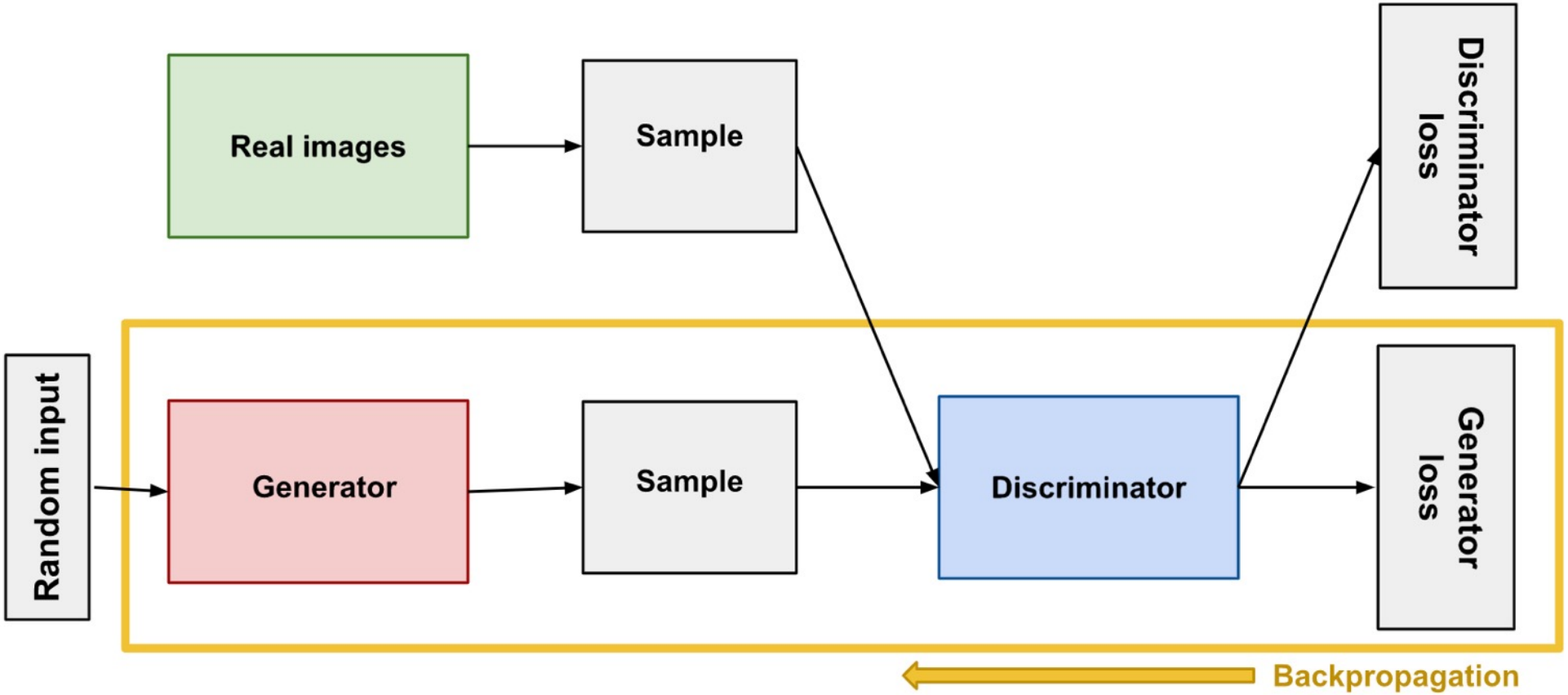
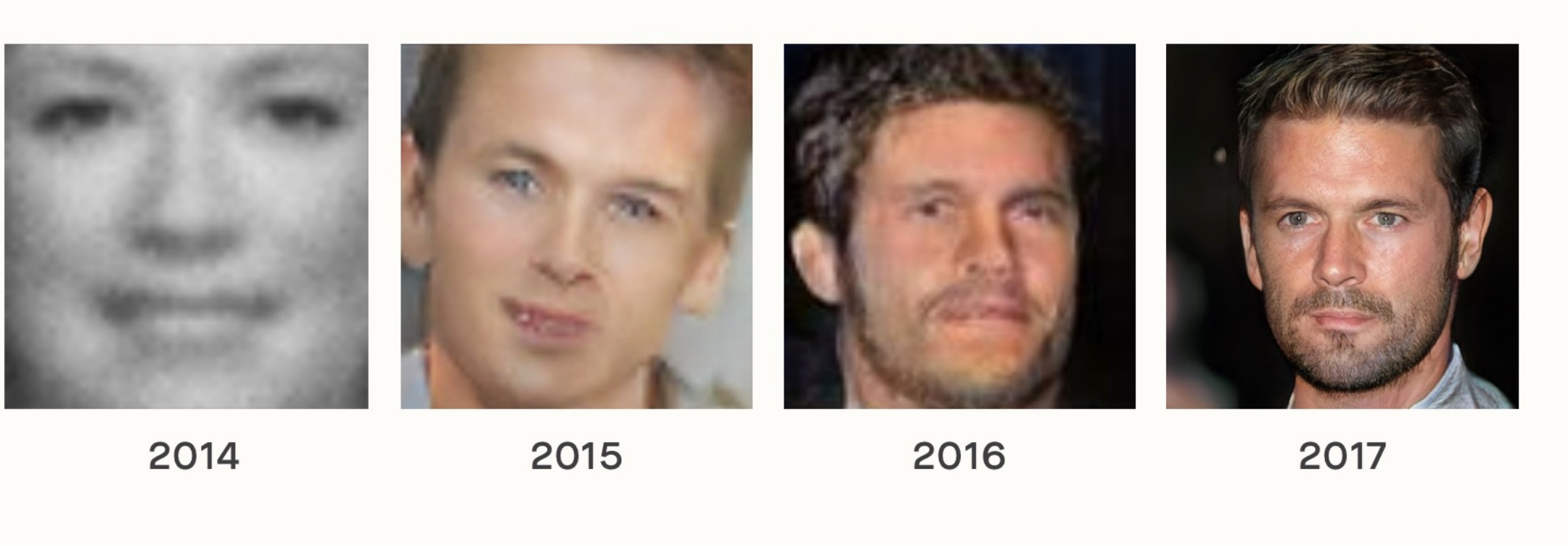


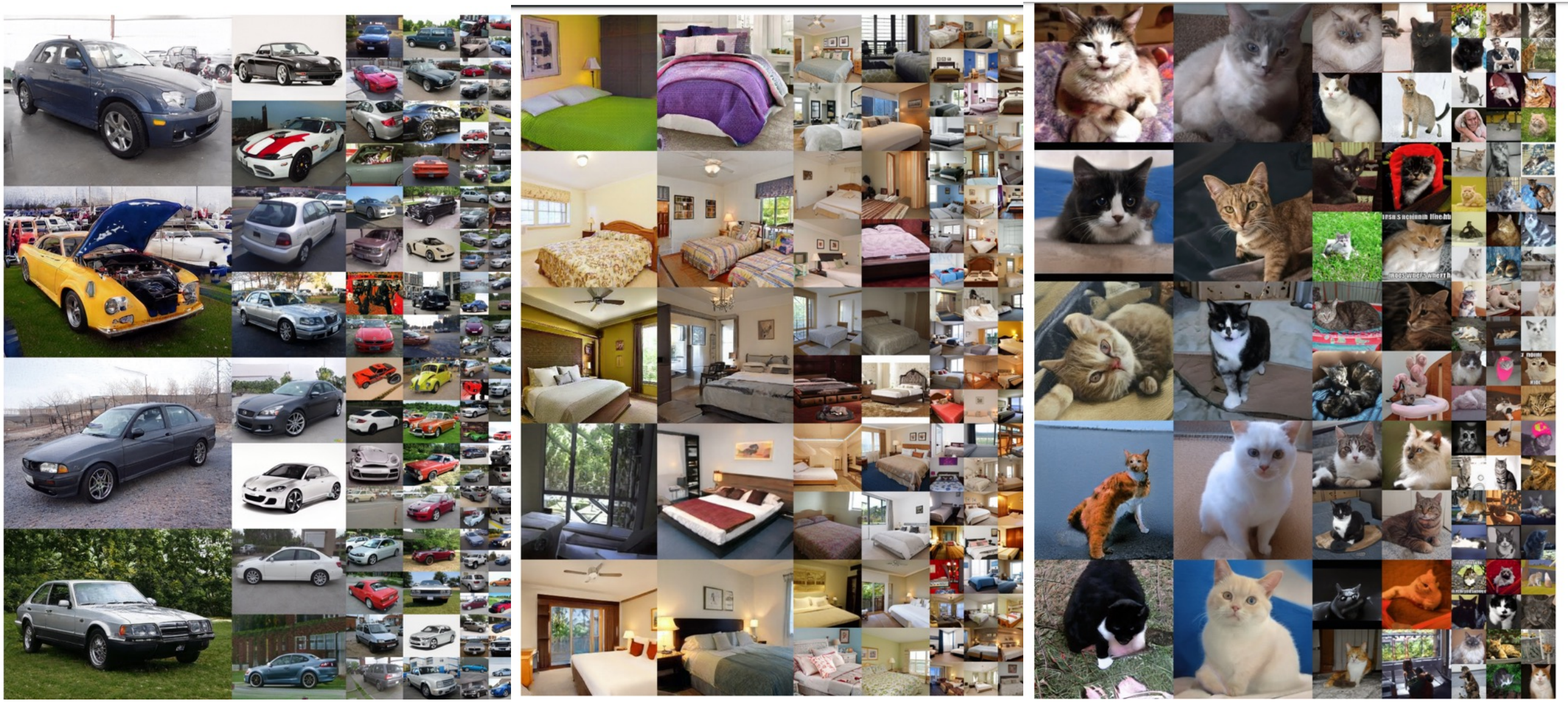
Image Credit: https://developers.google.com/machine-learning/gan

