

# CSCI 566: Deep Learning and Its Applications

Jesse Thomason

Lecture 11: Deep Learning and The World

# Administrivia: Paper Roleplaying Breakout 2-Pagers

- These are due the day of your breakout session
- Many folks have not turned these in yet, even from two weeks ago; the 5% late day penalties apply to the 2-pagers
- If you signed up for a breakout session using something *other than your username*, we may have trouble connecting your 2-pager to your session sign up
  - If you have turned your 2-pager in but know you signed up for a session using a string other than your username, let us know so we can connect the two

# Markov Decision Process with Rewards

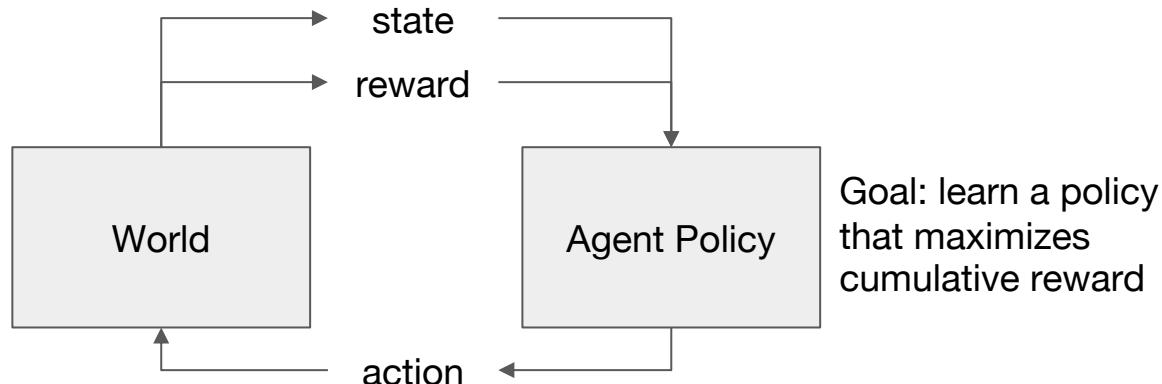
$\mathcal{S}$  : set of possible states

$\mathcal{A}$  : set of possible actions

$\mathcal{R}$  : distribution of reward given (state, action) pair

$\mathbb{P}$  : transition probability i.e. distribution over next state given (state, action) pair

$\gamma$  : discount factor



# Markov Decision Process with Rewards

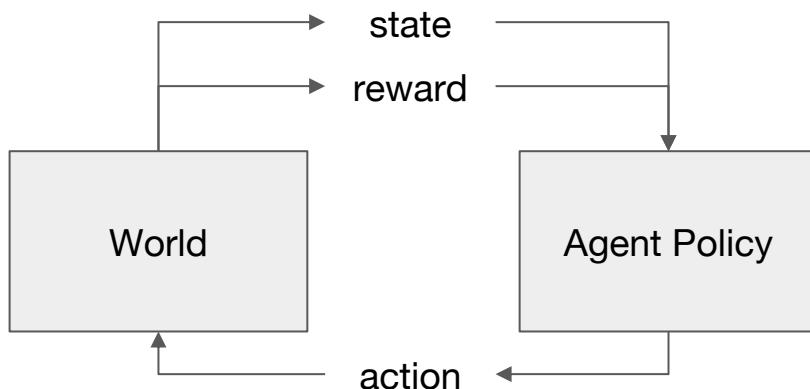
$\mathcal{S}$  : set of possible states

$\mathcal{A}$  : set of possible actions

$\mathcal{R}$  : distribution of reward given (state, action) pair

$\mathbb{P}$  : transition probability i.e. distribution over next state given (state, action) pair

$\gamma$  : discount factor

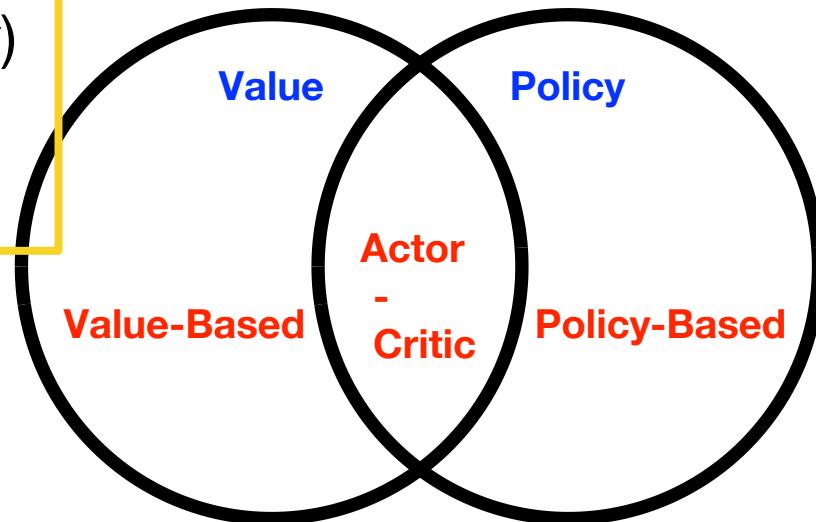


$$s_0 \sim p(s_0), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$$

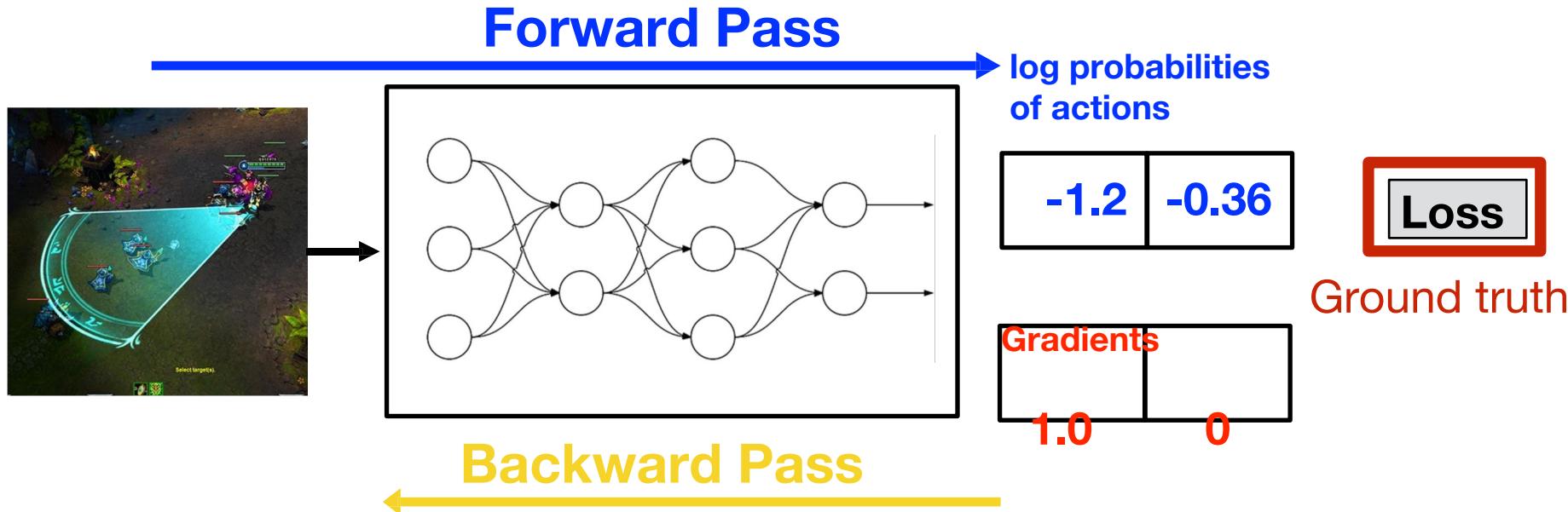
$$\pi^* = \arg \max_{\pi} \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t | \pi \right]$$

# Seeking An Optimal Policy

- Value Based (e.g., Deep Q-Network)
  - Learn Value Function
  - Implicit Policy
- Policy Based (e.g., guided policy)
  - No Value Function
  - Directly Learn Policy
- Actor-Critic
  - Learn Value Function
  - Learn Policy

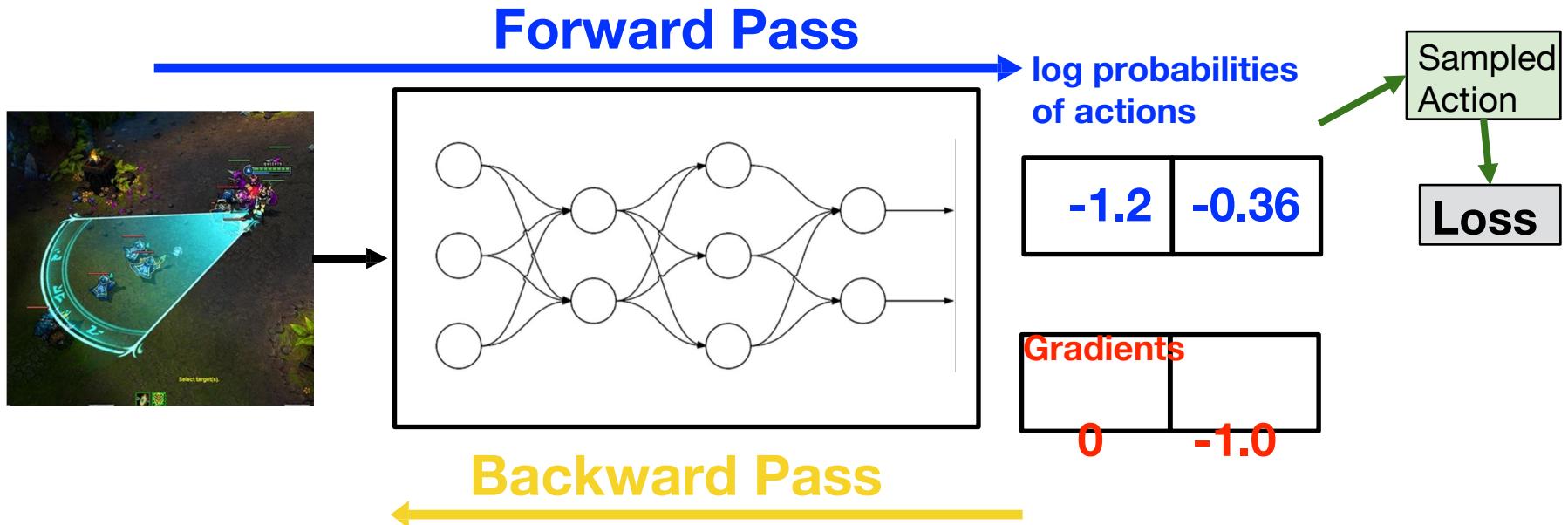


# Supervised Learning versus Policy Gradient



Learn:  $\pi: S \rightarrow A$  for  $\pi$  parameterized by  $\theta$

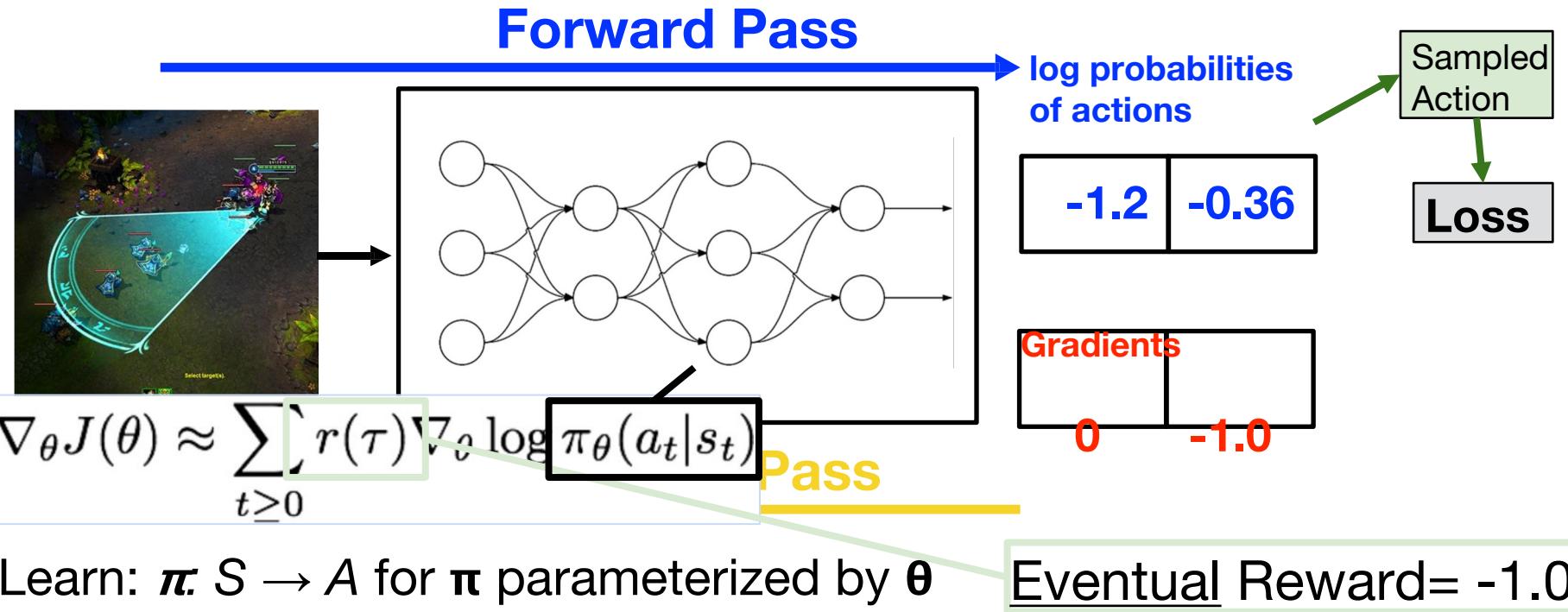
# Supervised Learning versus Policy Gradient



Learn:  $\pi: S \rightarrow A$  for  $\pi$  parameterized by  $\theta$

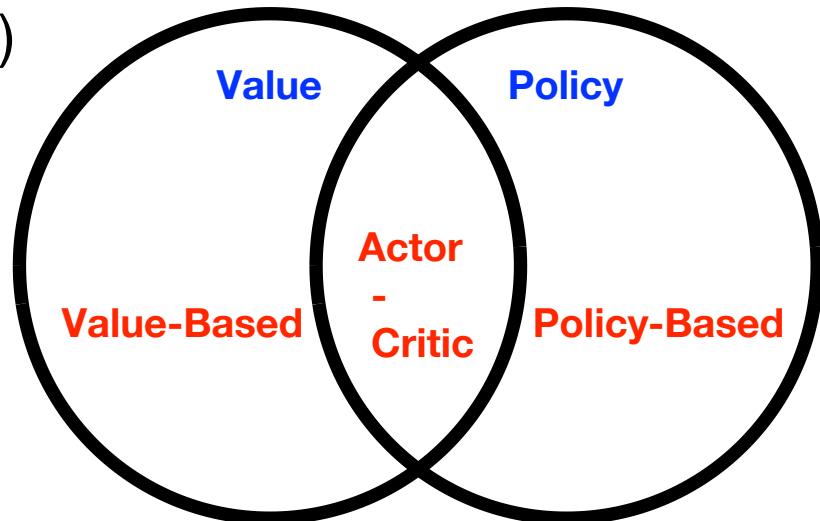
Eventual Reward = -1.0

# Training a Network with Policy Gradient (REINFORCE)



# Seeking An Optimal Policy

- Value Based (e.g., Deep Q-Network)
  - Learn Value Function
  - Implicit Policy
- Policy Based (e.g., guided policy)
  - No Value Function
  - Directly Learn Policy
- Actor-Critic
  - Learn Value Function
  - Learn Policy



# Bellman Equation

- Optimal Q-value function gives maximum expected return

$$Q^*(s, a) = \max_{\pi} \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right]$$

- Such an optimal function satisfies the optimality described by the *Bellman Equation*:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$

- which in a nutshell says: the highest value of a (state, action) pair is the immediate reward plus the discounted best future reward from here

# Value Iteration

- **Intuition:** Update the Q-value function iteratively

$$Q_{i+1}(s, a) = \mathbb{E} \left[ r + \gamma \max_{a'} Q_i(s', a') | s, a \right]$$

- Getting an accurate estimate  $Q^*(s, a)$  for every state-action pair is intractable for continuous state/action spaces and large discrete spaces too (e.g., all possible space invaders screens)
- What if we could *approximate* the Q function?

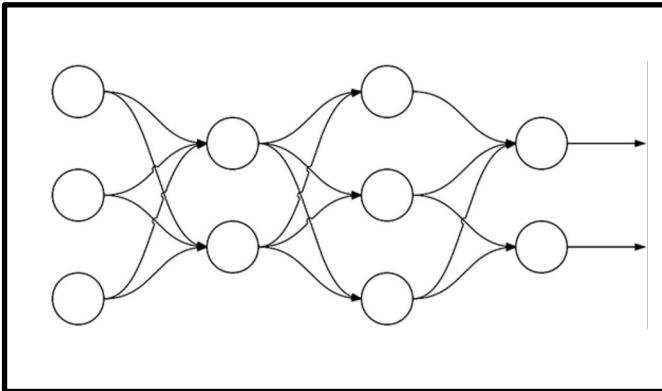
$$Q(s, a; \theta) \approx Q^*(s, a)$$

- What's a good way to do function approximation?

# Training a Network with Value Iteration

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

Forward Pass



Action Q-value  
estimates

.94 .87

Backward Pass

# Training a Network with Value Iteration

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

## Forward Pass

The Loss:  $L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot)} [(y_i - Q(s, a; \theta_i))^2]$

“Ground truth” Q\*

Approximation

$$\mathbb{E}_{s' \sim \mathcal{E}} [r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a]$$

Estimate via value iteration intuition

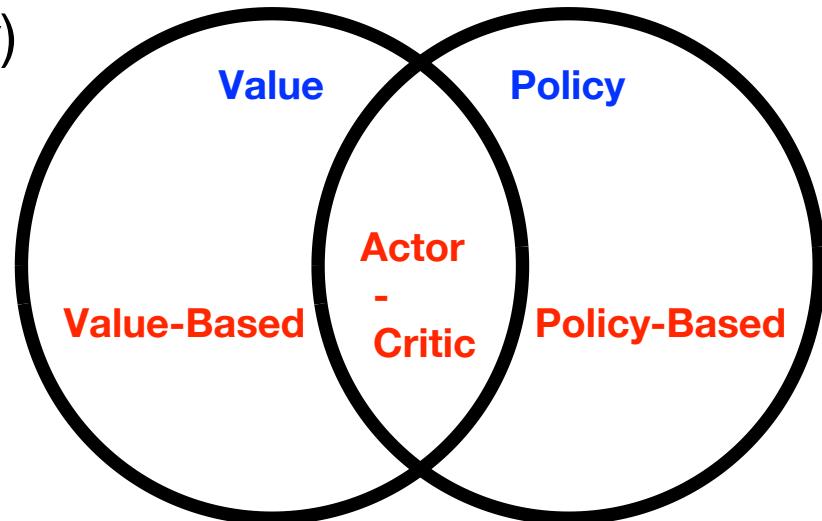
## Backward Pass

Update  $\theta$ :  $\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot); s' \sim \mathcal{E}} [(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)) \nabla_{\theta_i} Q(s, a; \theta_i)]$

Magnitude of change	Direction
---------------------	-----------

# Seeking An Optimal Policy

- Value Based (e.g., Deep Q-Network)
  - Learn Value Function
  - Implicit Policy
- Policy Based (e.g., guided policy)
  - No Value Function
  - Directly Learn Policy
- Actor-Critic
  - Learn Value Function
  - Learn Policy



# Seeking An Optimal Policy

- Policy gradient intuition: sample trajectories with suboptimal policy, then for high reward trajectories push up action probabilities taken, for low reward trajectories push down action probabilities taken
  - Problem? Rewards across sometimes literally random trajectories are going to have high variance
- Value iteration intuition: for every state-action pair, estimate the best future reward we could get if we chose that action from this state
  - Problem? Needs a lot of state-action data

# Advantage Actor-Critic Algorithm

- Take advantage of the strengths of policy gradients and value iteration while removing some of their weaknesses
- In particular:
  - Goal: Reduce variance of policy gradient updates
  - Method: change the parameter updates to upweight an action proportional to how much better it *turned out* to have been when taken versus our expectation

$$\nabla_{\theta} J(\theta) \approx \sum_{t \geq 0} \left( \sum_{t' \geq t} \gamma^{t'-t} r_{t'} - b(s_t) \right) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

Discounted reward achieved from this state      Estimate of reward from this state

The diagram illustrates the components of the Advantage Actor-Critic update rule. It shows the formula  $\nabla_{\theta} J(\theta) \approx \sum_{t \geq 0} \left( \sum_{t' \geq t} \gamma^{t'-t} r_{t'} - b(s_t) \right) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$ . A red box encloses the term  $\sum_{t' \geq t} \gamma^{t'-t} r_{t'}$ , which represents the discounted reward achieved from this state. A purple box encloses the term  $b(s_t)$ , which represents the estimate of reward from this state.

# Advantage Actor-Critic Algorithm

- How to estimate the reward we can achieve from a given state?
- Use Q-function and value function!
  - Value function: integrate over quality estimates from all possible actions based on policy probability to take them

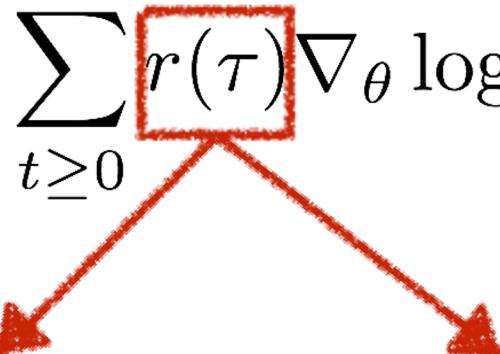
$$V^\pi(s) = \sum_{a \in A} \pi(a|s) * Q^\pi(s, a)$$

- Then we call the “advantage” an action  $a$  has at state  $s$ :

$$Q^\pi(s_t, a_t) - V^\pi(s_t)$$

# Advantage Actor-Critic Algorithm

- Goal: Reduce variance of policy gradient updates

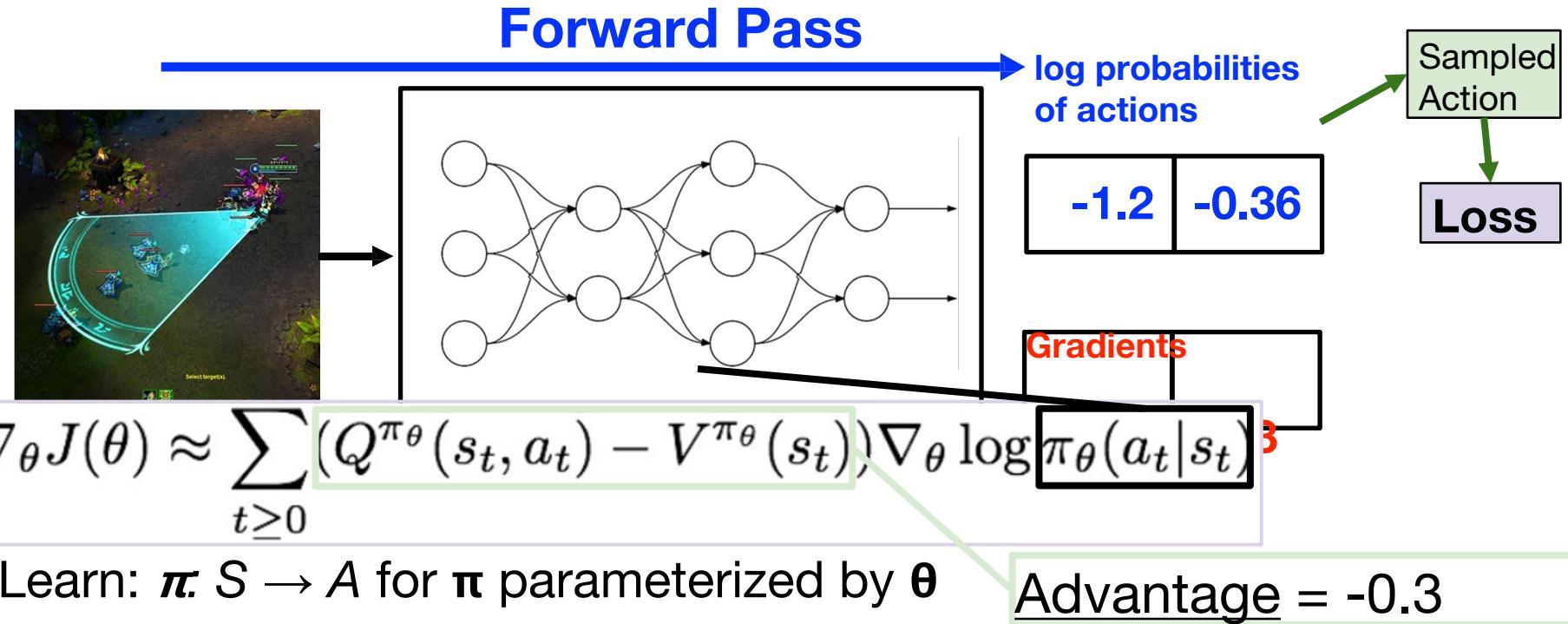
$$\nabla_{\theta} J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

$$Q(s_t, a_t)$$
$$Q(s_t, a_t) - V(s_t)$$

- Method: change the parameter updates to upweight an action proportional to how much better it turned out to have been to take it versus our expectation

# Advantage Actor-Critic Algorithm

- Can we learn Q and Value functions?
  - Q-Learning + Policy Gradient methods!
- Train both an **Actor** (policy) and a **Critic** (Q-function)
- Actor: Which action to take?
- Critic: How good the action was and how to adjust parameters?
- Advantage Function:  $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ 
  - How much better is  $a$  than expected?
  - Reduces the variance of policy gradient updates

# Training a Network with Advantage Actor-Critic (A2C)



# High-profile Recent Example of Deep RL?

- ChatGPT!
  - Reinforcement Learning from Human Feedback (RLHF)
    - Prithviraj Ammanabrolu was doing it before it was cool
- Upshot:
  - Remember the core function of a language model?
  - Autoregressive language model?
  - $p(w_i | w_{i-1} w_{i-2} \dots w_1)$
  - ...  $\pi(a|s)$  for next word *actions* and context *states*
- ChatGPT training uses a policy gradient fine-tuning approach

# High-profile Recent Example of Deep RL

LE

Can you explain to me how RL with human feedback works for language models like I am five?



Sure! So, imagine you have a robot named Rufus who wants to learn how to talk like a human. Rufus has a language model that helps him understand words and sentences.

First, Rufus will say something using his language model. For example, he might say "I am a robot."

Then, a human will listen to what Rufus said and give him feedback. The human might say something like a natural sentence a human would say. The human might say "That's not how a robot talks." Rufus. Humans don't usually say 'I am a robot.' They might say 'I'm a robot' or 'I'm a machine.'"

Rufus will take this feedback and use it to update his language model. He will try to say the sentence again, using the new information he received from the human. This time, he might say "I'm a robot."

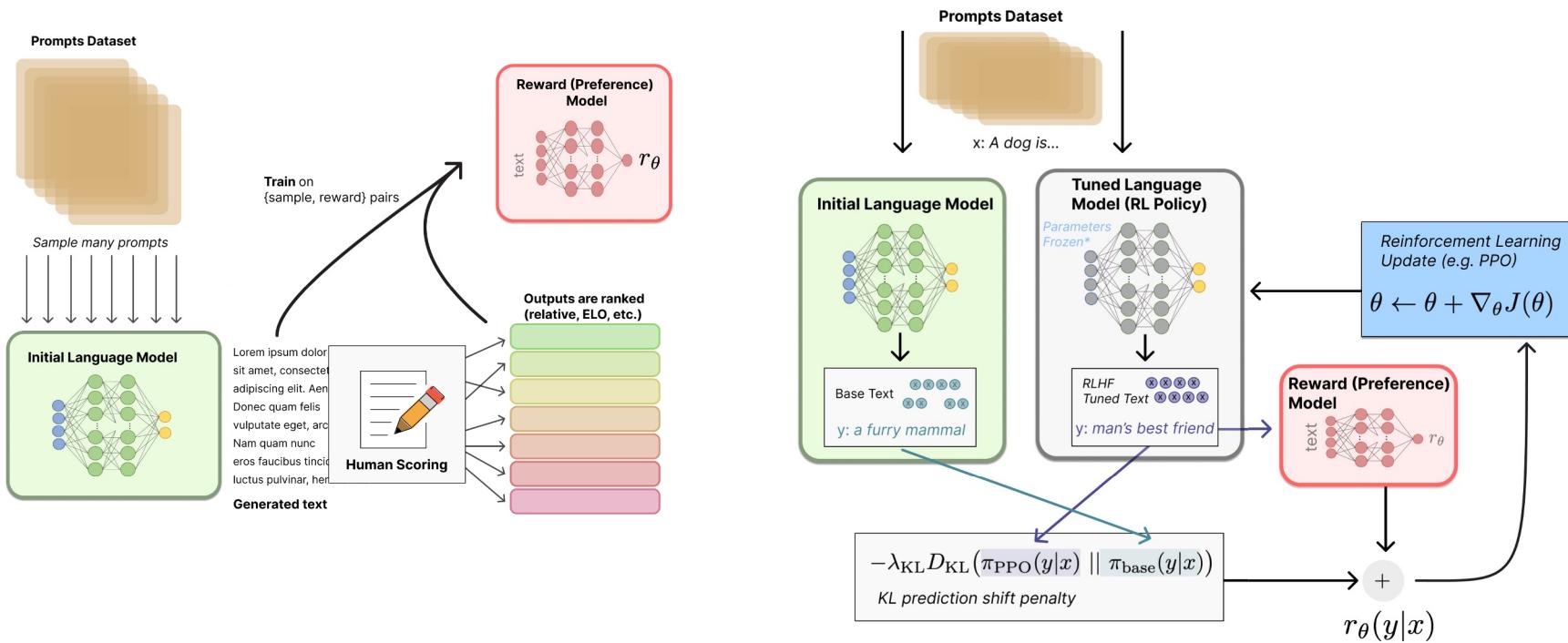
The human will listen again and give Rufus more feedback. This process will continue until Rufus can say sentences that sound natural to a human.

Over time, Rufus will learn how to talk like a human thanks to the feedback he receives from humans. This is how language models can be improved using RL with human feedback.

# Proximal Policy Optimization (PPO)

- Policy gradient:  $J(\theta) = E_{\tau \sim \pi_\theta} R(\tau) = \sum_{\tau} P(\tau; \theta)R(\tau)$        $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$
- Selecting a learning rate is a challenge; hard to pick a good learning rate given high variance in rewards
  - + Policy gradient has us toss trajectories after iterations
- PPO seeks to address these drawbacks.
- The popular implementation right now is called “Clipped PPO”
  - Upshot: dynamically adjust learning updates and keep past samples around as long as they seem ‘trustworthy’ even as the policy changes so we can learn more efficiently

# Reinforcement Learning from Human Feedback (RLHF)



$f: X \rightarrow Y;$   
 $f(x)=y$

$f: \phi_x(X) \rightarrow \phi_y(Y);$   
 $f(\mathbf{x})=\mathbf{y}$

$f: X \times \Theta \rightarrow Y;$   
 $f(x;\theta)=y; \theta \in \Theta$  for model M

$f: \phi_x(X) \times \Theta \rightarrow \phi_y(Y);$   
 $f(\mathbf{x};\theta)=\mathbf{y}$

- DL enables function approximation for  $f$  and is constrained by:
  - Data representation  $\phi$
  - Model architecture M
  - Loss function L
  - Optimization algorithm
  - Training data D

$$\theta^* = \min_{\theta \in \Theta} (\sum_{(x,y)} L(f(\mathbf{x};\theta), \mathbf{y}))$$

$$\theta^* = \min_{\theta \in \Theta} (\sum_{(x,y) \in D} L(f(\mathbf{x};\theta), \mathbf{y}))$$

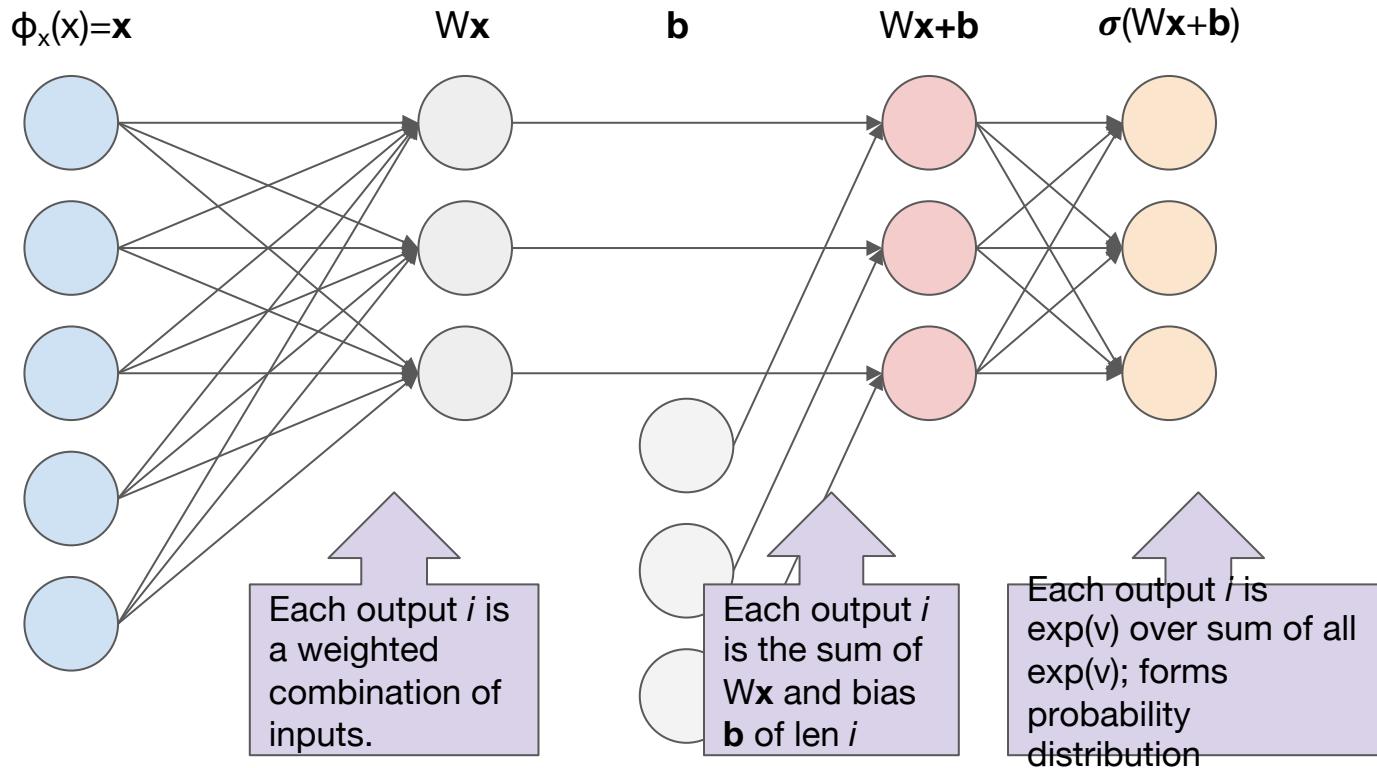
# Framing Problems for Machine Learning

- If you can:
  - Squeeze your input and output into numerical representations (e.g., vectors)
  - Compose neural layers that eat something shaped like the input and spit out something shaped like the output
  - Define a loss function and apply an optimizer that are valid in the underlying problem and model space
- Then:
  - Your DL model will train and loss will go down, *even if you aren't actually solving the problem you wanted to solve*