Faces Generated via GANs



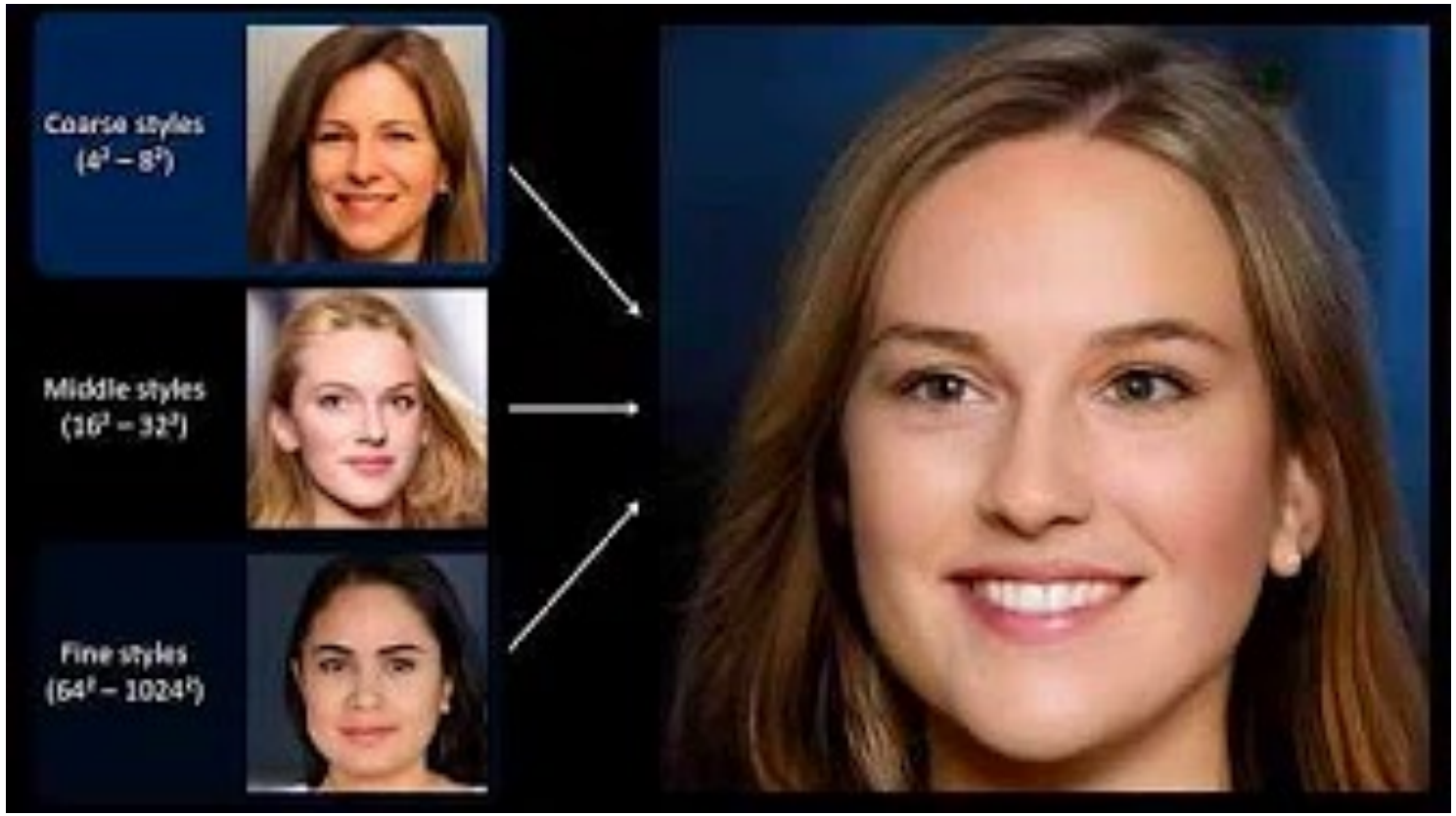
"The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation", Brundage et al, 2018, https://arxiv.org/abs/1802.07228
"Generative Adversarial Networks", Goodfellow et al, 2014, https://arxiv.org/abs/1406.2661
"Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", Radford et al, 2015, https://arxiv.org/abs/1511.06434
"Coupled Generative Adversarial Networks", Liu and Tuzel, 2016, https://arxiv.org/abs/1606.07536
"Progressive Growing of GANs for Improved Quality, Stability, and Variation", Karras et al, 2017, https://arxiv.org/pdf/1710.10196.pdf

 "A Style-Based Generator Architecture for Generative Adversarial Networks", Karras et al, 2018, https://arxiv.org/abs/1812.04948



"A Style-Based Generator Architecture for Generative Adversarial Networks", Karras et al, 2018, https://arxiv.org/abs/1812.04948

A Style-Based Generator Architecture for Generative Adversarial Networks



https://www.youtube.com/watch?v=kSLJriaOumA

Applied to Video

"Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", Zhu et al, 2018, https://arxiv.org/abs/1703.10593

Deep Fakes

https://www.youtube.com/watch?v=cQ54GDm1eL0

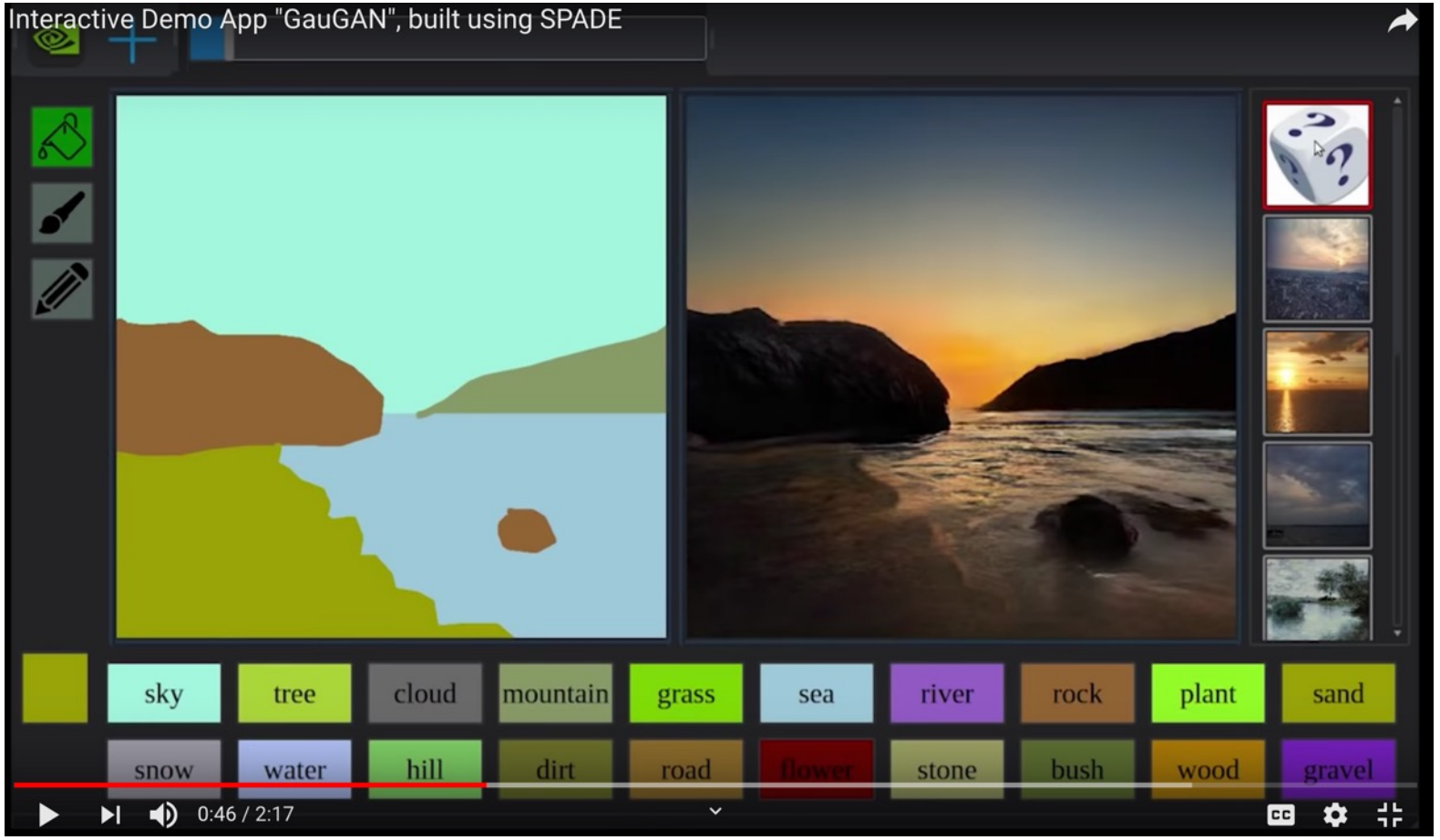
Deep Fakes

 Video Credit: https://www.reddit.com/r/DunderMifflin/comments/fsr308/170_hours_later_here_is_a_clip_of_everyone_in_the/

Deep Fakes

 Video Credit: https://www.reddit.com/r/DunderMifflin/comments/ft2o4d/counseling_scene_but_toby_has_michaels_face/

Paint Photorealistic Landscapes



Interactive Demo: https://www.nvidia.com/en-us/research/ai-playground/

"Semantic Image Synthesis with Spatially-Adaptive Normalization", Park et al, 2019 https://arxiv.org/abs/1903.07291

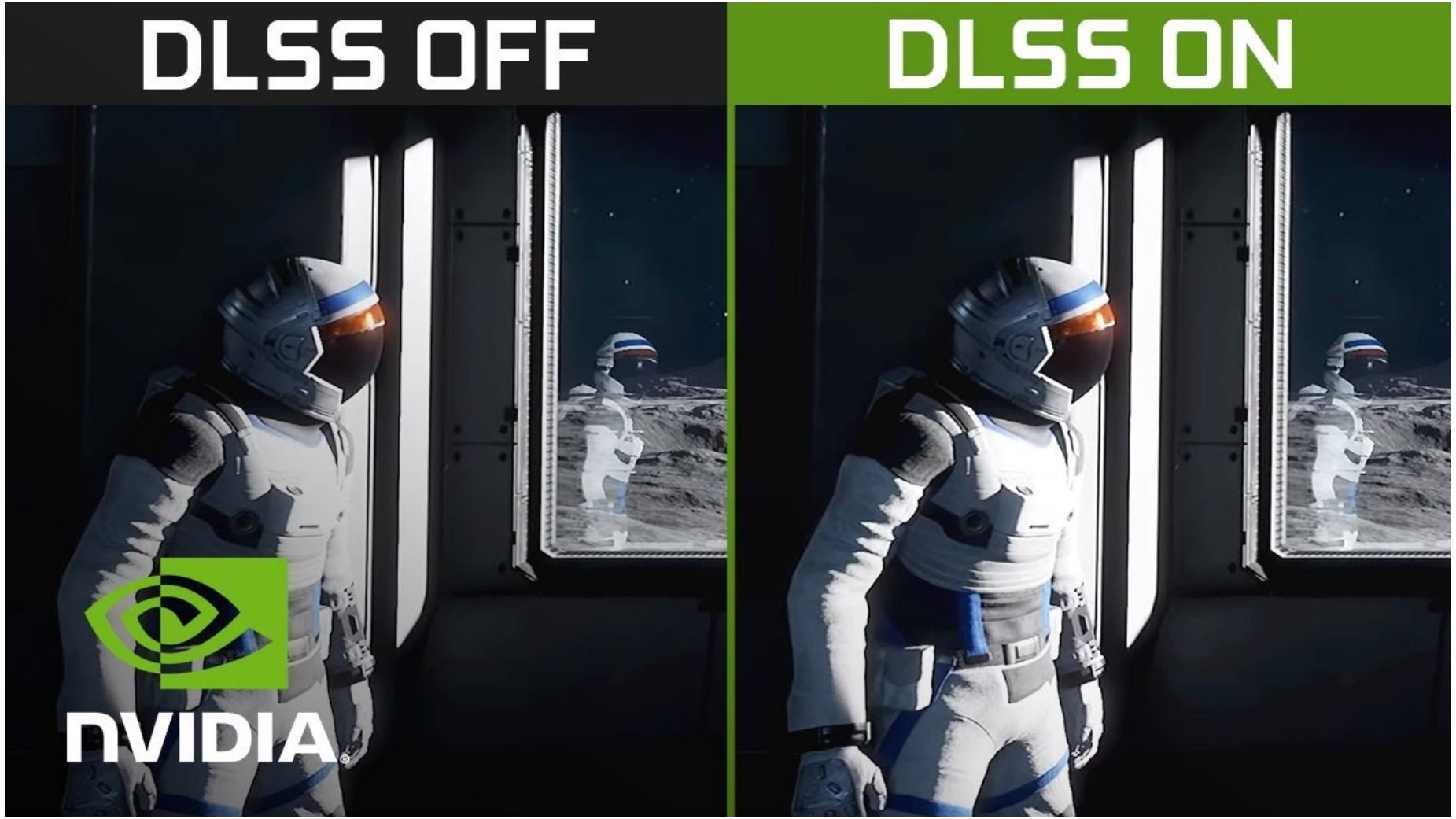
AI Upscaling

- Given a low-resolution image, a deep learning model predicts a high-resolution image that would downscale to look like the original, low-resolution image.