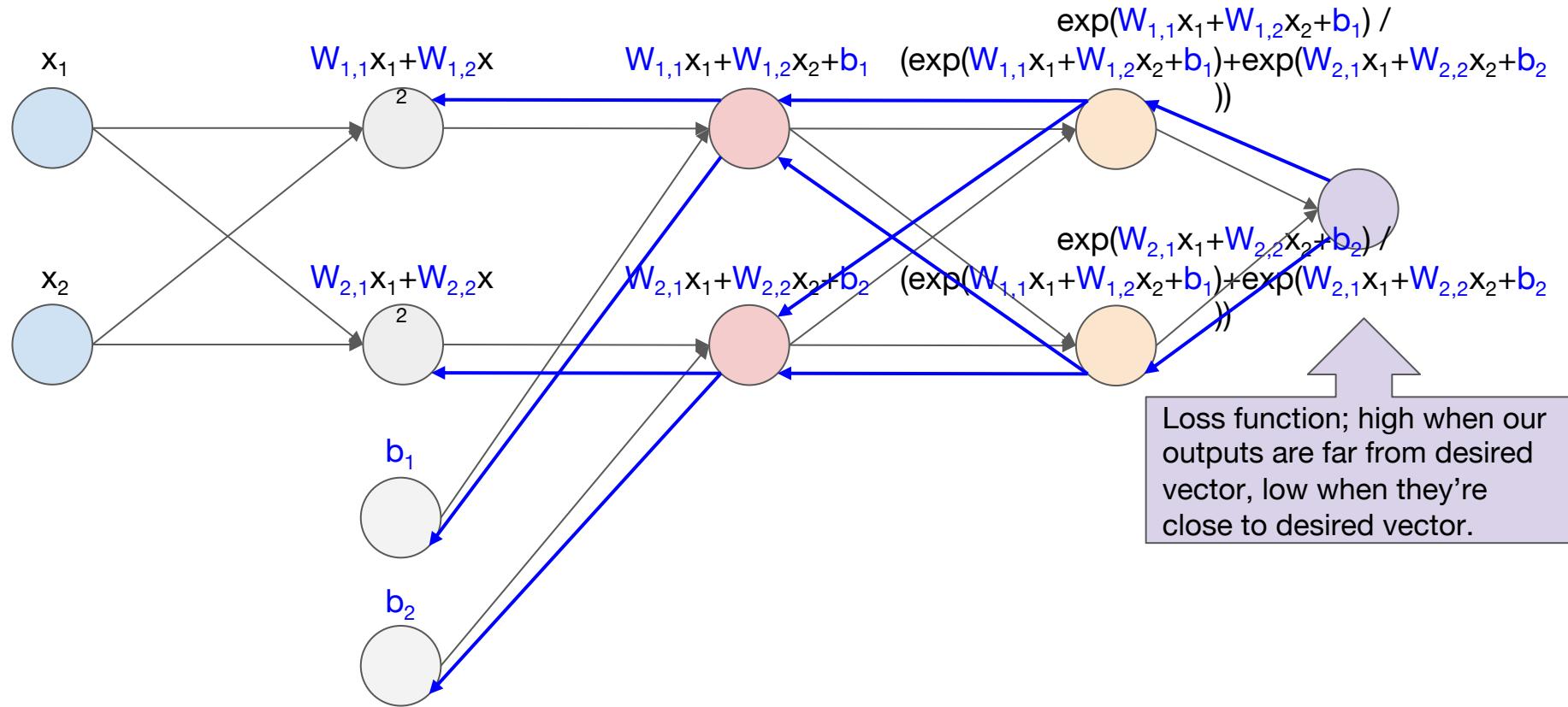
# Framing Problems for Machine Learning

- What is the most general formulation of the goal of Machine Learning we can express?
  - Given an input  $x \in X$ , produce a desired output  $y \in Y$ , using a *learned function*  $f(x) \approx y$ .
- What's a simple, non-degenerate learned function?
  - First assume we can transform input  $x$  into a *vector representation*,  $\phi(x) = \mathbf{x}$
  - Then we can study a simple linear model:  $f(\phi(x)) = W\phi(x) + b$
  - *Weight matrix*  $W$ , *bias vector*  $b$

$$f(\mathbf{x}; \theta) = \sigma(W\mathbf{x} + \mathbf{b})$$



# Backwards Pass



# Where Does Deep Learning Come In?

- Linear function:  $f(\mathbf{x}, \theta) = \mathbf{W}\mathbf{x} + b$
- What loss function could we use for this  $n$ -ary classification problem?
  - Cross entropy
- Non-linear transformation:

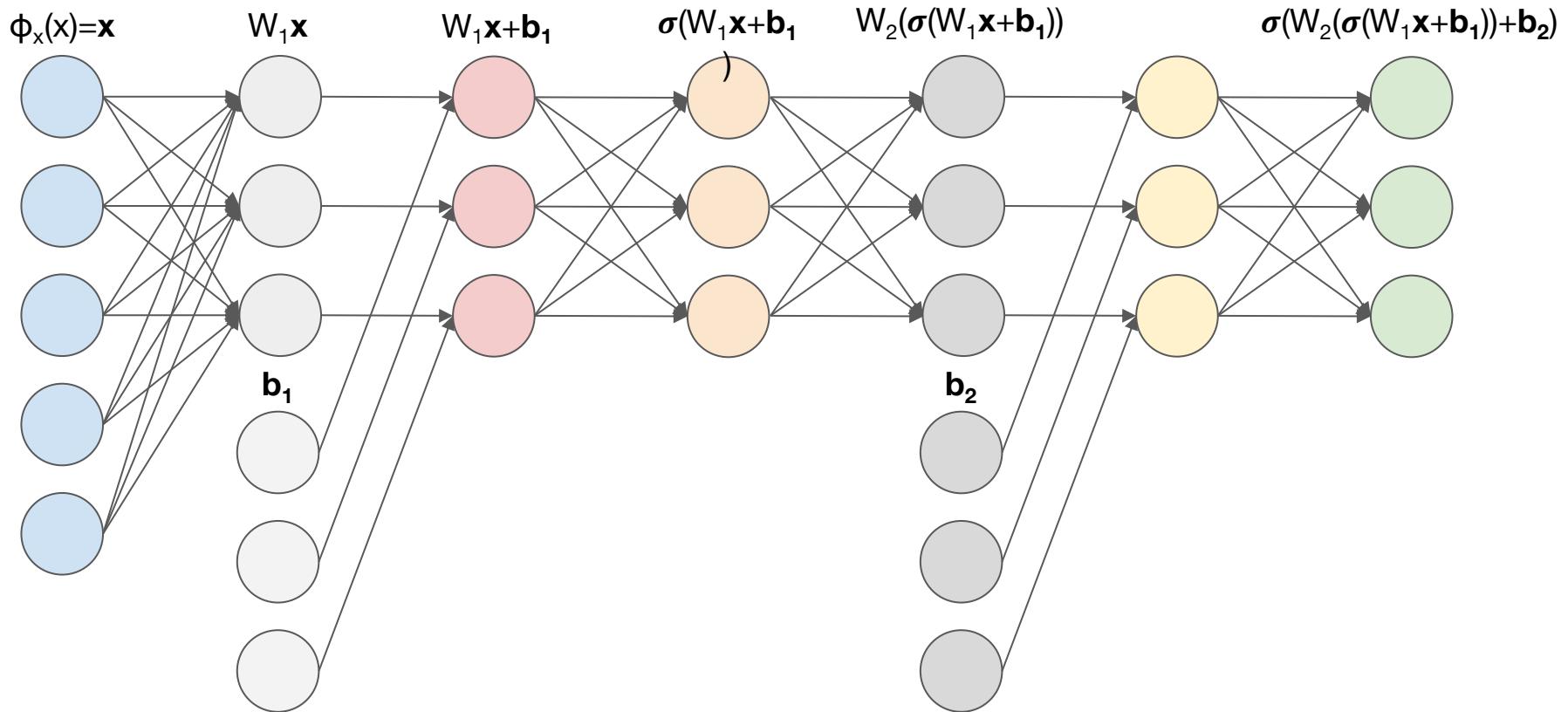
$$\text{Loss} = -\frac{1}{N} \sum_{i=1}^N \hat{y_i} \log(P(y_i = j | x_i)) \quad P(y_i = j | x_i) = \frac{e^{f(x_i; W, b)}}{\sum_{j=1}^C e^{f(x_j; W, b)}}$$

---

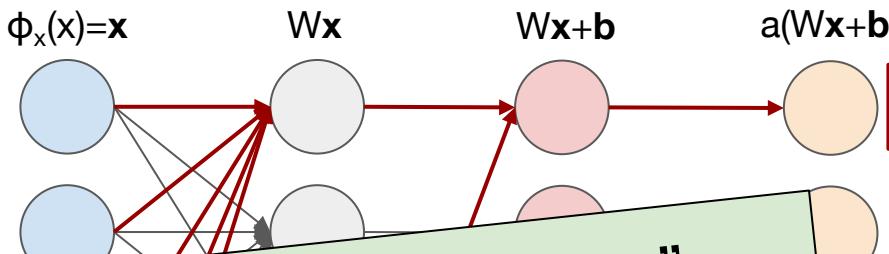
- Our function can only capture the degree to which each dimension of the input,  $\mathbf{x}_i$ , contributes to the likelihood of each class  $y_j$ 
  - "... brought out the ice cold food."
  - "... brought out the ice cold gazpacho."
  - "... Amazing service all around. Yeah right! 😊"

$$f(\mathbf{x}, \theta) = \sigma(W_2(\sigma(W_1\mathbf{x} + \mathbf{b}_1)) + \mathbf{b}_2)$$

$$W_2(\sigma(W_1\mathbf{x} + \mathbf{b}_1)) + \mathbf{b}_2$$



# Activation Functions



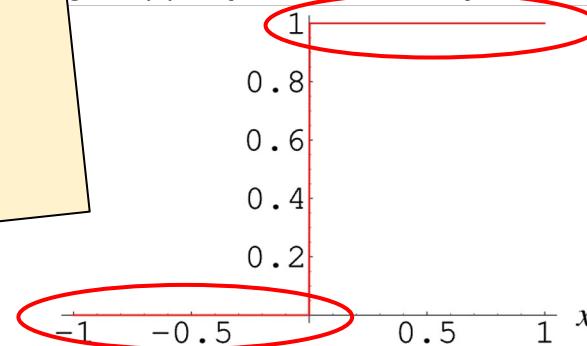
$$= a(W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + W_{1,4}x_4 + W_{1,5}x_5 + b_1)$$

What's the “problem” with defining  $a$  this way?

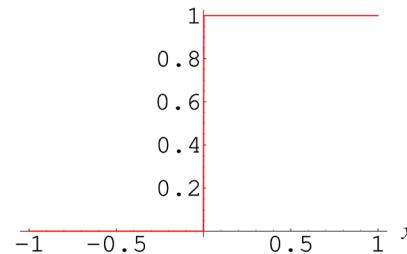
To perform backpropagation, we need differentiable functions with non-zero gradients.

If we think about a “neuron” as inspiration, the output after  $a$  should be 1 if  $W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + W_{1,4}x_4 + W_{1,5}x_5 + b_1$  exceeds a threshold and 0 else.

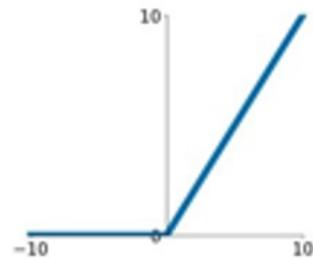
$$\text{g., } a(x) = \{1 \text{ if } x > 0 \text{ else } 0\}$$



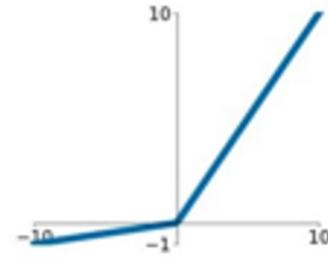
# Activation Functions



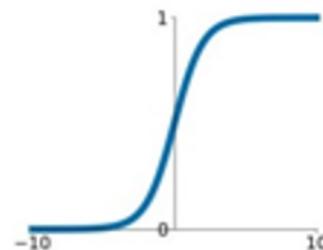
**ReLU**  
 $\max(0, x)$



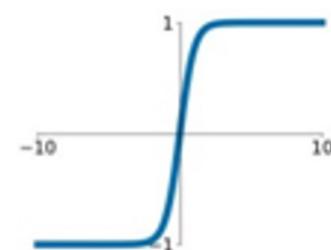
**Leaky ReLU**  
 $\max(0.1x, x)$



**Sigmoid**  
 $\sigma(x) = \frac{1}{1+e^{-x}}$



**tanh**  
 $\tanh(x)$



# Initialization, Normalization, Regularization

- Activations are well-behaved close to zero
- Weight initialization:
  - Gaussian; need to pick variance
  - Glorot / He initializations choose a variance based on activation function and number of incoming connections
- Data normalization: z-scoring features to make them Gaussian
- Model regularization:
  - Dropout for implicit ensemble learning
  - BatchNorm layers to address covariate shift through layers

# Origins of (Widespread) Deep Learning: AlexNet



mite      container ship      motor scooter      leopard



mite  
black widow  
cockroach  
tick  
starfish



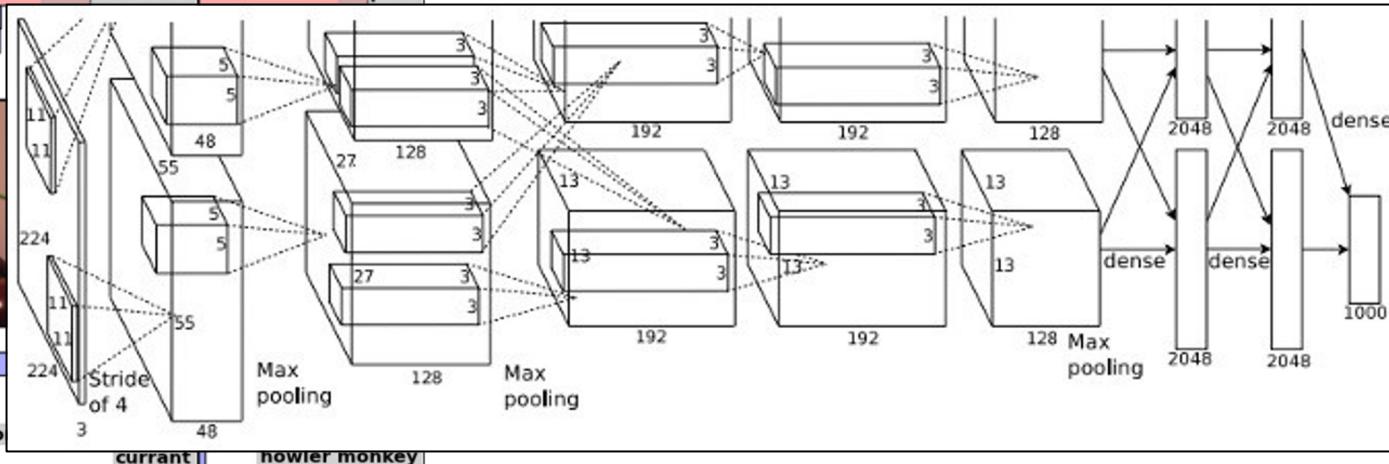
container ship  
lifeboat  
amphibian  
fireboat  
drilling platform



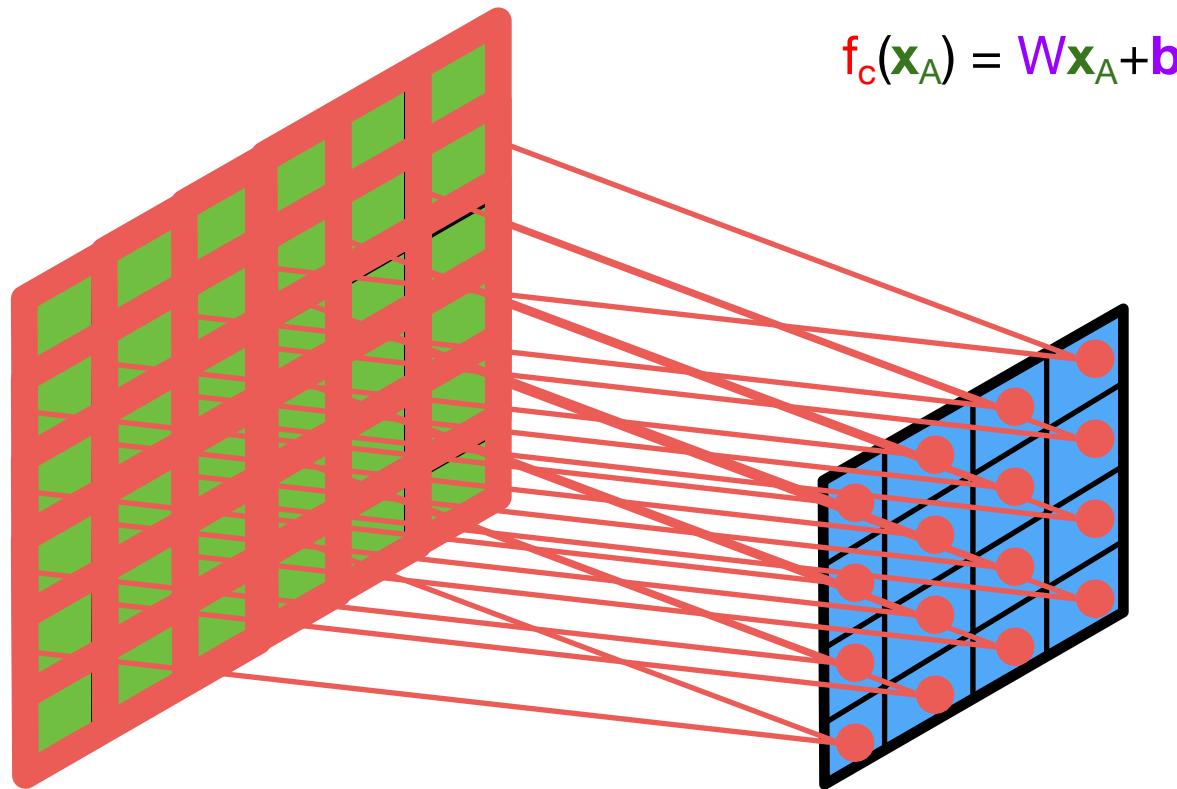
grille  
mushroom



convertible  
grille  
pickup  
beach wagon  
fire engine  
dead-man's-fingers  
ffo



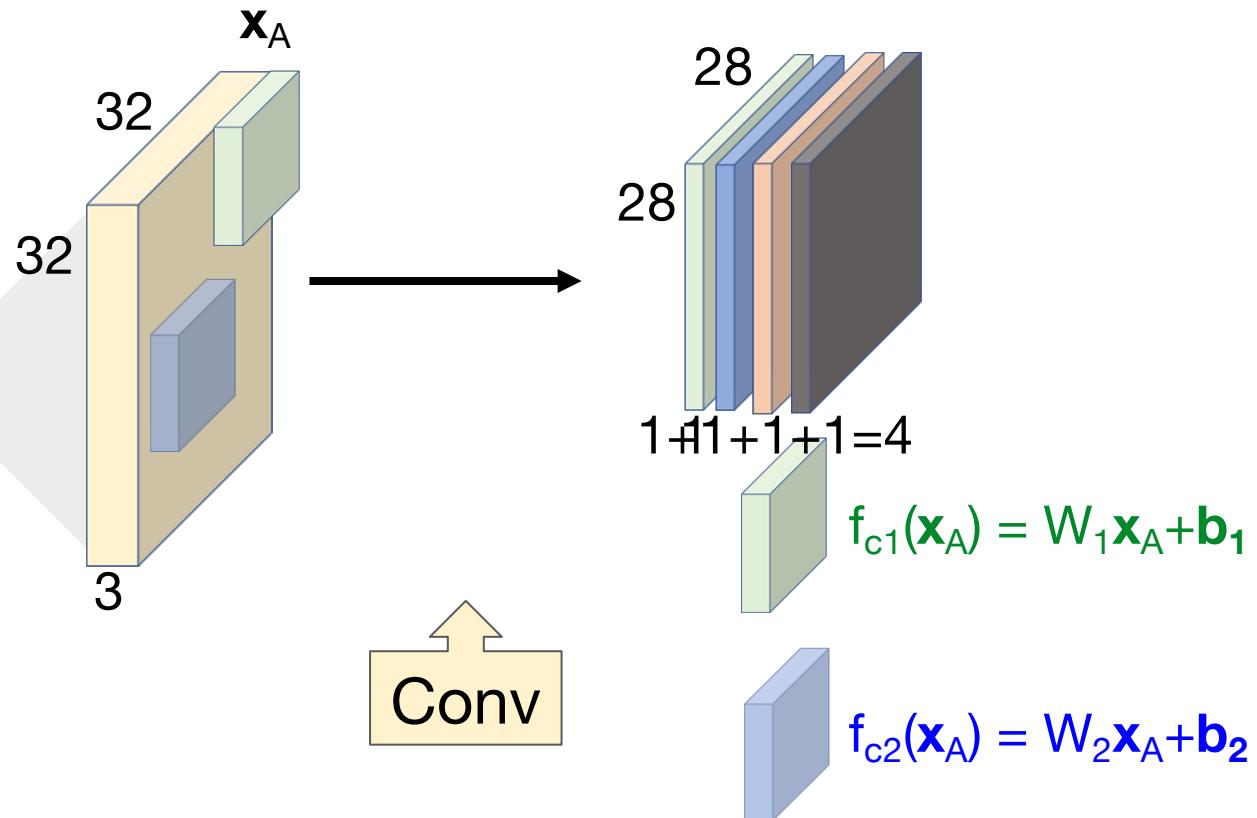
# Convolutional Layer



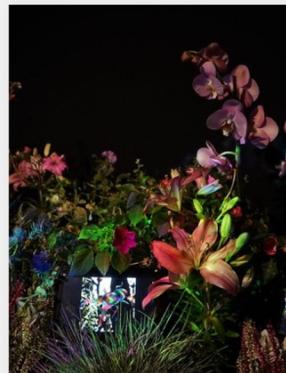
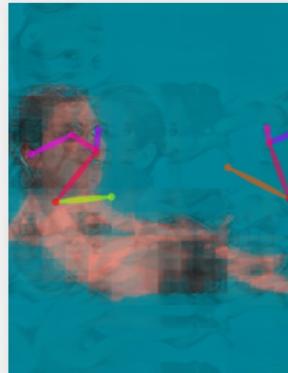
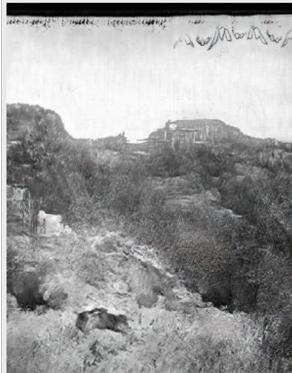
# Convolutional Layer



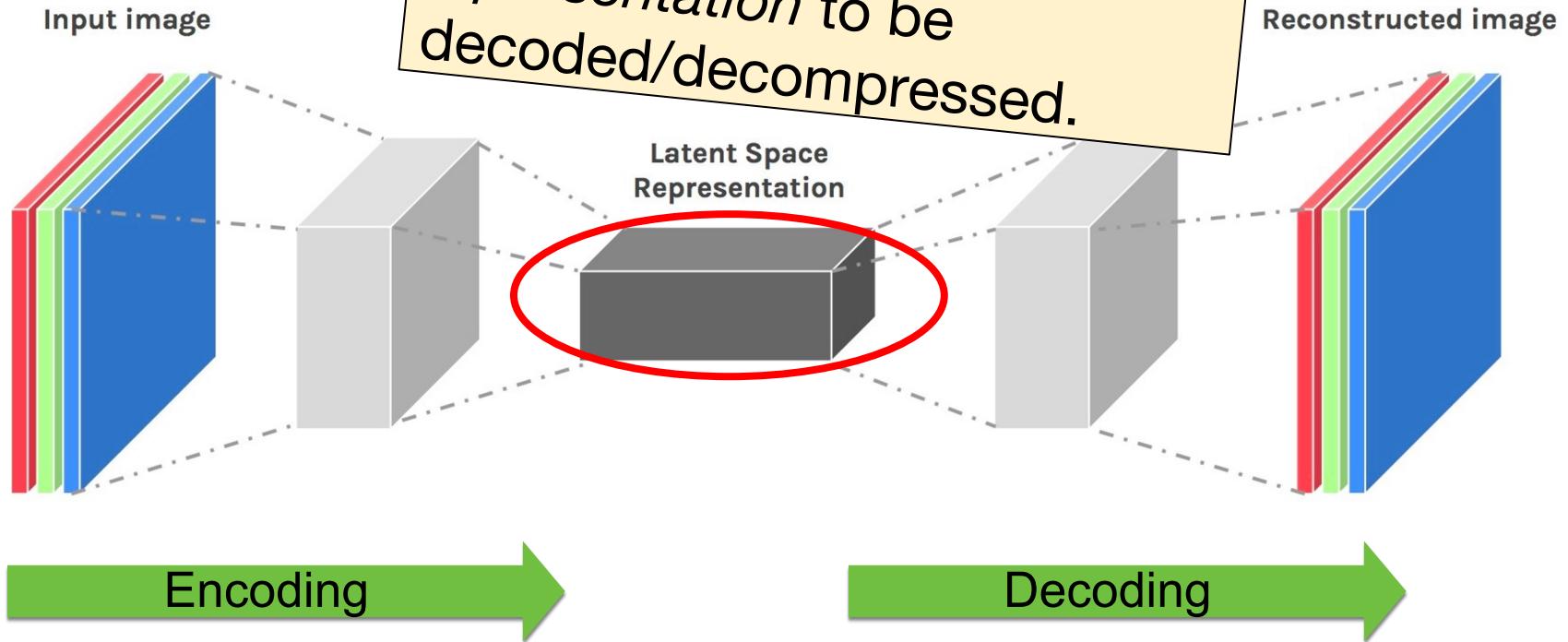
$$\phi_x(\text{ }) = [0, 1]^{(32 \times 32 \times 3)}$$