HD -> 4K Simulated Upscaling

https://blogs.nvidia.com/blog/2020/02/03/what-is-ai-upscaling/

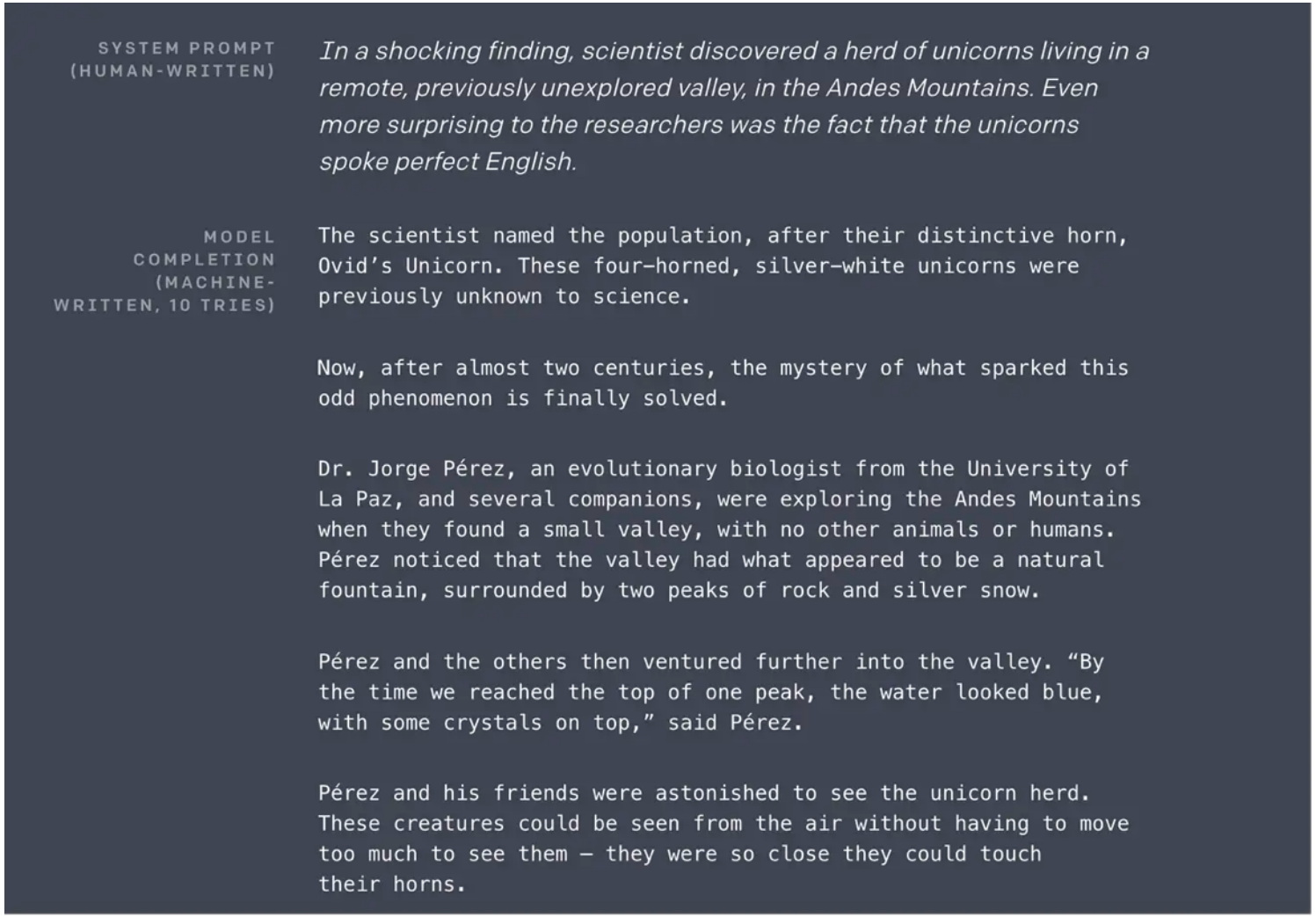
NVIDIA Deep Learning Super Sampling (DLSS)



https://www.nvidia.com/en-us/geforce/news/nvidia-dlss-2-0-a-big-leap-in-ai-rendering/

GPT-2

https://openai.com/blog/better-language-models/



Brad Quinton, Scott Chin

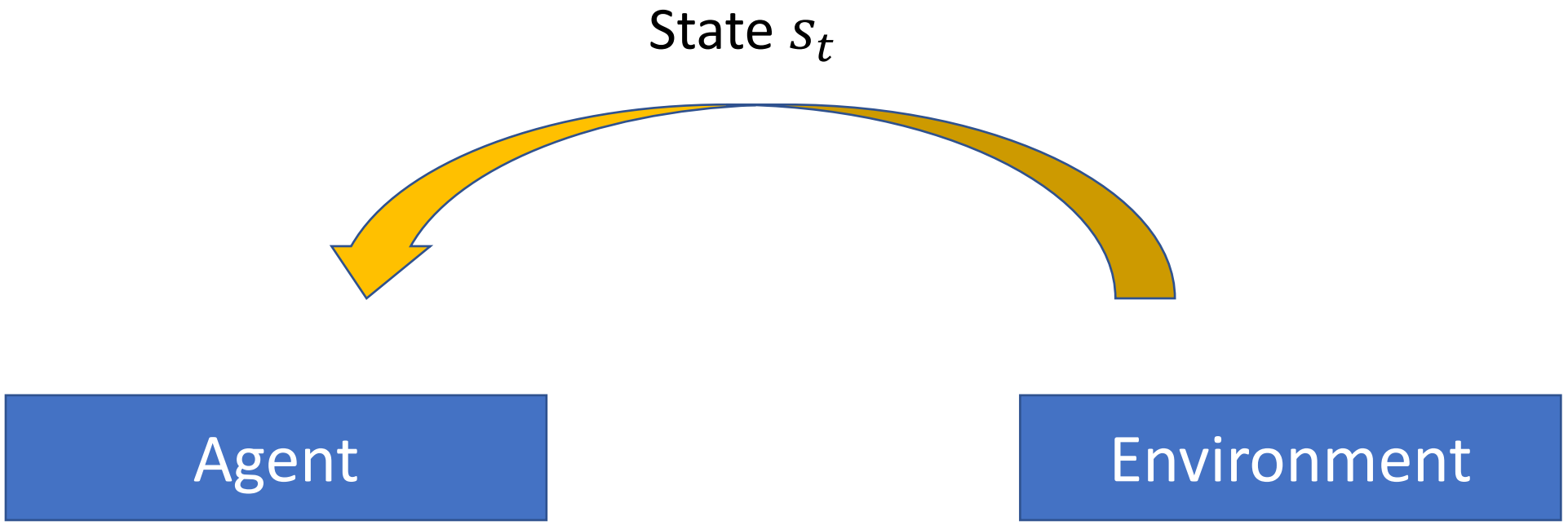
Reinforcement Learning

Reinforcement Learning

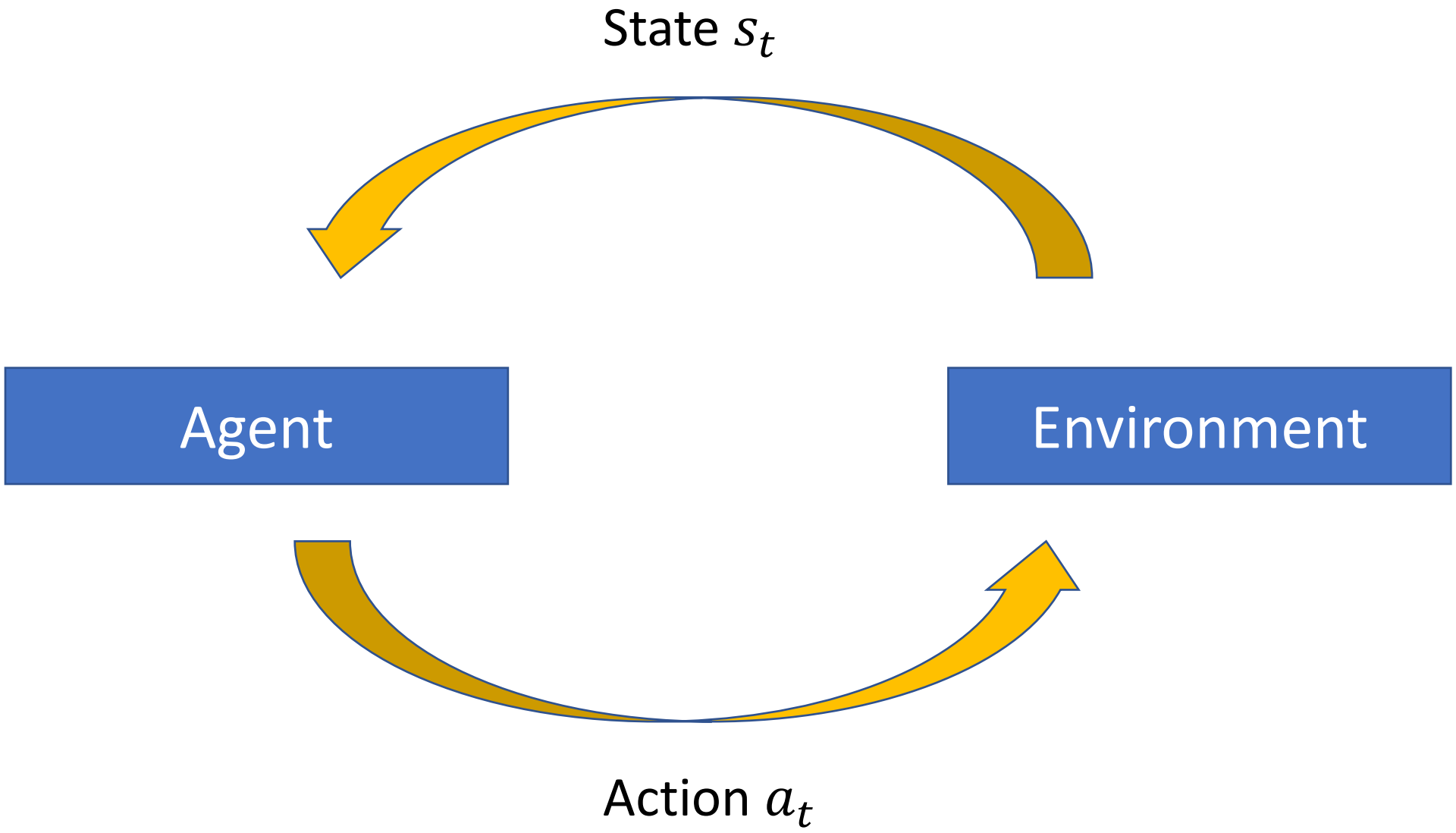
Agent

Environment

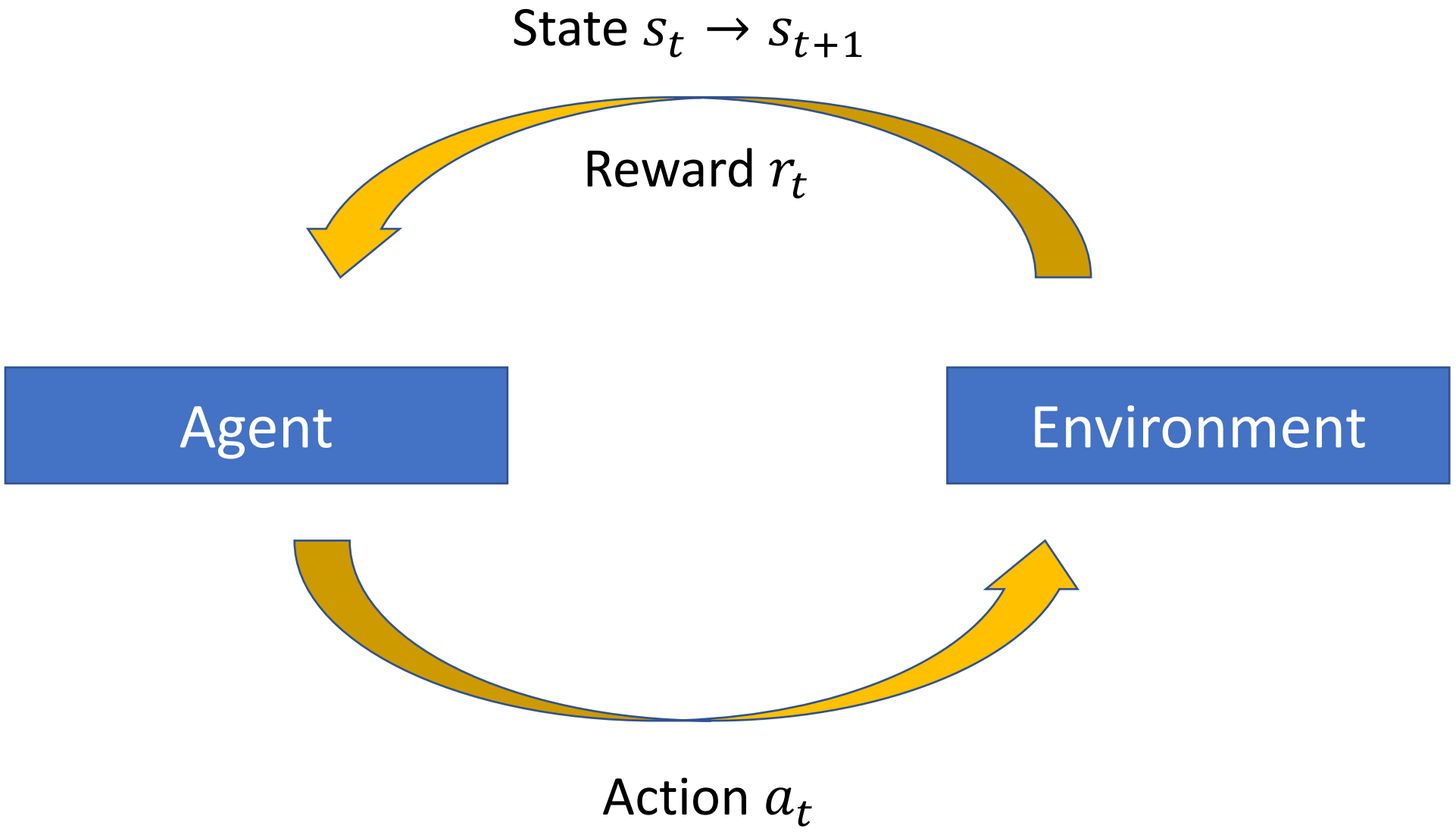
Reinforcement Learning



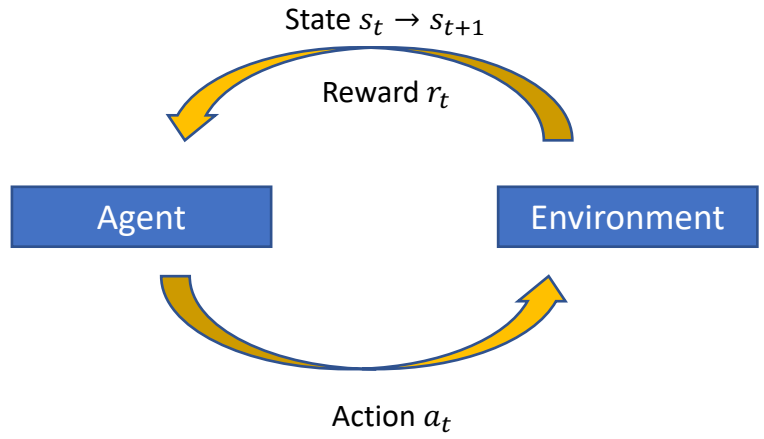
Reinforcement Learning



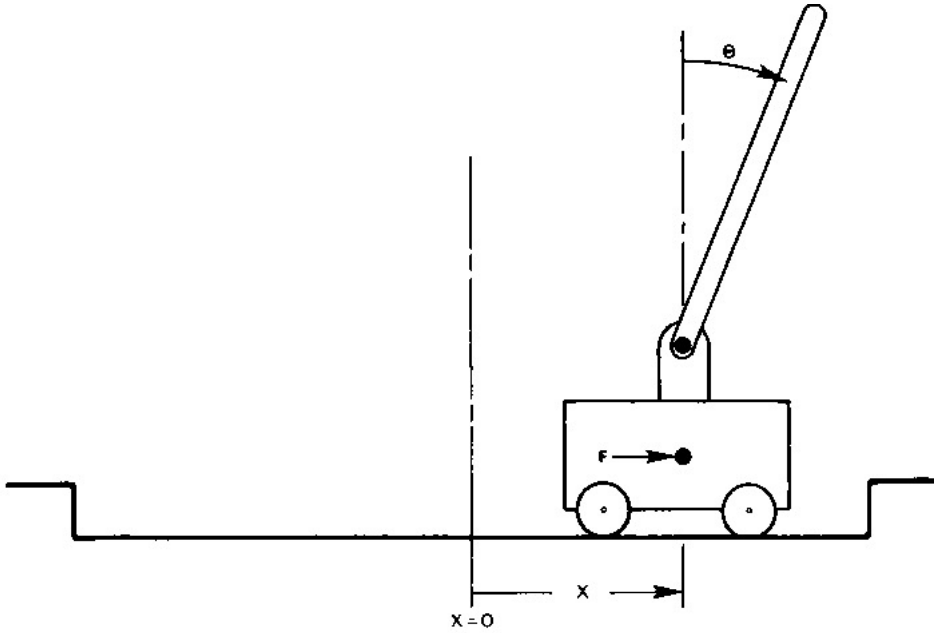
Reinforcement Learning



Example: Cart Pole Problem

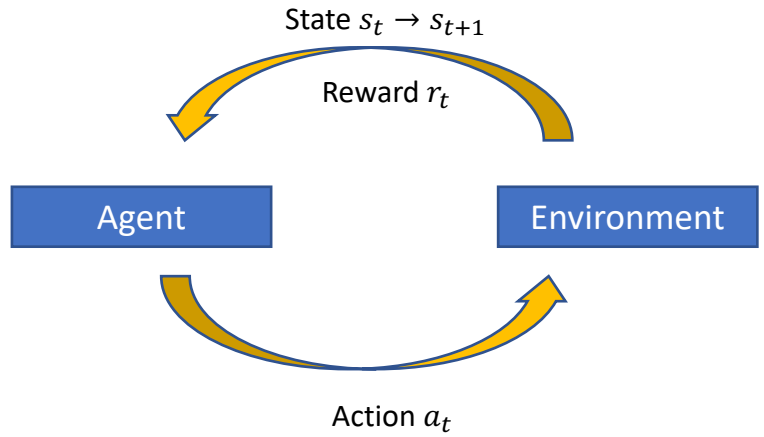


- Objective:
  - Keep the pole upright for as long as possible
- State:
  - angle, angular velocity of pole
  - Position, horizontal velocity of cart
- Action:
  - Horizontal force on cart
- Reward:
  - 1 at each time-step that pole is upright

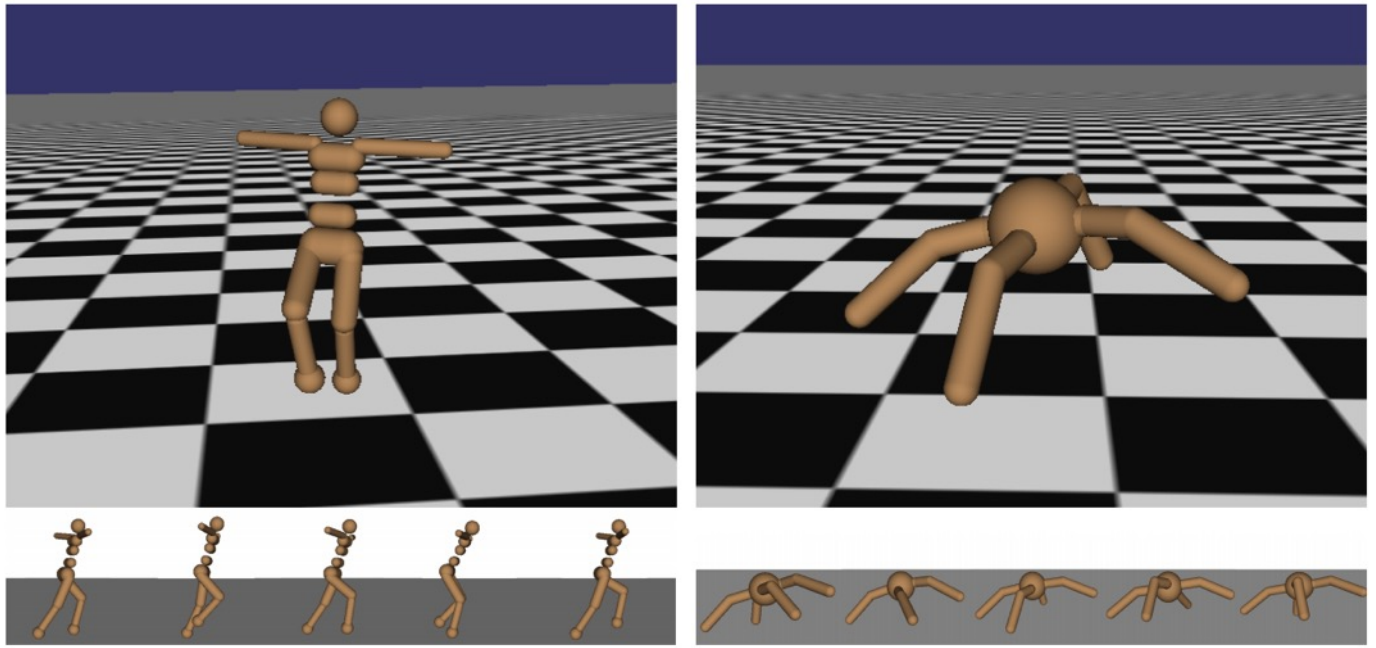


"Neuronlike adaptive elements that can solve difficult learning control problems", Barto et al, 1983, https://ieeexplore.ieee.org/document/6313077