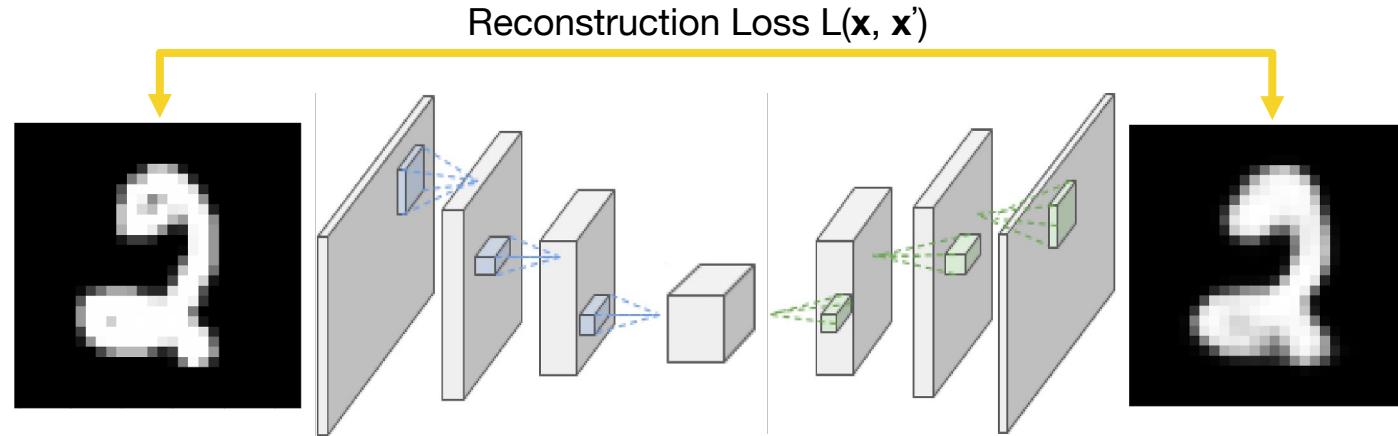
# Deep Learning for Vision: Generation



# Autoencoder



# CNN Autoencoder



Input  $x$

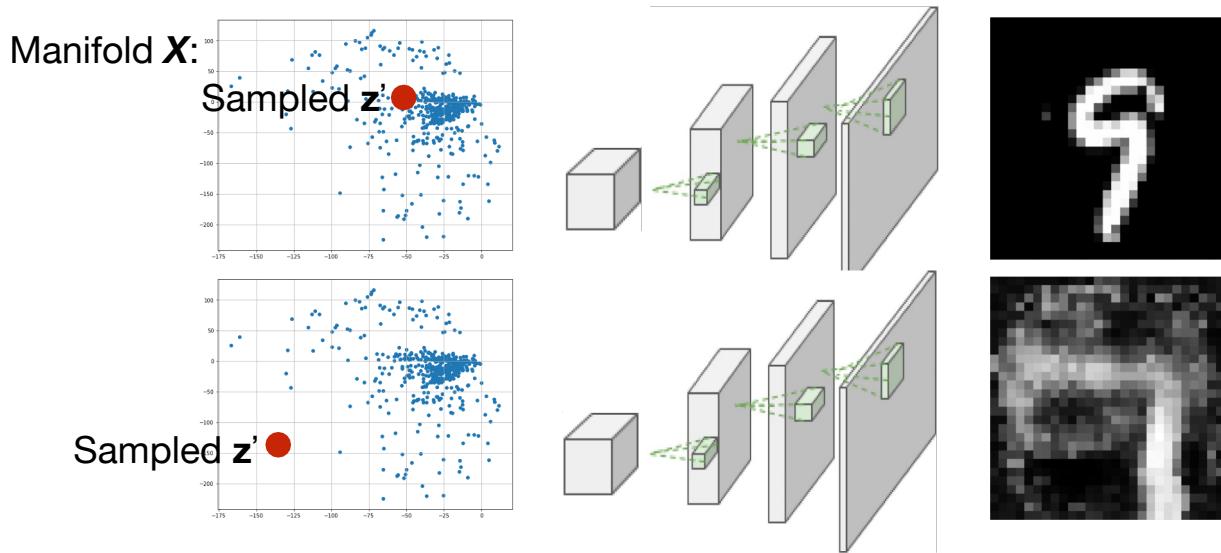
Encoder  
 $f_e(x; \theta_e) =$   
 $z$

Latent  $z$

Decoder  
 $f_d(z; \theta_d) = x'$

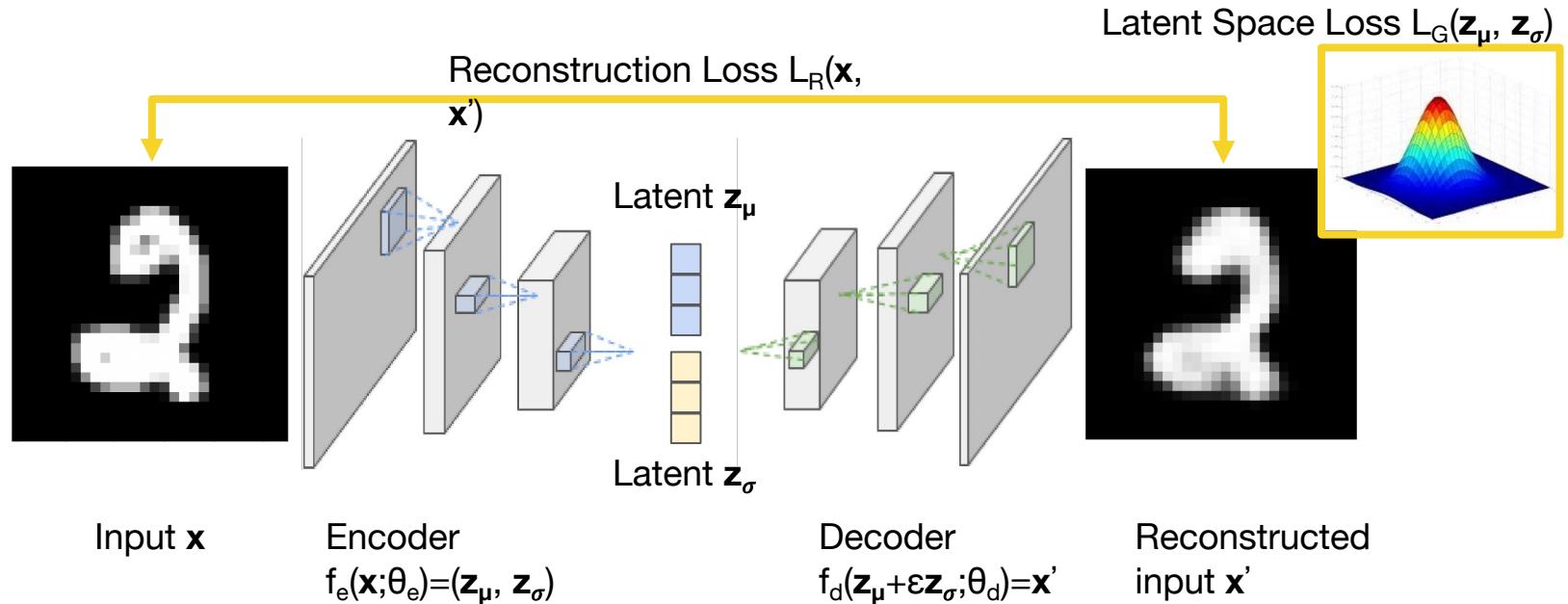
Reconstructed  
input  $x'$

# Autoencoder Decoder As Data Generator



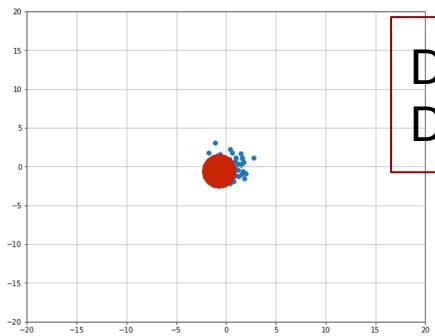
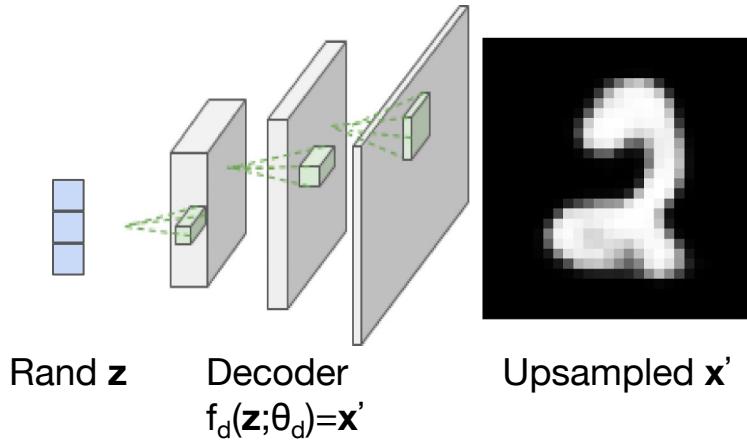
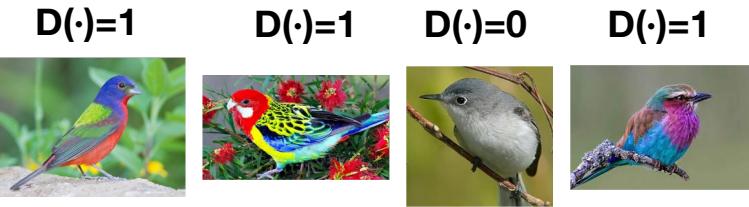
- What if we sample vector  $z'$  as draws from a Gaussian?
- Will we achieve reconstructions that look like our input data  $X$ ?
  - No! Our data  $X$  encodes to a particular manifold in  $R^{|z|}$

# Variational Autoencoder

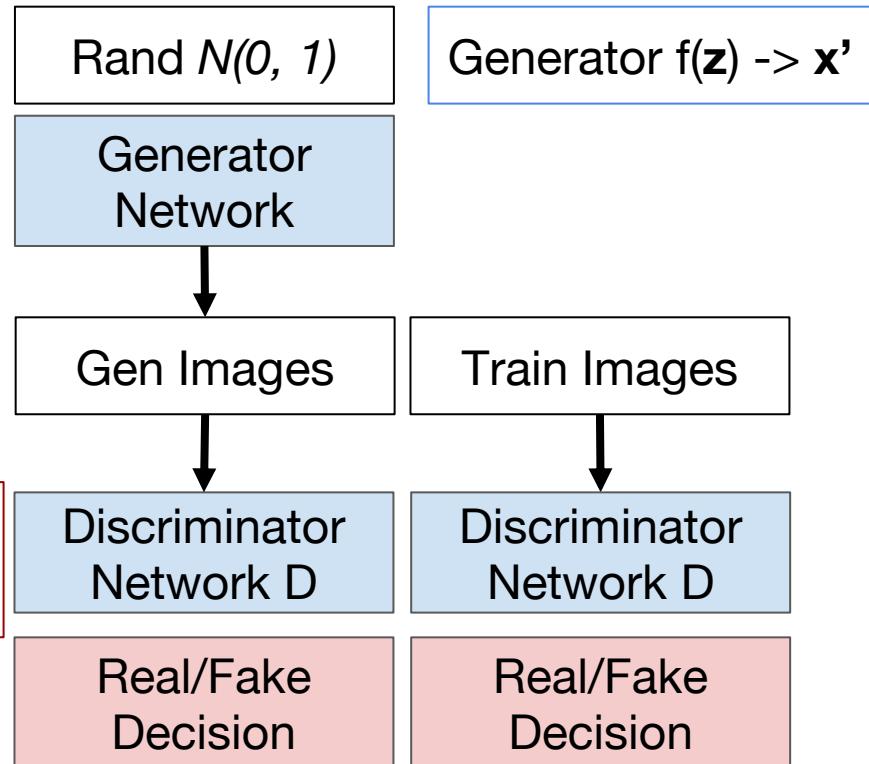


- We learn a mean and variance for the latent manifold for every input, then constrain those to be Gaussian

# Generative Adversarial Networks



Discriminator  $D(\mathbf{x}') \rightarrow 0$   
Discriminator  $D(\mathbf{x}) \rightarrow 1$



# Classes of Sequence-to-Sequence Problems

- One-to-one
  - One input produces one output
    - Image classification

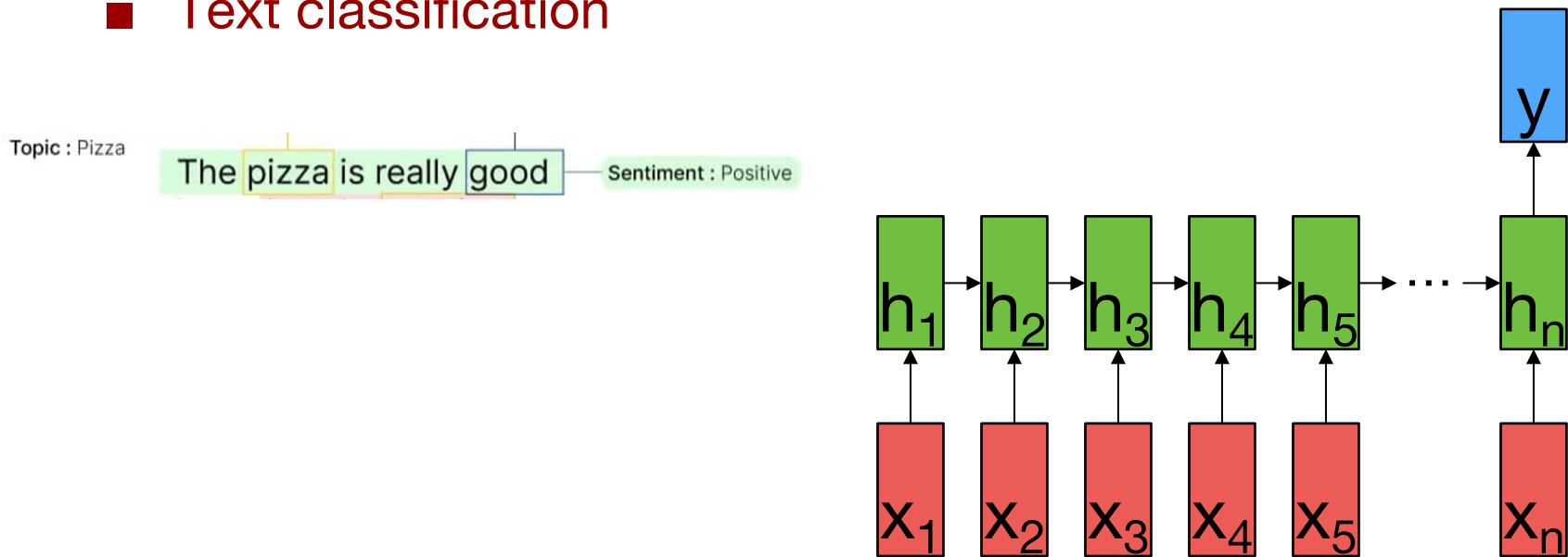


Person



# Classes of Sequence-to-Sequence Problems

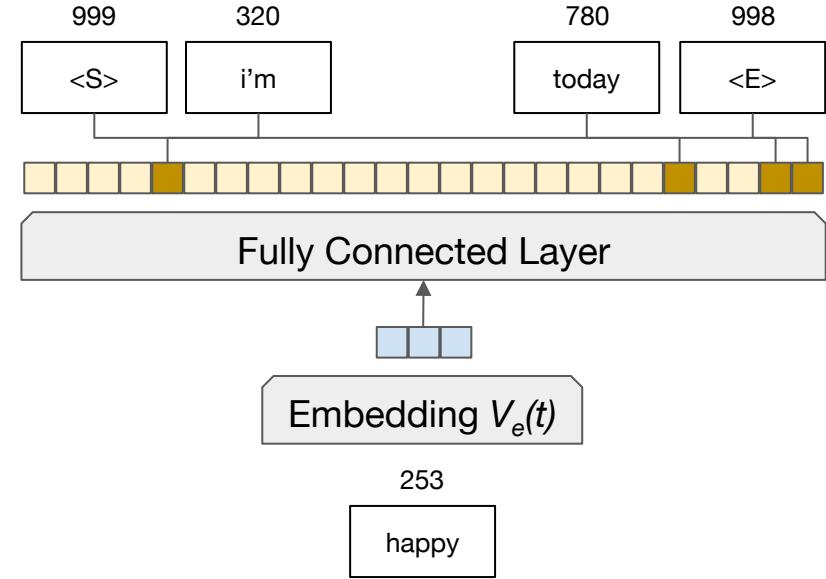
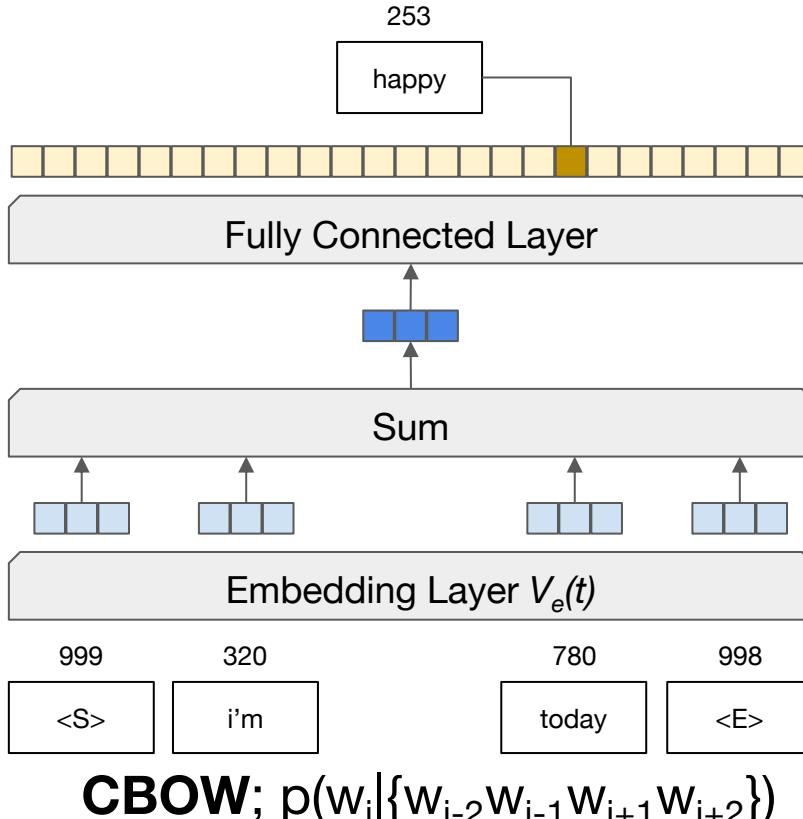
- Many-to-one
  - Sequence of inputs produces a single output
    - Text classification



## *n*-Gram Model

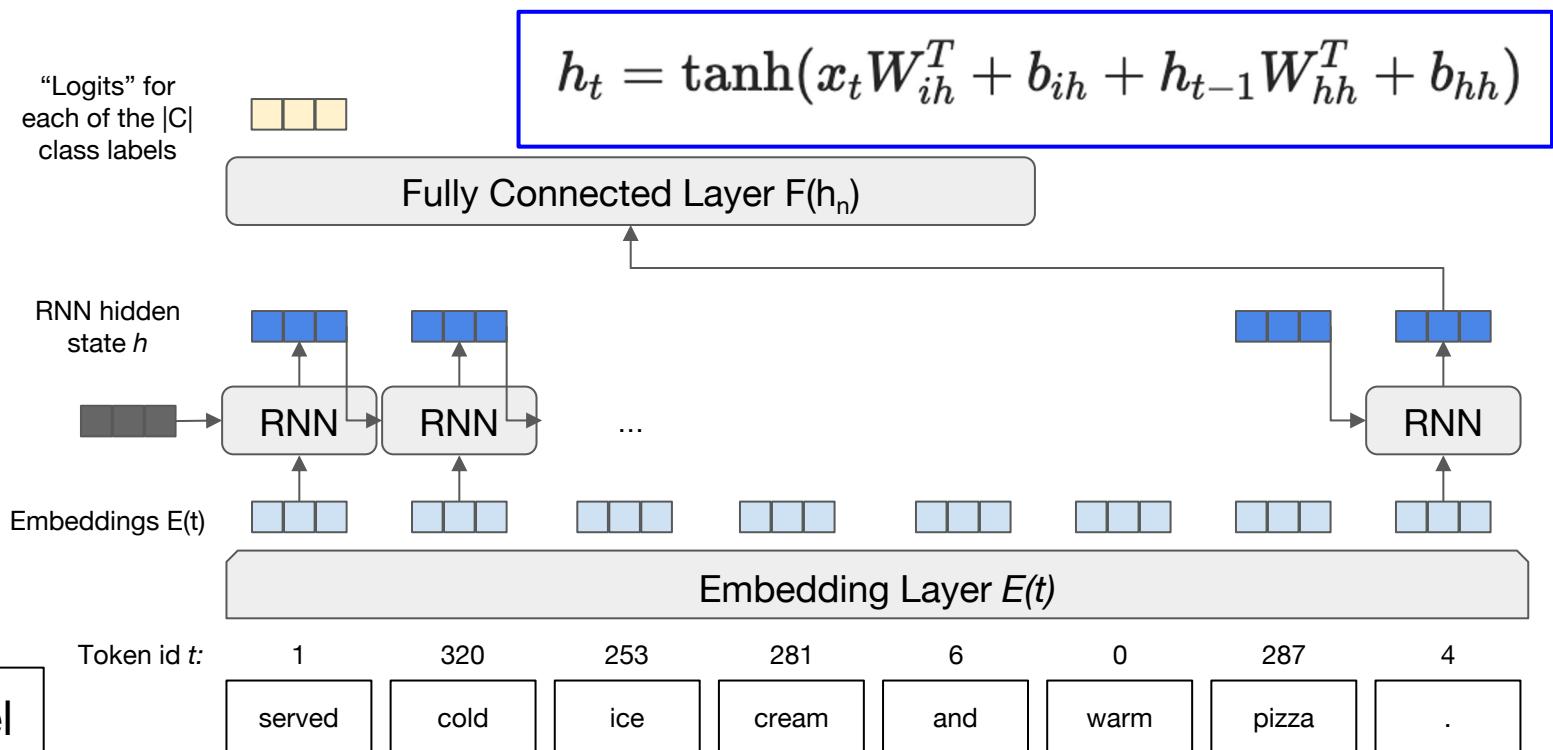
- An *n*-gram model represents the context of each word  $w$  as its co-occurrence with other words 1-, 2-, ...,  $n-1$ -steps away
- *n*-Grams are based on rigid, explicit counting, plus some smoothing to relax that rigidity
- How else could we relax  $p(w_i|w_{i-1}w_{i-2}\dots w_{i-n+1})$ ?
  - Previously, we kept a  $|V|$  count for each distance  $i-k$ 
    - We could just keep a  $|V|$  count that merges distances
    - “Bag of Words”-based model