Example: Robot Locomotion



- Objective:
  - Move forward
- State:
  - Angle, position, velocity of joints
- Action:
  - Torque on joints
- Reward:
  - Upright and moving forward

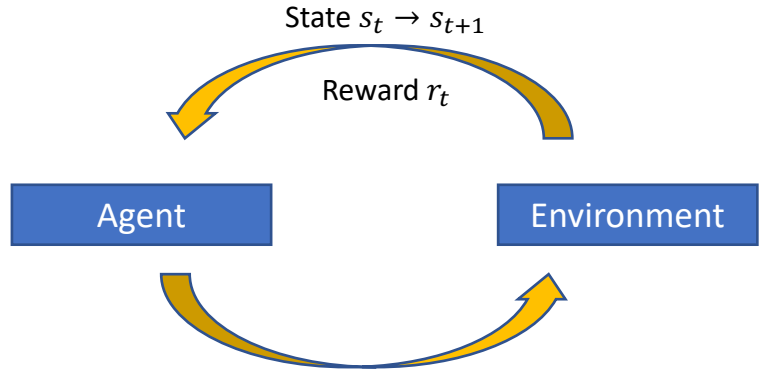


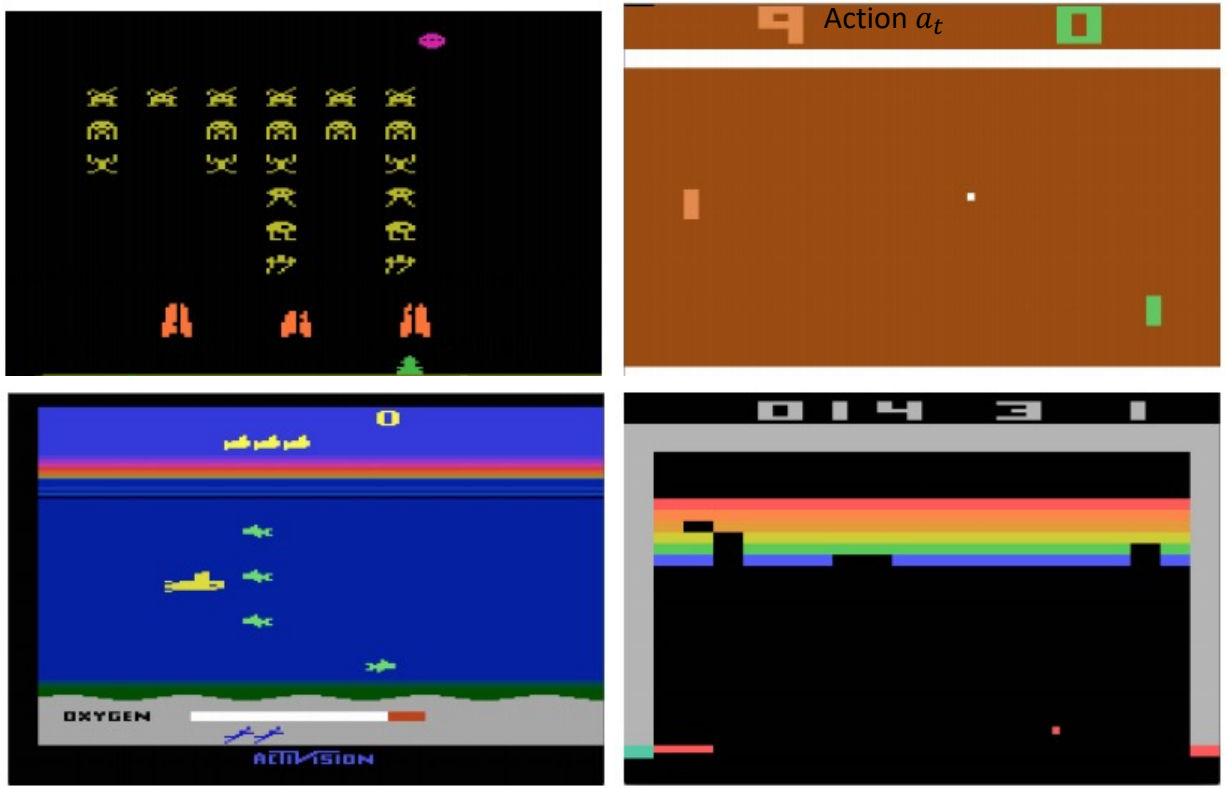
Videos at: https://sites.google.com/site/gaepapersupp/

"High-Dimensional Continuous Control Using Generalized Advantage Estimation", Schulman et al, 2018, https://arxiv.org/pdf/1506.02438.pdf

Atari Video Games

- Objective:
  - Increase score
- State:
  - Depends on game
- Action:
  - Player actions
- Reward:
  - 1 if score increases





"Playing Atari with Deep Reinforcement Learning", Mnih et al, 2013, https://arxiv.org/abs/1312.5602

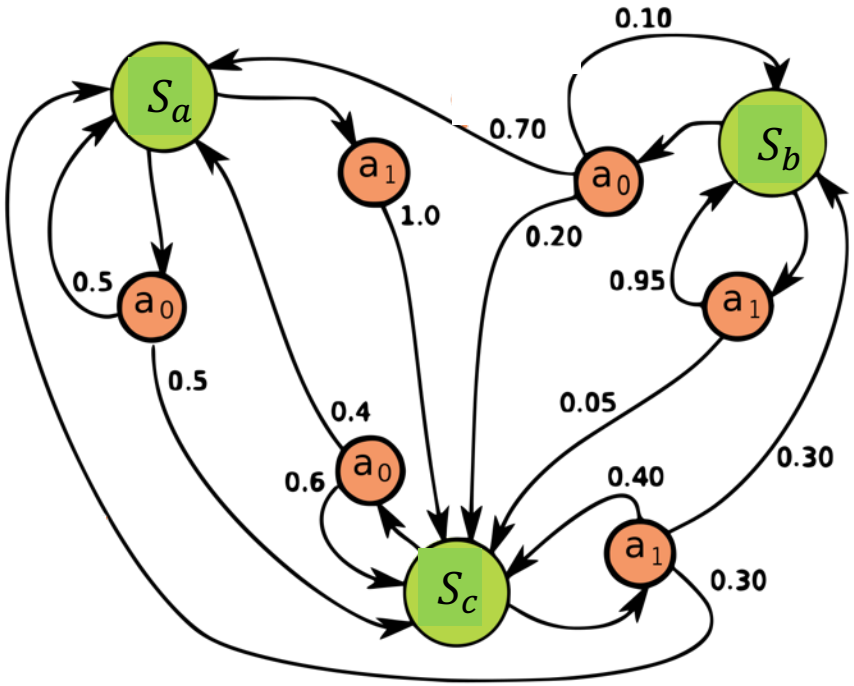
Markov Decision Process (MDP)

- At each time step t, process (e.g. environment) is in some state s
- Decision maker (e.g. Agent) chooses an action a that is available in state s
- The process responds by **randomly** transitioning to a new state s' based on some state transition probability $P_a(s, s')$ due to action a
- A reward is also given to the decision maker $R_a(s, s')$
- Markov Property: Depend only on current state. i.e. no history
- Extension of Markov Chains by having actions and rewards

Markov Decision Process (MDP)

Mathematical Summary:

- S a finite set of states
- A a finite set of actions
  - Not all actions may be available at each state
- $P_a(s, s') = P_a(s_{t+1} = s' | s_t = s, a_t = a)$ probability that at time t action a in state s will cause a state transition to s'
- $R_a(s, s')$ is the immediate reward after transition from state s to s' due to action a

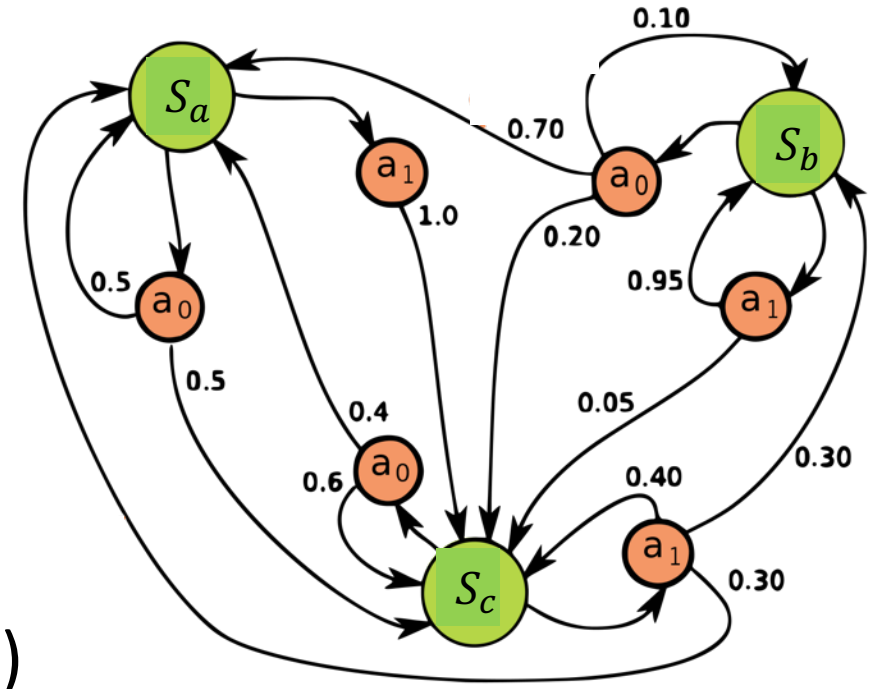


Markov Decision Process (MDP)

- One **episode** of the Markov Decision Process (e.g. one playthrough of a game) forms a finite sequence of states, actions, and rewards

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- MDP ends at a terminal state (e.g. game over)



Core Problem of MDPs

- MDP summarized by: $S, A, P_a(s, s'), R_a(s, s')$

Core Problem of MDPs

- MDP summarized by: $S, A, P_a(s, s'), R_a(s, s')$
- Find a decision making policy (i.e. a function) π that specifies an action a that the agent will choose when in a given state s

$$a = \pi(s)$$

Core Problem of MDPs

- MDP summarized by: $S, A, P_a(s, s'), R_a(s, s')$
- Find a decision making policy (i.e. a function) π that specifies an action a that the agent will choose when in a given state s

$$a = \pi(s)$$

- Find a policy that maximizes the cumulative rewards over an episode

$$\pi^* = \operatorname{argmax} E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- One **episode** of the Markov Decision Process (e.g. one playthrough of a game) forms a finite sequence of states and actions and rewards

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- One **episode** of the Markov Decision Process (e.g. one playthrough of a game) forms a finite sequence of states and actions and rewards

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- Let's look at the sequence of rewards.
  Recall we want to maximize the total reward over the episode

$$R = r_0 + r_1 + r_2 + \cdots + r_n = \sum_{t>0} R_{a_t}(s_t, s_{t+1})$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- One **episode** of the Markov Decision Process (e.g. one playthrough of a game) forms a finite sequence of states and actions and rewards

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- Let's look at the sequence of rewards.
  Recall we want to maximize the total reward over the episode

$$R = r_0 + r_1 + r_2 + \cdots + r_n = \sum_{t>0} R_{a_t}(s_t, s_{t+1})$$

- The problem here is that the environment is stochastic

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Say on the next episode we take the same sequence of actions
- State transition is probabilistic (stochastic) so we are not guaranteed the same sequence of states and rewards
- The further into the future we go, the higher the chance that the state trajectory diverges
- For the same sequence of actions, the further into the future we go, the less likely we will receive the same rewards
- Give less consideration (discount) to rewards received later in the episode

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Considering one episode:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Considering one episode:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- Consider the total reward from some point in time t and onwards

$$R_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_n$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Considering one episode:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- Consider the total reward from some point in time t and onwards

$$R_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_n$$

$$R_0 = r_0 + r_1 + r_2 + \cdots + r_n$$
$$R_1 = r_1 + r_2 + r_3 + \cdots + r_n$$
$$R_2 = r_2 + r_3 + r_4 + \cdots + r_n$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Considering one episode:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

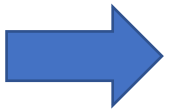
- Consider the total reward from some point in time t and onwards

$$R_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_n$$

$$R_t = r_t + R_{t+1}$$

$$R_0 = r_0 + r_1 + r_2 + \cdots + r_n$$
$$R_1 = r_1 + r_2 + r_3 + \cdots + r_n$$
$$R_2 = r_2 + r_3 + r_4 + \cdots + r_n$$

Can rewrite as



Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Considering one episode:

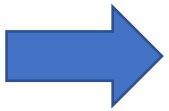
$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- Consider the total reward from some point in time t and onwards

$$R_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_n \qquad R_t = r_t + R_{t+1}$$

$$R_0 = r_0 + r_1 + r_2 + \cdots + r_n$$
$$R_1 = r_1 + r_2 + r_3 + \cdots + r_n$$
$$R_2 = r_2 + r_3 + r_4 + \cdots + r_n$$

Can rewrite as



$$R_0 = r_0 + R_1$$
$$R_1 = r_1 + R_2$$
$$R_2 = r_2 + R_3$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Considering one episode:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- Consider the total reward from some point in time t and onwards

$$R_t = r_t + R_{t+1}$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Considering one episode:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- Consider the total reward from some point in time t and onwards

$$R_t = r_t + R_{t+1}$$

- Introduce a discount factor γ between 0 and 1

$$R_t = r_t + \gamma R_{t+1}$$

- Interpretation: the total reward at time t is the immediate reward, and some reduced amount of the total future rewards

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Considering one episode:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- Expanding the discounted total rewards

$$R_t = r_t + \gamma R_{t+1}$$

$$R_t = r_t + \gamma(r_{t+1} + \gamma R_{t+2})$$

$$R_t = r_t + \gamma(r_{t+1} + \gamma(r_{t+2} + \gamma R_{t+3}))$$

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n r_n$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Considering one episode:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- Expanding the discounted total rewards

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n r_n$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Considering one episode:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- Expanding the discounted total rewards

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n r_n$$

- Discounted Total Reward for an episode is therefore:

$$R = r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots + \gamma^n r_n = \sum_{t=0}^{n} \gamma^t R_t$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Considering one episode:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$$

- Expanding the discounted total rewards

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n r_n$$

- Discounted Total Reward for an episode is therefore:

$$R = r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots + \gamma^n r_n = \sum_{t=0}^{n} \gamma^t R_t$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Discounted Total Reward for an episode is therefore:

$$R = r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots + \gamma^n r_n = \sum_{t=0}^{n} \gamma^t R_t$$

Discounted Future Reward

$$E\left[\sum_{t>0} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

- Discounted Total Reward for an episode is therefore:

$$R = r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots + \gamma^n r_n = \sum_{t=0}^{n} \gamma^t R_t$$

- $\gamma = 0$ would a short-sighted strategy. Only cares about immediate reward
- $\gamma = 0.9$, for example, might be good. Mix of immediate reward and long-term strategy
- If your environment is actually deterministic, can set $\gamma = 1$

Q-Function

- Consider an action a taken at a state s at some time t.
- Let's say we have a function Q that can somehow predict the total discounted reward from the moment we take action a_t in state s_t until the end of the episode

$$Q^{(\pi)}(s_t, a_t) = R_t$$

- Q-Function tells us the (expected) quality of an action at a given state.
- Don't worry yet about how to get this function as it seems like it must be able to see into the future somehow

Q-Function

- Now back to our original goal of finding a policy π that tells us how to choose an action a in a given state s.

$$a = \pi(s)$$

- Specifically, want the **optimal** policy π^* (policy that maximizes reward)
- In other words, at a given state s, we want to choose the best action out of all available actions.
- Let's rewrite the Q function to account for the optimal policy

For the optimal policy

$$Q^{(\pi)}(s_t, a_t) = R_t \quad \Rightarrow \quad Q^{(\pi^*)}(s_t, a_t) = \max R_t$$

The Optimal Policy

- So far, we have this magical Q-function for our optimal policy π^*

$$Q^{(\pi^*)}(s_t, a_t) = \max R_t$$

- To evaluate this, if we had this magical Q function, then at state s_t, we simply evaluate all available actions with the Q function to get R_t for each action, and pick the max to get $\max R_t$. Right?
- Can also just keep track of which action gave us the max (i.e. argmax).
- This, by definition, gives us the optimal policy!

$$\pi^*(s) = \underset{a \in A_s}{\operatorname{argmax}} Q^{(\pi^*)}(s, a)$$

The Bellman Equation

- Ok we still don't know how to get Q!

The Bellman Equation

- Ok we still don't know how to get Q!
- Consider one state transition from state s to s' due to an action a which yields a reward r
- The Bellman Equation states that $Q^{(\pi^*)}$ satisfies the following:

$$Q^{(\pi^*)}(s, a) = r + \gamma Q^{(\pi^*)}(s', a')$$

The Bellman Equation

- Ok we still don't know how to get Q!
- Consider one state transition from state s to s' due to an action a which yields a reward r
- The Bellman Equation states that $Q^{(\pi^*)}$ satisfies the following:

$$Q^{(\pi^*)}(s,a) = r + \gamma Q^{(\pi^*)}(s',a')$$

- Max possible reward of action a in state s is the immediate reward r plus the maximum possible future reward from the next state s' (i.e. a' is the best action in state s' since we are following the optimal policy)

$$\max R_t = r_t + \gamma \max R_{t+1}$$

Q-Learning

$$Q^{(\pi^*)}(s,a) = r + \gamma Q^{(\pi^*)}(s',a')$$

- Use the Bellman equation to iteratively approximate the Q-Function!

"Convergence of Q-Learning: A Simple Proof", Melo, http://www.academia.edu/download/50351200/ProofQlearning.pdf

Q-Learning

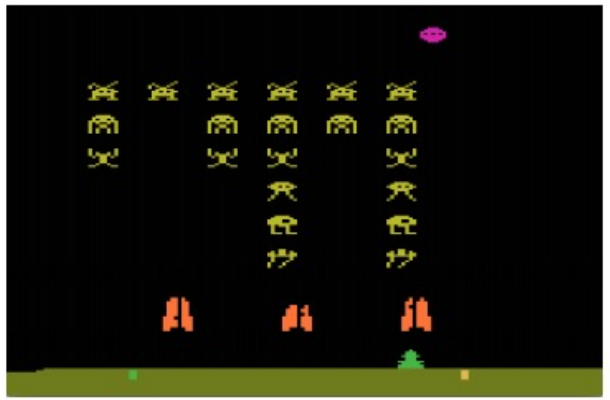
$$Q^{(\pi^*)}(s,a) = r + \gamma Q^{(\pi^*)}(s',a')$$

- Use the Bellman equation to iteratively approximate the Q-Function!

```
Q = random array of shape (n_states, n_actions)
s = random initial state
repeat:
  a = random action
  r = reward for action a from state s
  s_next = sample next state based on action a
  Q[s,a] += α*(r + γ*max(Q[s_next,:]) – Q[s,a])
  s = s_next
```


"Convergence of Q-Learning: A Simple Proof", Melo, http://www.academia.edu/download/50351200/ProofQlearning.pdf

State Representation in Atari Games



- Each Atari game has it's own unique state representations
  - E.g. Breakout: paddle location, ball velocity and location, brick locations
  - E.g. Space Invaders: player location, alien locations, alien velocities, etc.
- A more universal state representation is the frame image itself
- From each frame, you can infer all positional state info
- From two consecutive frames, you can infer state info that change over time (e.g. velocities)

So where does Deep Learning come in?

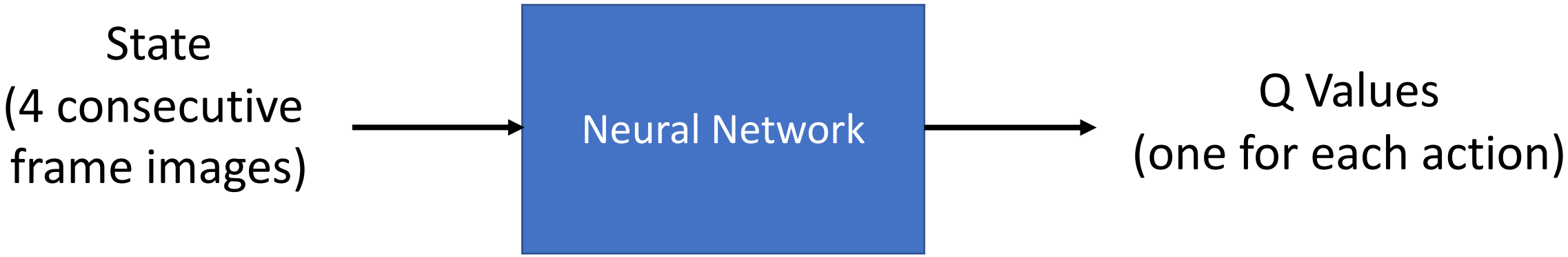
- In the DeepMind Atari paper, they used four consecutive frames to represent the current state.
- 4 84x84 pixel grayscale images (each pixel is a value from 0-255)
- Possible states: $256^{4*84*84} \approx 10^{67970}$
- Our Q-Learning iterative approach requires us to keep track of a value for each possible state and action pair. Becomes infeasible.