# Word2Vec: Compressed Co-Occurrence Matrices

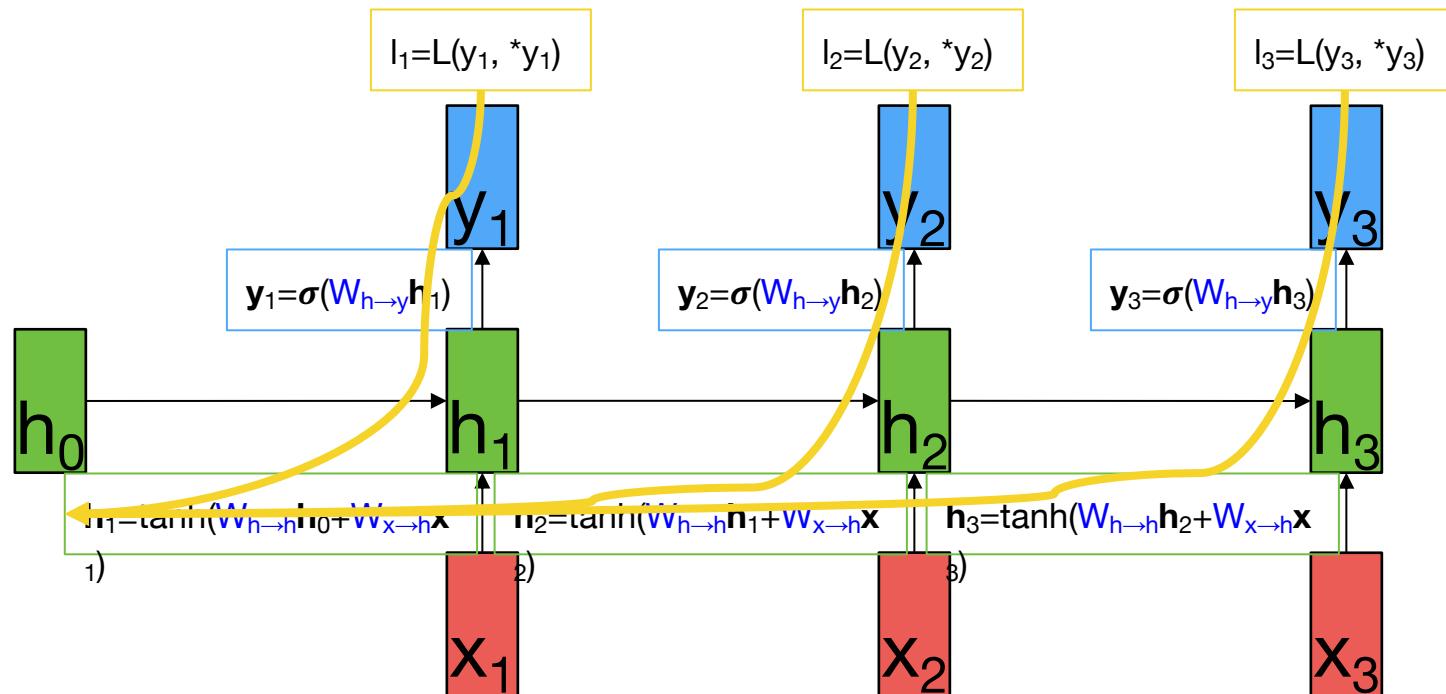


**Skip-Gram;**  
 $p(w_{i-2}|w_i)p(w_{i-1}|w_i)p(w_{i+1}|w_i)p(w_{i+2}|w_i)$

# Text Classification with a Recurrent Neural Network



# Backpropagation for RNN



# Long short-term memory

- Intuition of LSTMs is to “flip” the default recurrence behavior from “squeeze out the history” to “mostly consider the history”

**Vanilla RNN**

$$h_t = \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

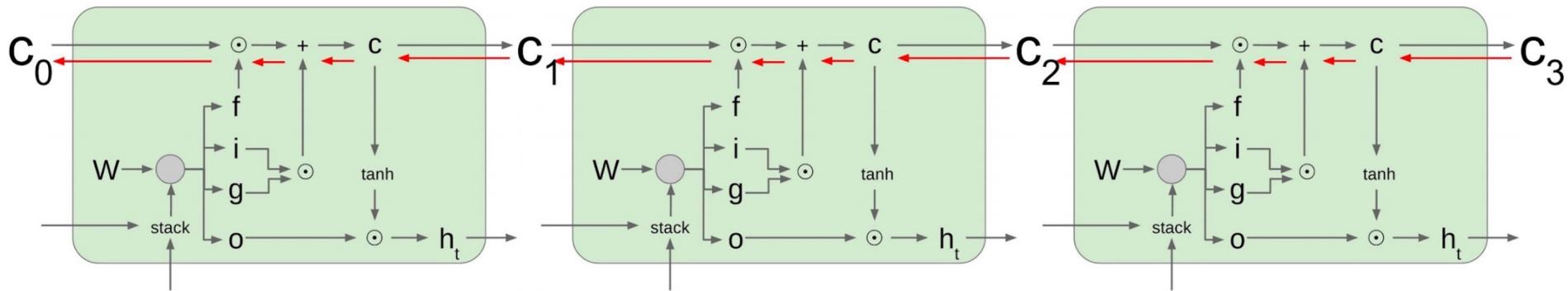
**LSTM**

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

# Long short-term memory

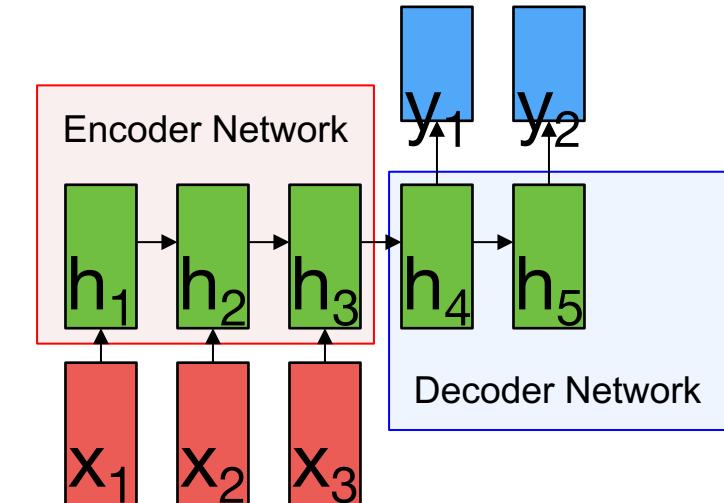
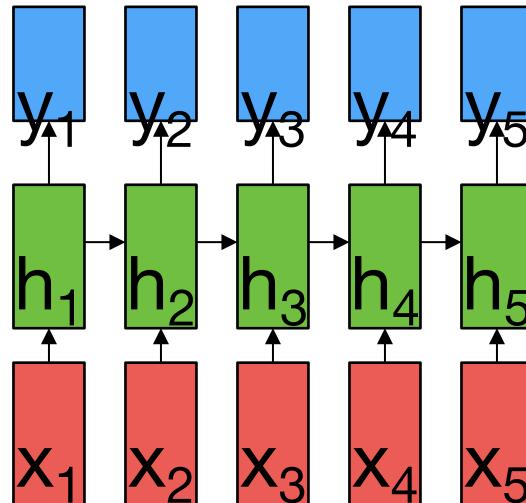


$$c_t = f \odot c_{t-1} + i \odot g$$

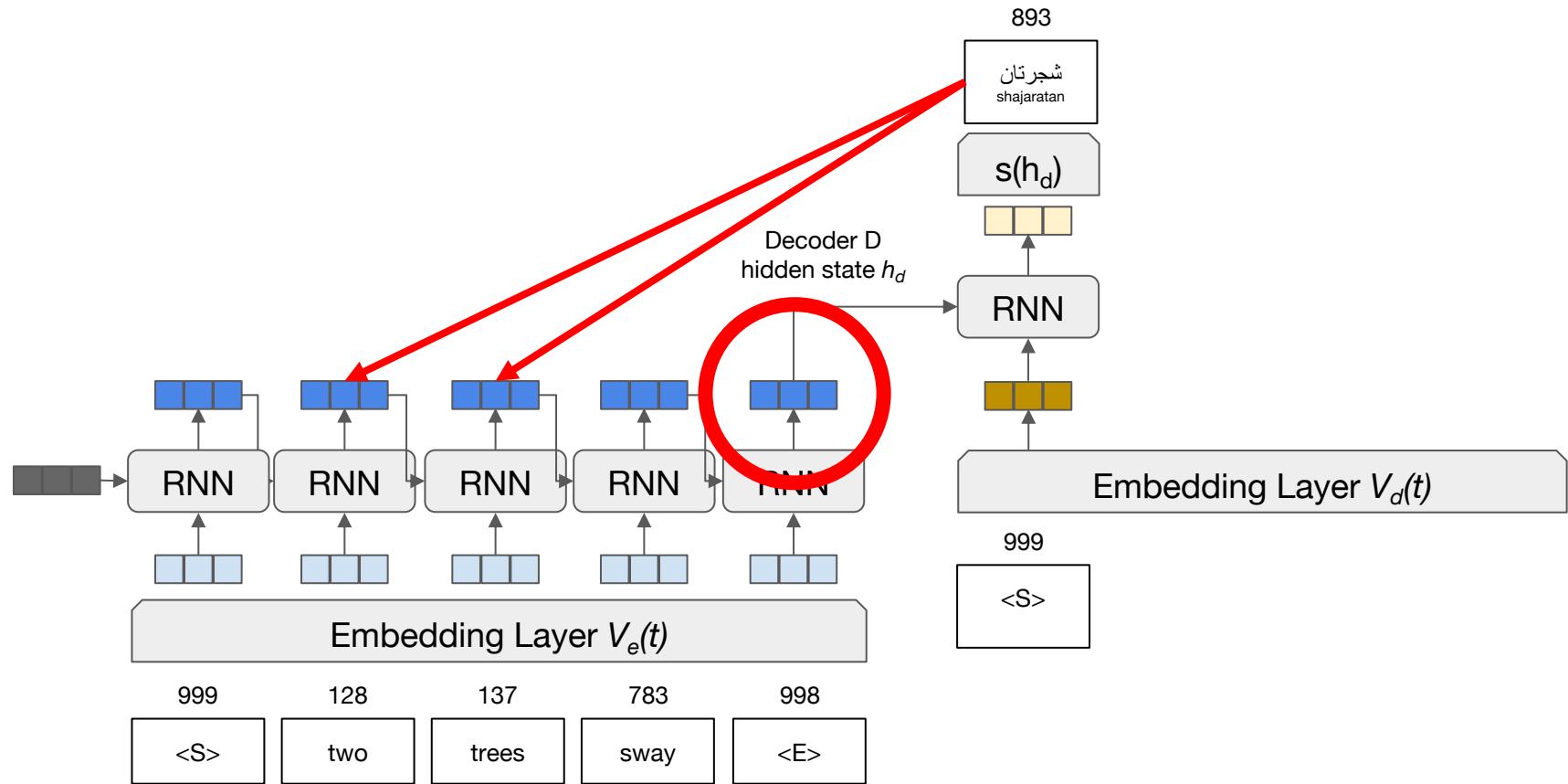
$$h_t = o \odot \tanh(c_t)$$

# Classes of Sequence-to-Sequence Problems

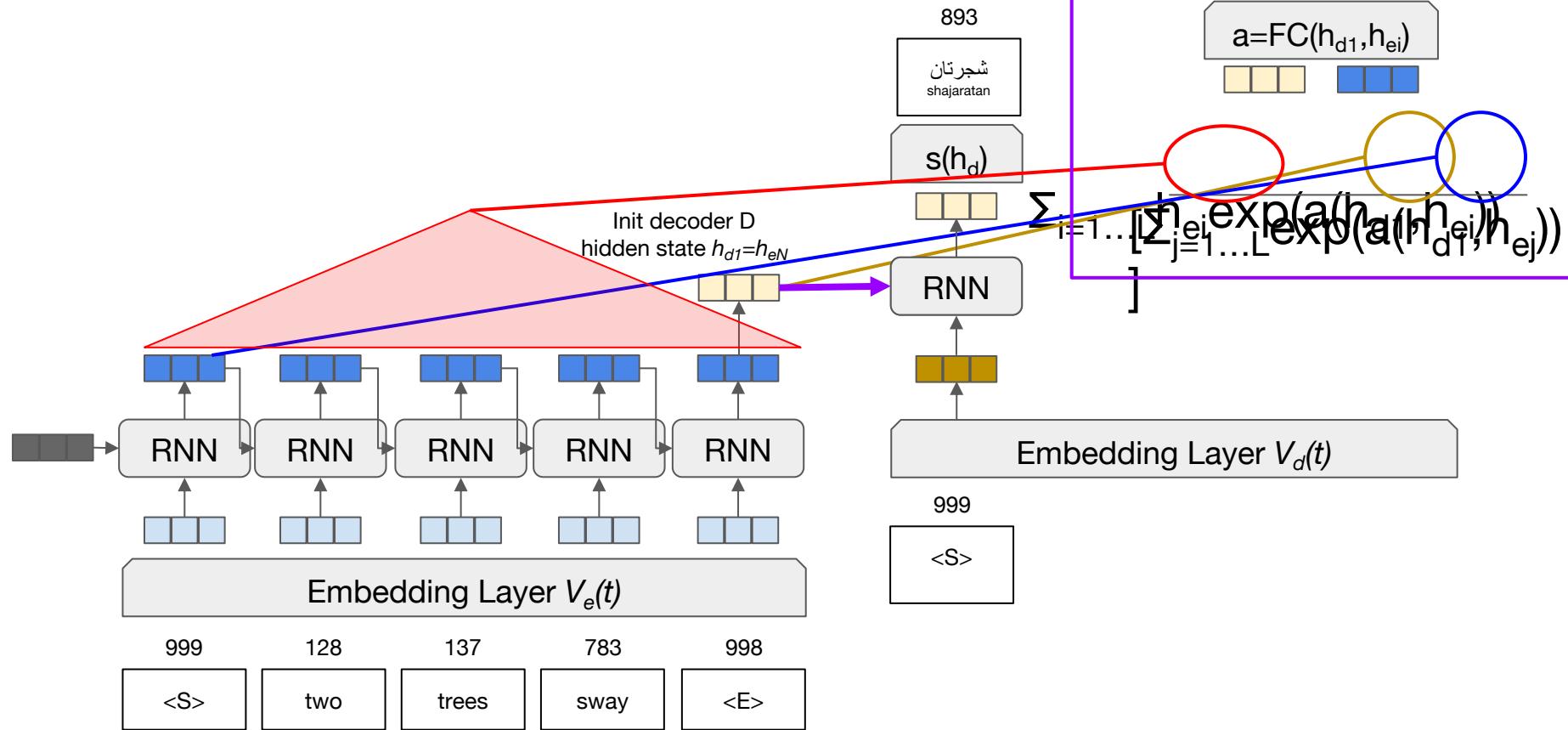
- Many-to-many variants:
  - Sequential one-to-one
    - Robot actions
  - Encoder-decoder
    - Machine Translation



# Machine Translation and The Case for Contextual Info



# Use Attention to Rewrite RNN State Input



# Most Recent High-Profile Deep Learning Technologies?



Write the complete script of a Seinfeld scene in which Jerry needs to learn the bubble sort algorithm. Include frequent jokes and note audience laughter in the script.



INT. MONK'S CAFE - DAY

ChatGPT

Jerry is sitting at the counter with George.

Jerry: (sighs) I can't believe I have to learn the bubble sort algorithm for my computer science class.

George: (laughs) Bubble sort? That's the most basic sorting algorithm there is. Even a monkey could do it.

(audience laughs)

Jerry: Yeah, well I'm not a monkey, I'm a comedian. (picks up menu)

Kramer enters and sits down next to Jerry.

Kramer: Hey, what's so funny?

Jerry: I have to learn the bubble sort algorithm for my computer science class.

Kramer: (laughs) The bubble sort? That's easy. You just keep swapping adjacent elements until the list is sorted.

(audience laughs)

Jerry: (confused) Swapping adjacent elements? I have no idea what that means.