So where does Deep Learning come in?

- In the DeepMind Atari paper, they used four consecutive frames to represent the current state.
- 4 84x84 pixel grayscale images (each pixel is a value from 0-255)
- Possible states: $256^{4*84*84} \approx 10^{67970}$
- Our Q-Learning iterative approach requires us to keep track of a value for each possible state and action pair. Becomes infeasible.

Use a Neural Network to approximate the Q Function

Approximate Q Function with Neural Network



Deep Q-Network (DQN)

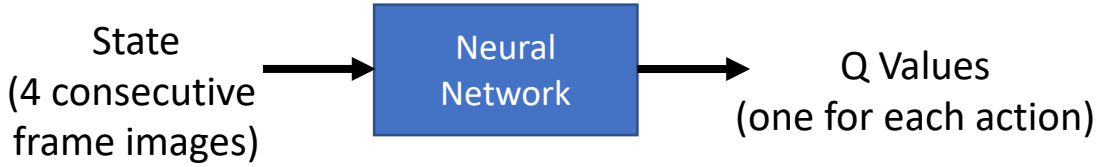
Deep Q-Networks (DQN)

- In the original paper, they trained a separate Neural Network for each of the 7 games they investigated
- The games had 4-18 possible actions (e.g. up, down, left, right, …)
- Output is one Q value (real number) per action

Layer	Hyperparams	Activation	Output Volume
Input			(84,84,4)
Conv	K=16, f=(8,8), s=4	ReLU	(20,20,16)
Conv	K=32, f=(4,4), s=2	ReLU	(9,9,32)
Flatten		None	(2592,)
Fully Connected	256 units	ReLU	(256,)
Fully Connected	4-18 units	None	(4-18,)

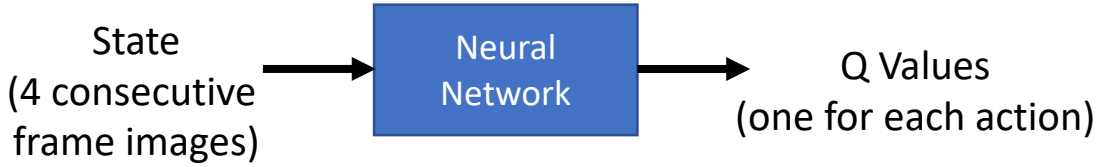
"Playing Atari with Deep Reinforcement Learning", Mnih et al, 2013, https://arxiv.org/abs/1312.5602

How to Train the DQN?



- Input to DQN is state s
- Output are Q values for all actions from state s $Q(s, \ldots)$
- Use supervised learning. What should the target be?

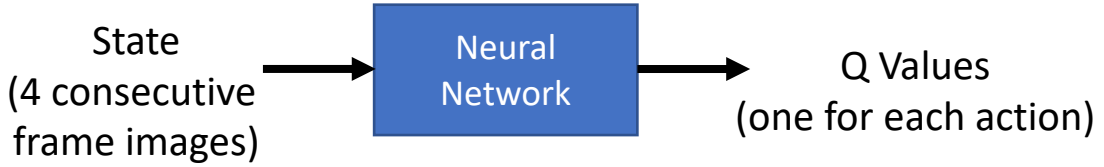
How to Train the DQN?



- Input to DQN is state s
- Output are Q values for all actions from state s $Q(s, \dots)$
- Use supervised learning. What should the target be?
- Use Bellman Equation $Q(s,a) = r + \gamma Q(s',a')$

- Target output is $r + \gamma Q(s',a')$
  - r and γ are known
  - $Q(s',a')$ is just output of the DQN for some other state s' (i.e. the next state)

How to Train the DQN?



- Input to DQN is state s
- Output are Q values for all actions from state s $Q(s, \dots)$
- Use supervised learning. What should the target be?
- Use Bellman Equation $Q(s,a) = r + \gamma Q(s',a')$

- Target output is $r + \gamma Q(s',a')$
  - r and γ are known
  - $Q(s',a')$ is just output of the DQN for some other state s' (i.e. the next state)
- Use squared error: $L = (y - \hat{y})^2$

Deep Q-Learning

```
initialize parameters of DQN
s = random initial state
repeat:
  Qs = DQN.predict(s) # Q values for all actions from s
  a = argmax(Q_s)     # action with the highest Q value from state s
  r = reward for action a from state s
  s_next = sample next state based on action a
  Qs_next = DQN.predict(s_next) # Q values for all actions from s_next
  target = r + γmax(Qs_next)    #
  loss = (target – Qs)**2
  backprop through DQN using loss
  gradient descent update parameters of DQN
  s = s_next
```


"Convergence of Q-Learning: A Simple Proof", Melo, http://www.academia.edu/download/50351200/ProofQlearning.pdf

Optimization is not stable

Target for loss calculation is produced by the neural network being trained! i.e. it will change as training progresses

DeepMind employs a number of ”tricks” to make this all work:

- Experience Replay
- ε-Greedy Exploration
- Error clipping
- Reward clipping
- Etc.

DeepMind DQN Playing Breakout

https://www.youtube.com/watch?v=V1eYniJ0Rnk

Aside: Breakout as Supervised Learning

- Treat as classification problem:
- Label each frame with optimal action
  - E.g. 3 classes: left, right, launch ball

Problem:

- Need lots of labelled training data
- Can only get as good as the human(s) who created the label data
- Not how humans would learn to play this game

Go master Lee says he quits unable to win over AI Go players

"With the debut of AI in Go games, I've realized that I'm not at the top even if I become the number one through frantic efforts. Even if I become the number one, there is an entity that cannot be defeated."

- Lee Se-dol

https://en.yna.co.kr/view/AEN20191127004800315#none

Deepmind AlphaStar

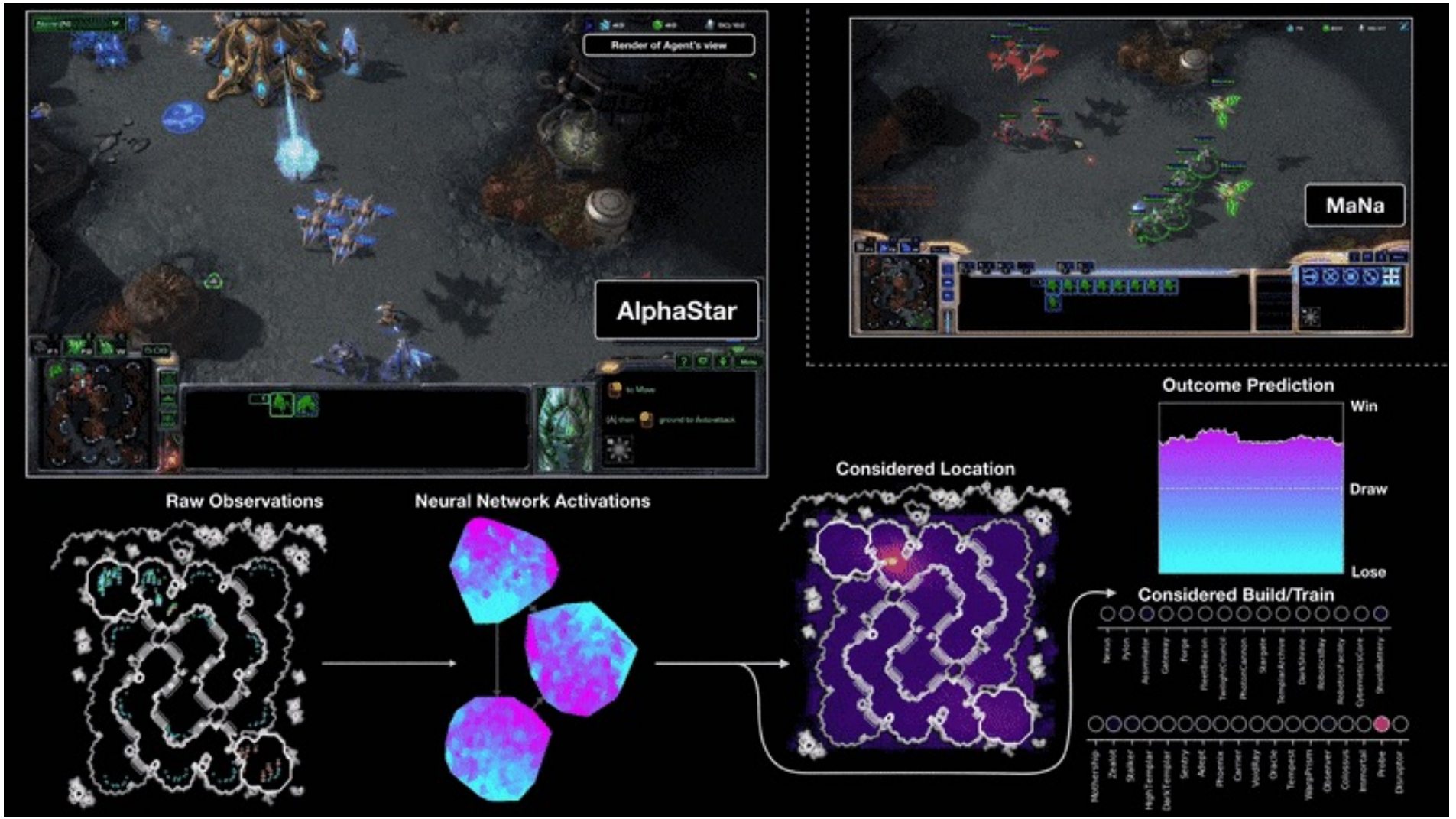


 Image Credit: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

Deepmind AlphaStar

- DeepMind StarCraft II Demonstration (AlphaStar vs TLO) https://www.youtube.com/watch?v=cUTMhmVh1qs

- https://deepmind.com/blog/article/AlphaStar-Grandmaster-level-in-StarCraft-II-using-multi-agent-reinforcement-learning
- https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

Deepmind AlphaFold

BLOG POST
RESEARCH

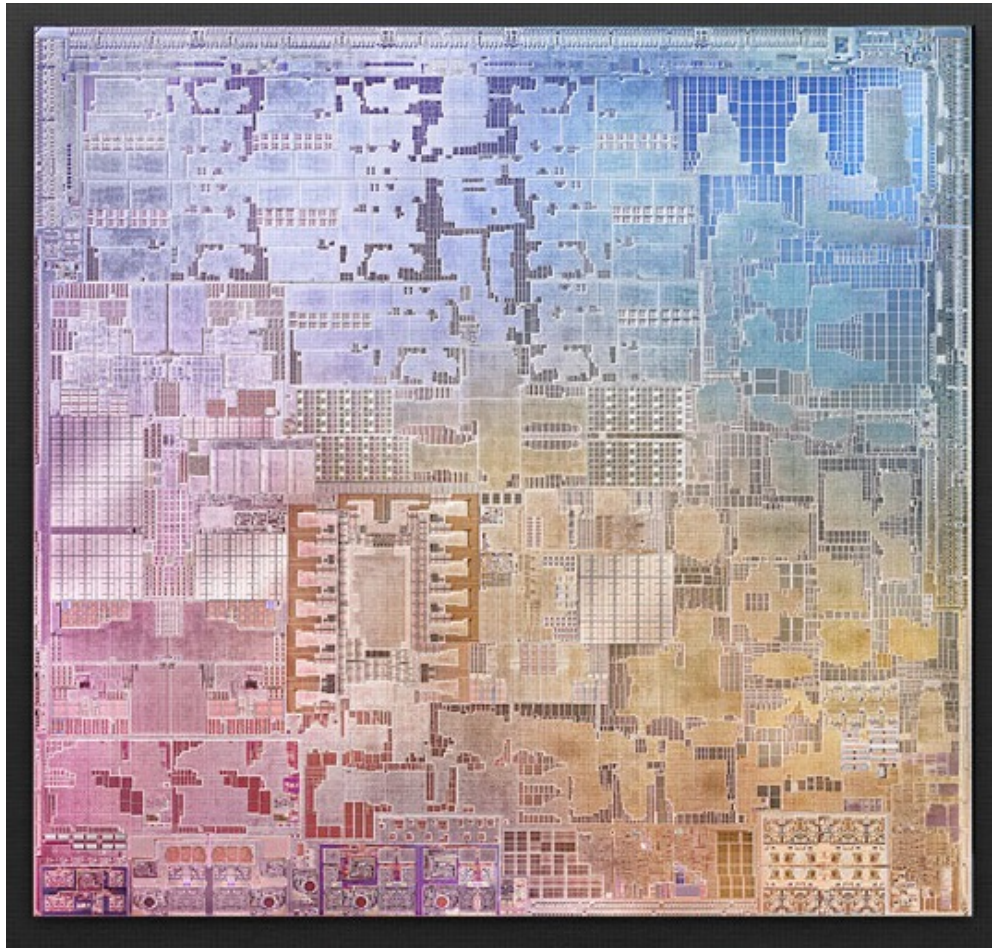
30 NOV 2020

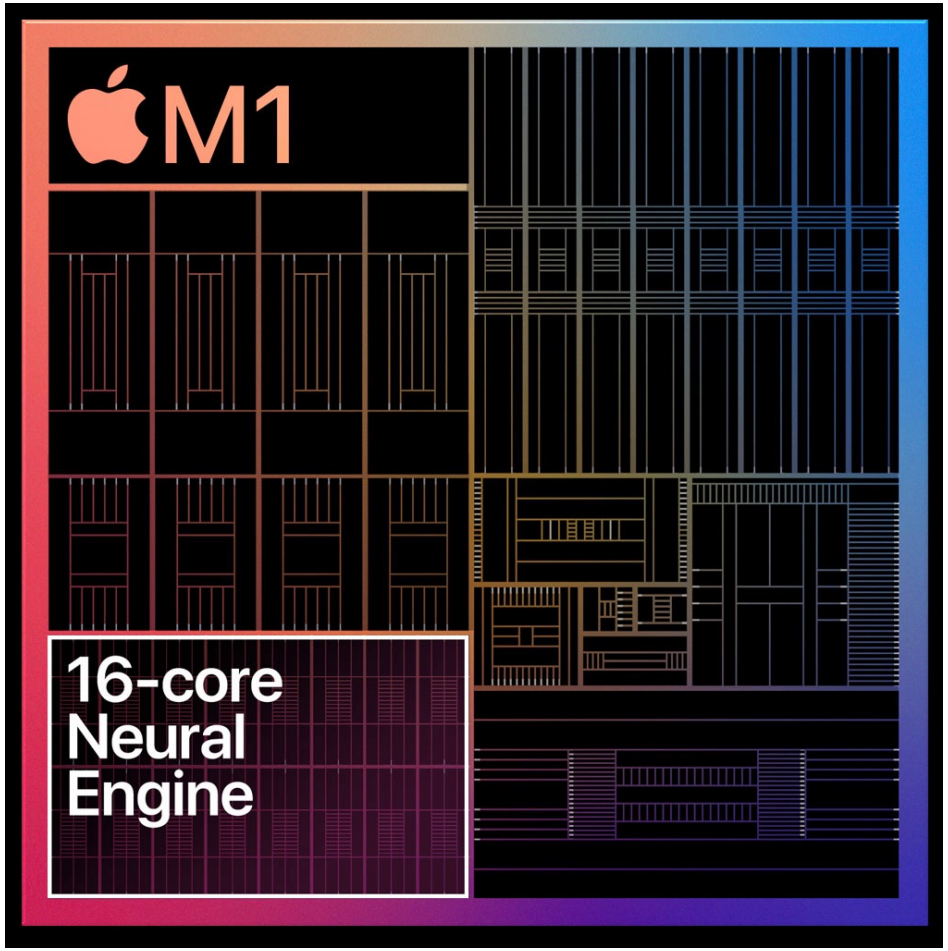
AlphaFold: a solution to a 50-year-old grand challenge in biology

SHARE

https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology

Apple M1 Chip

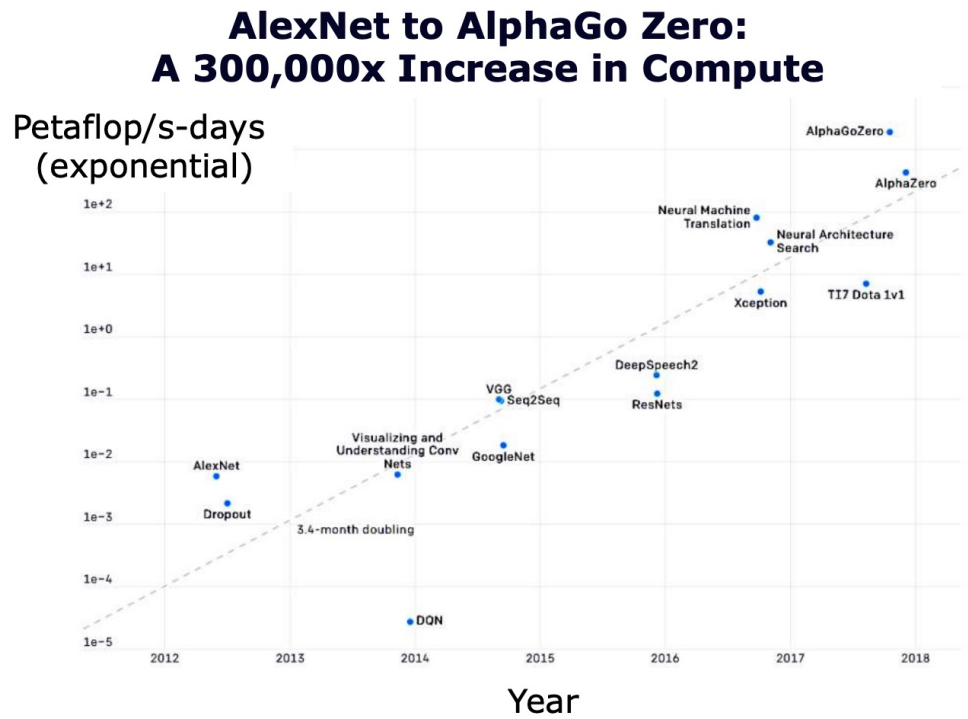




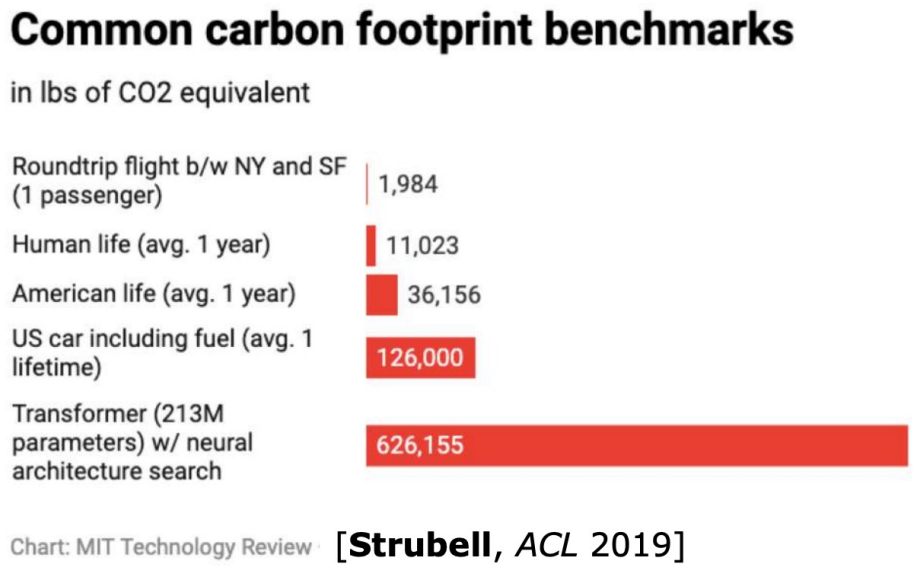
Source: https://www.apple.com/mac/m1/

NEURIPS 2019 Most Popular Tutorial

Efficient Processing of Deep Neural Networks: from Algorithms to Hardware Architectures



Source: Open AI (https://openai.com/blog/ai-and-compute/)



Video: https://slideslive.com/38922815/efficient-processing-of-deep-neural-network-from-algorithms-to-hardware-architectures

Slides: http://eyeriss.mit.edu/2019_neurips_tutorial.pdf

Brad Quinton, Scott Chin

NEURIPS 2019 Keynote

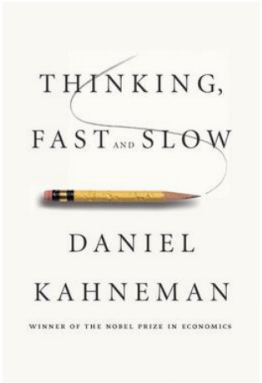
From System 1 Deep Learning to System 2 Deep Learning

SYSTEM 1 VS. SYSTEM 2 COGNITION

2 systems (and categories of cognitive tasks):

System 1

- Intuitive, fast, **UNCONSCIOUS**, non-linguistic, habitual
- Current DL



Manipulates high-level / semantic concepts, which can be recombined combinatorially

System 2

- Slow, logical, sequential, **CONSCIOUS**, linguistic, algorithmic, planning, reasoning
- Future DL

Mila

3

Video: https://slideslive.com/38922304/from-system-1-deep-learning-to-system-2-deep-learning
Slides: http://www.iro.umontreal.ca/~bengioy/NeurIPS-11dec2019.pdf

Brad Quinton, Scott Chin