Text-to-Image with Stable Diffusion

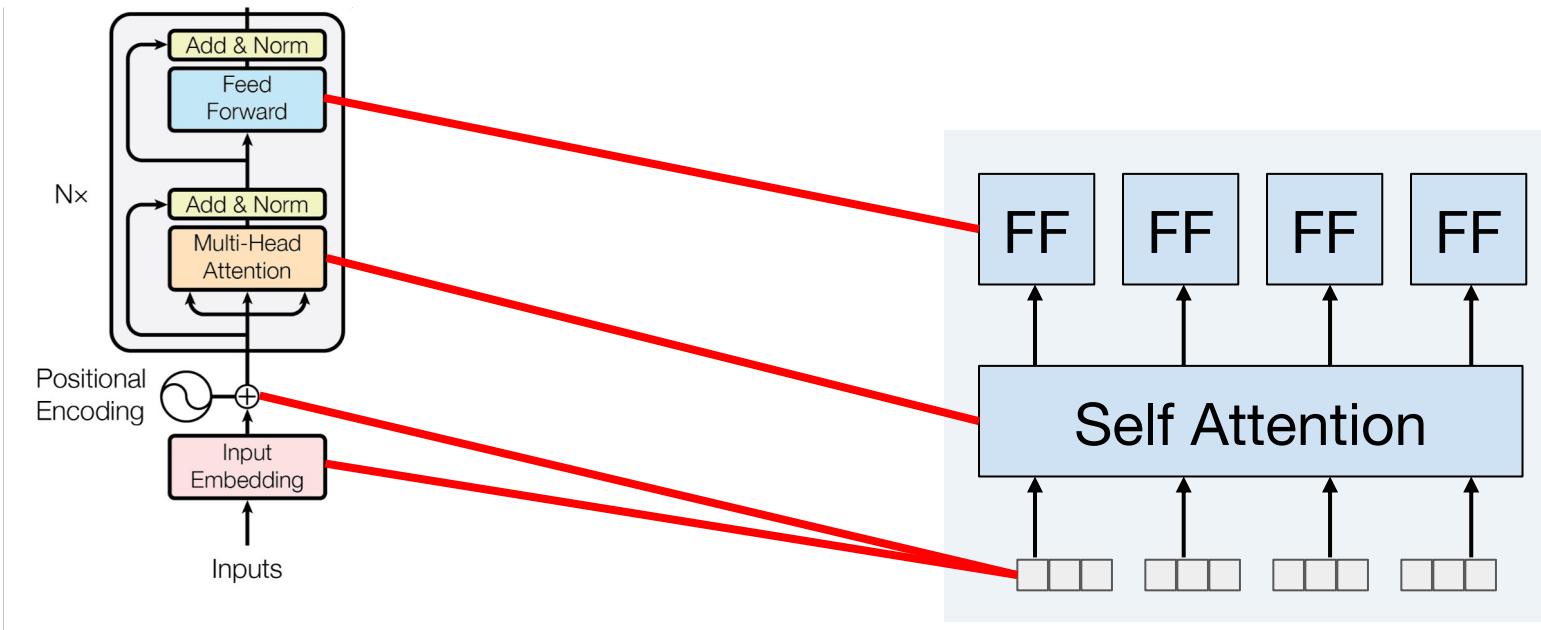


Stable Diffusion

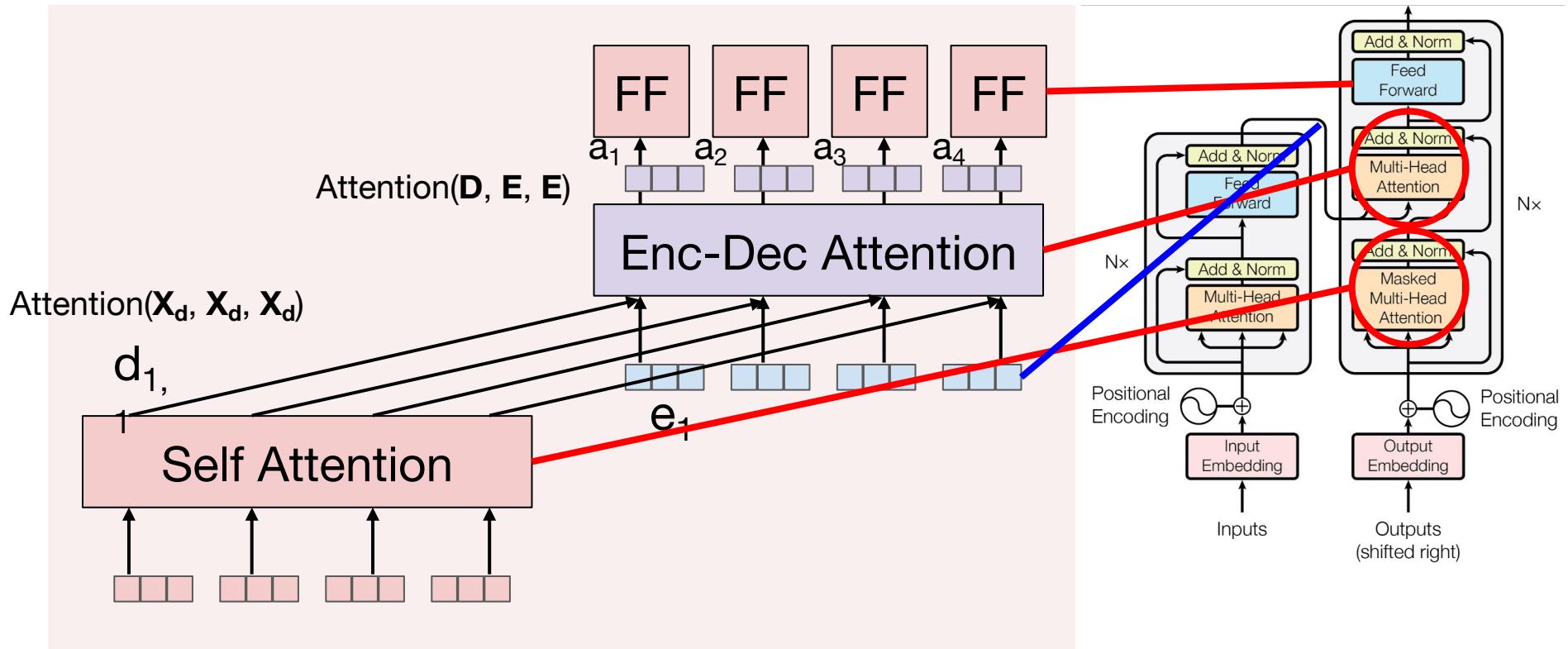
# Transformers

- Transformer  $T : S_a \rightarrow S_b$  is a function from...
  - A set  $a$  in  $S_a$  to a set  $b$  in  $S_b$
- Simple transformer consists of a single layer encoder  $E$  and a single layer decoder  $D$  with single-headed attention

# Transformers: Self-attention



# Encoder-Decoder Attention



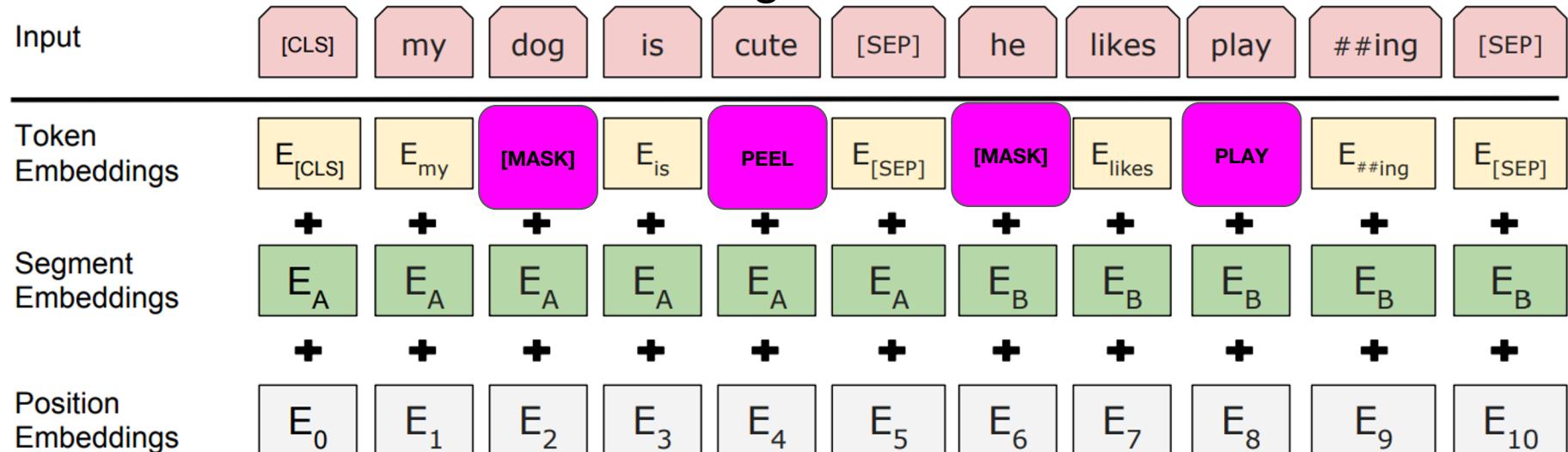
# Bidirectional Encoder Representations from Transformers

Input



# Masked Language Modeling Pretraining

15% of words selected for training

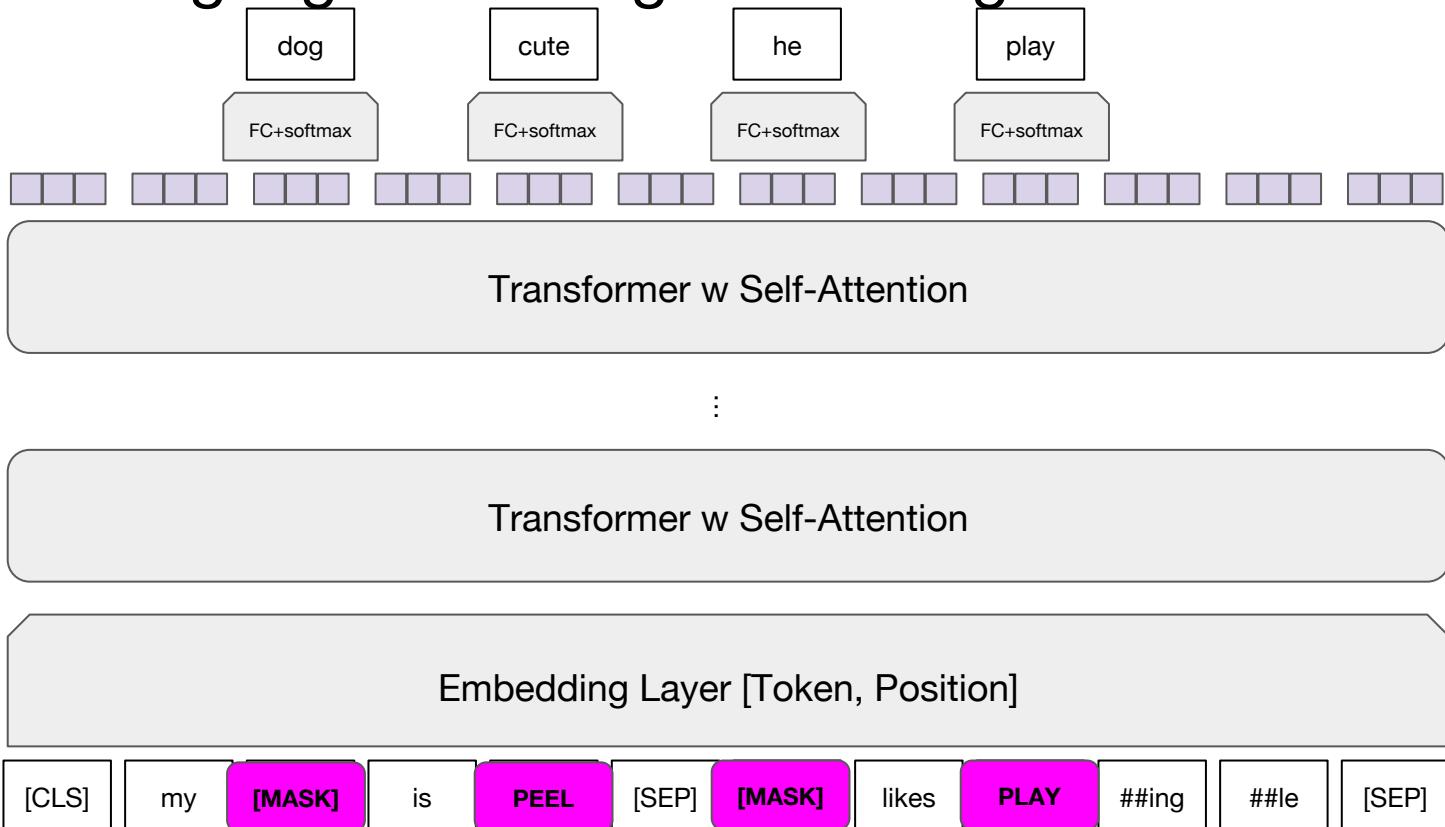


80% of those [MASK]

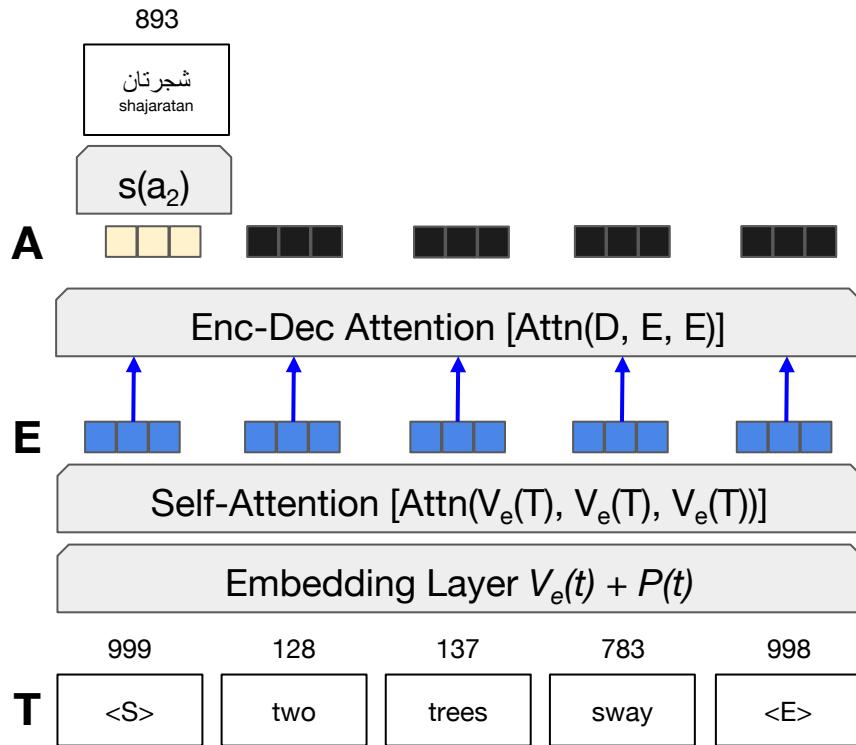
10% random token

10% original token

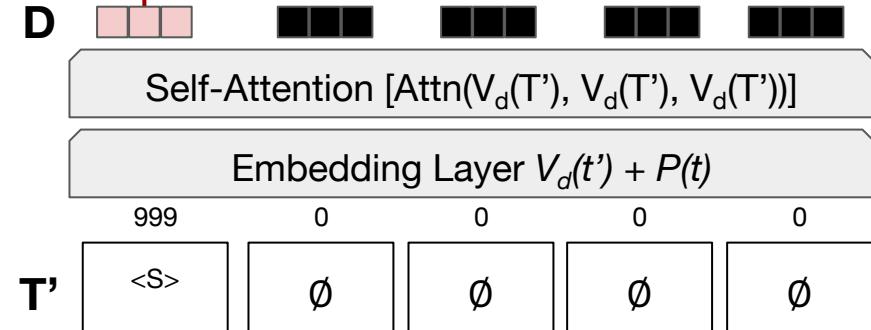
# Masked Language Modeling Pretraining



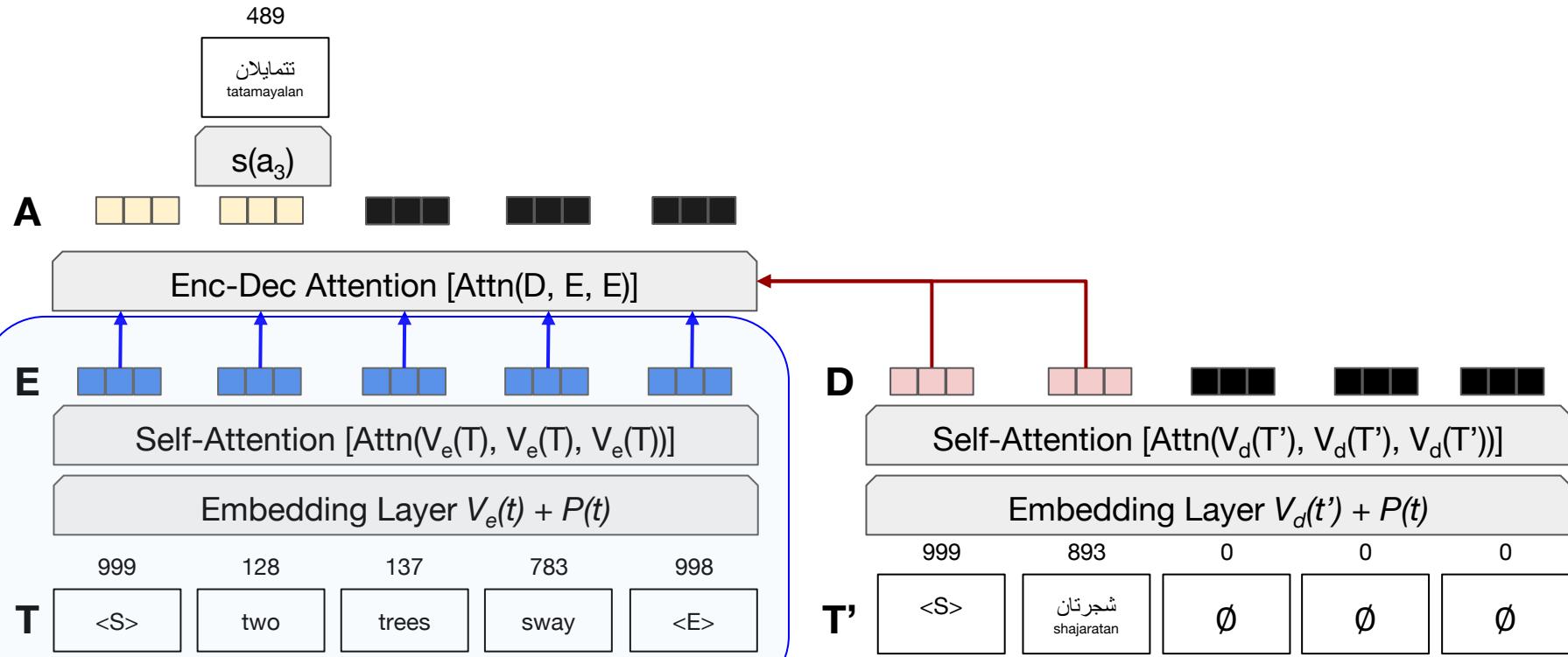
# Sequence-to-Sequence Transformer



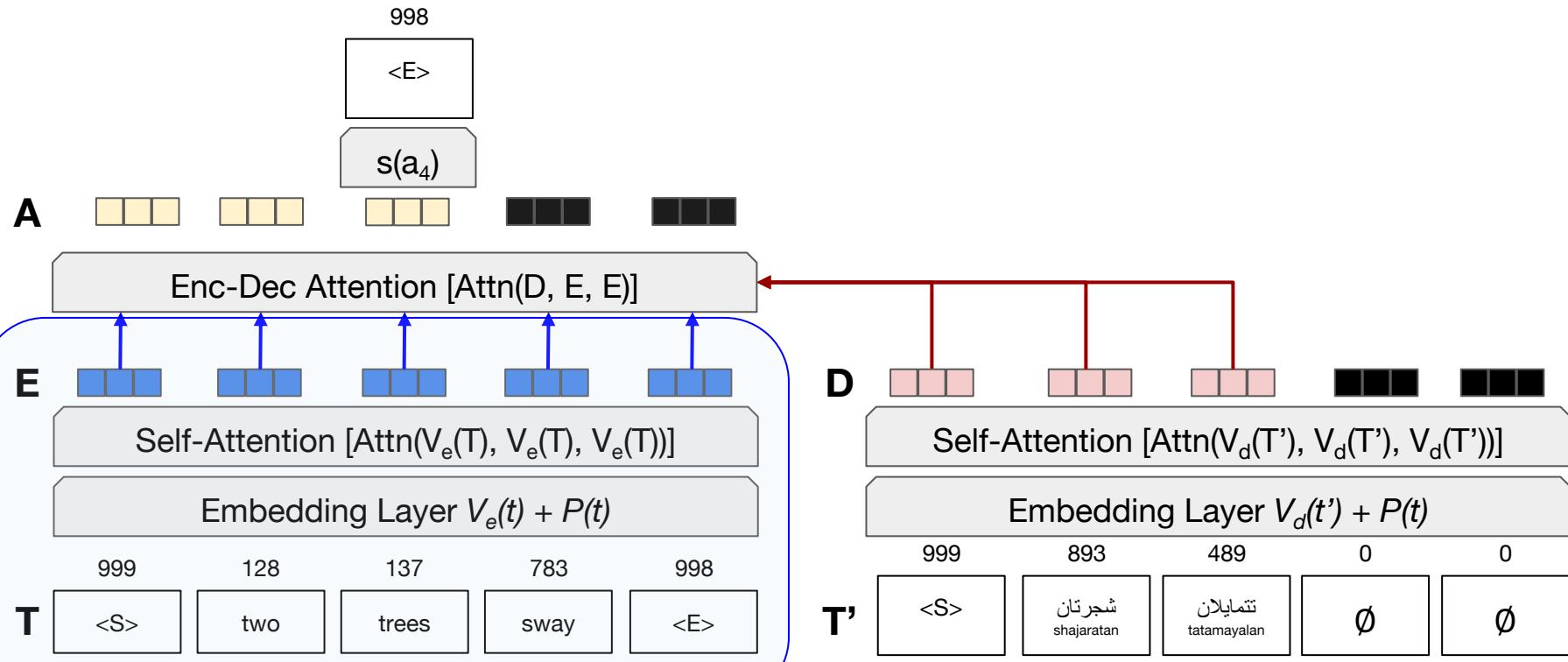
$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$



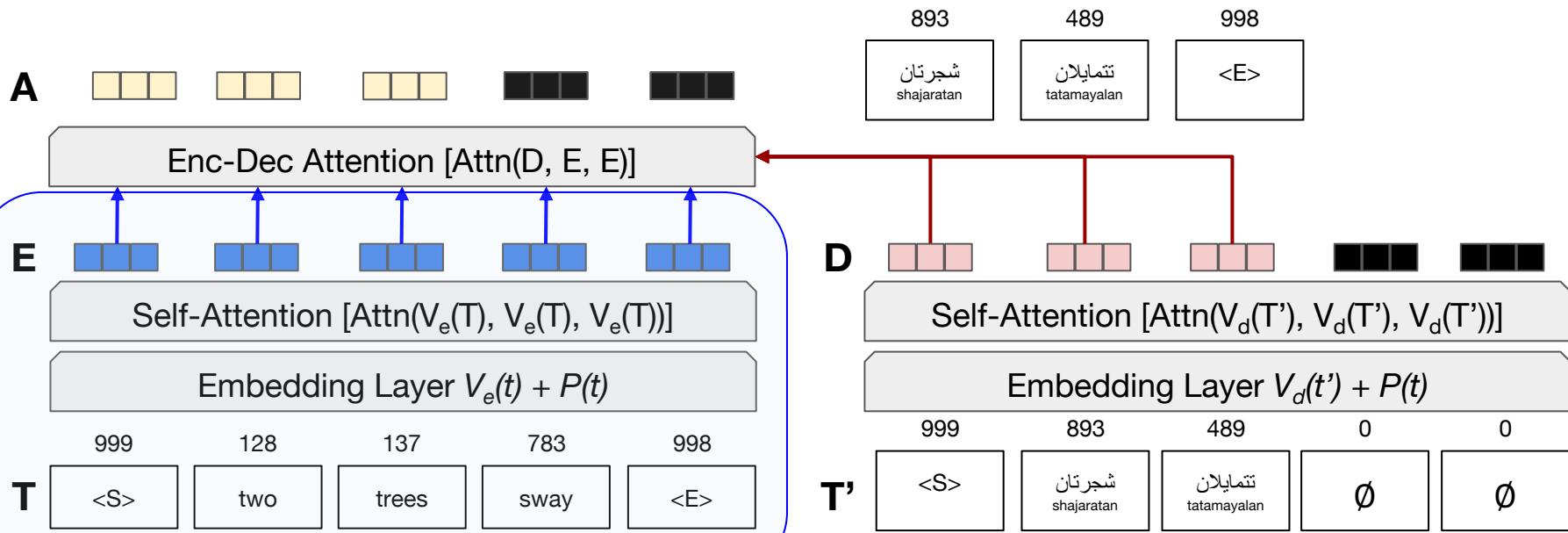
# Sequence-to-Sequence Transformer



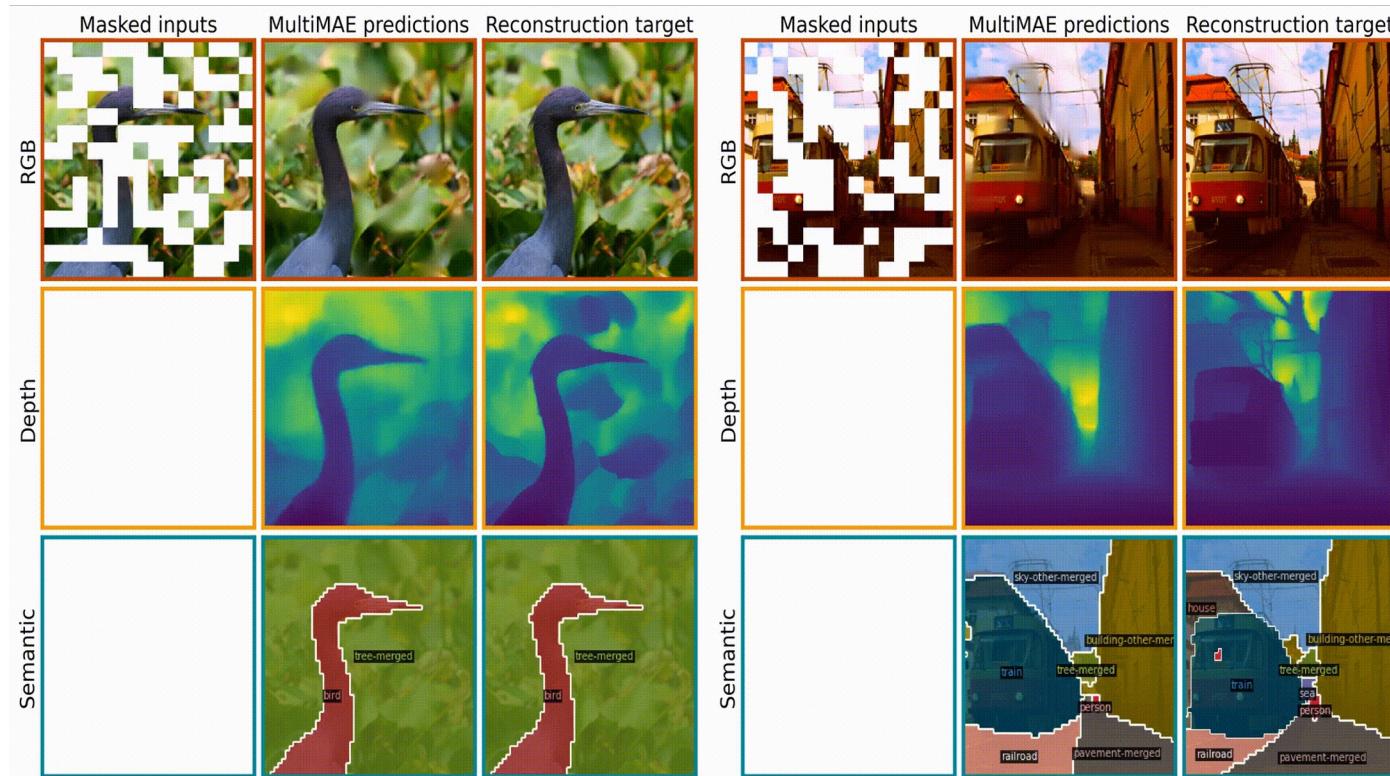
# Sequence-to-Sequence Transformer



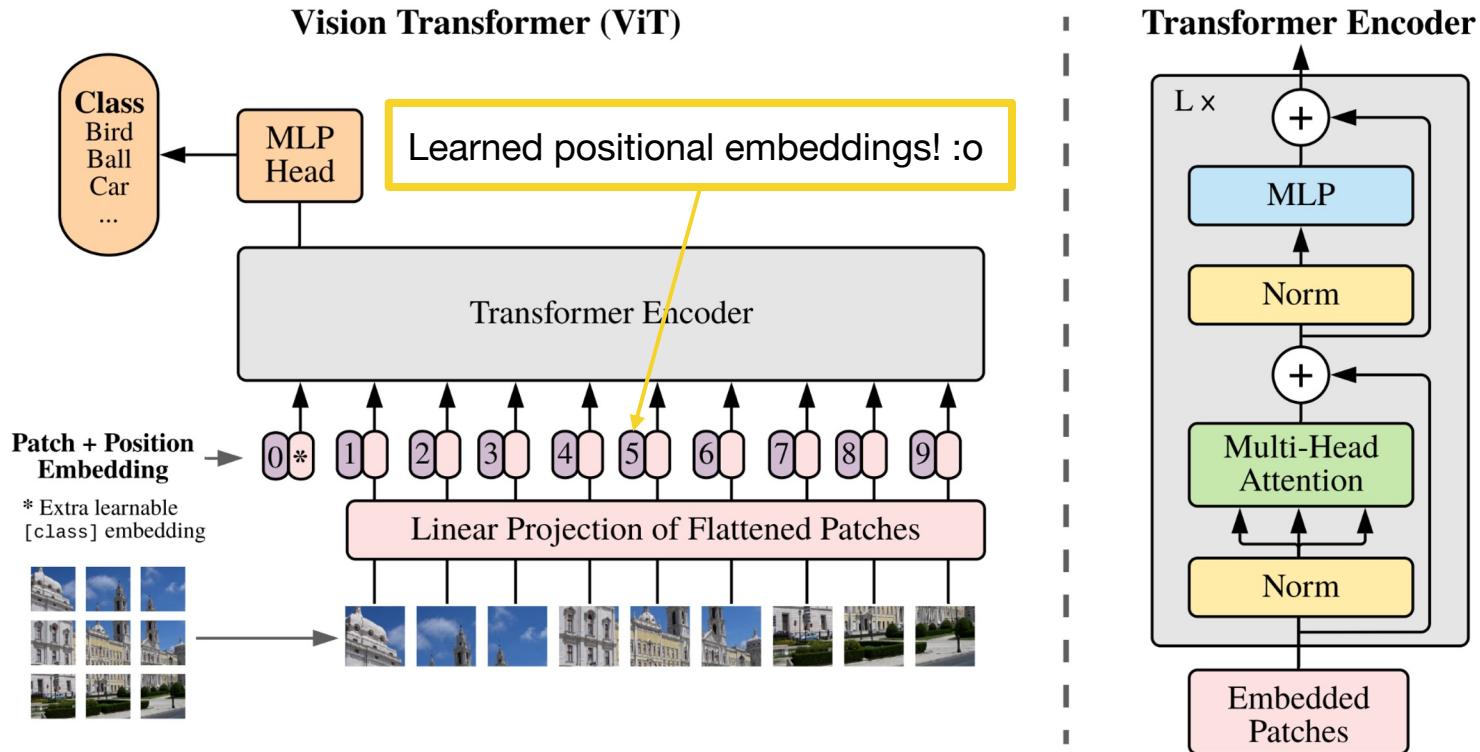
# Sequence-to-Sequence Transformer



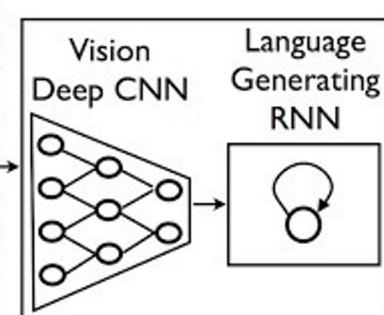
# Deep Learning for Vision: Multi-Task Processing



# Transformers for Vision: Patches



# Deep Learning for Language and Vision: Captioning



**A group of people shopping at an outdoor market.**

**There are many vegetables at the fruit stand.**

- Modern flashy systems don't do image-to-text, but instead...

# Deep Learning for Language and Vision: Text-to-Image

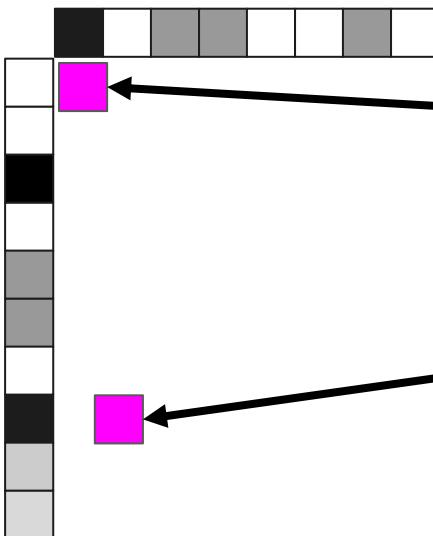


# Formulating the Retrieval Problem as a Linear Model

$$\sum_{i=1, j=1}^{n,m} a_i \theta_{i,j} b_j$$

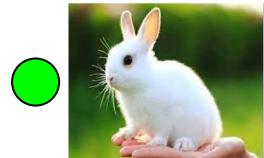


a	bird	hand	in	men	on	rabbit	the
2	0	1	1	0	0	1	0



Weight of (blue, "a")

Weight of (white, "bird")



“A rabbit sits in the palm of a hand.”



“A white fox is looking at the camera.”



“Children play soccer in a field.”



“Men are talking on a basketball court.”

Image  
Embedder

$$\psi(\mathcal{I})$$

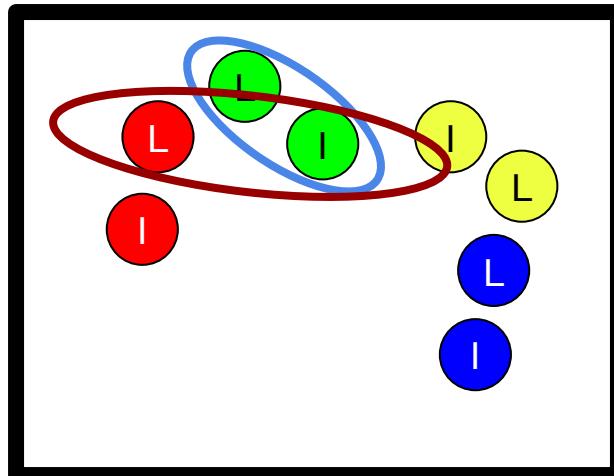
Caption  
Embedder

$$\omega(\mathcal{L})$$

Pull matching

image  
embeddings  
Learned  
Projection

together.

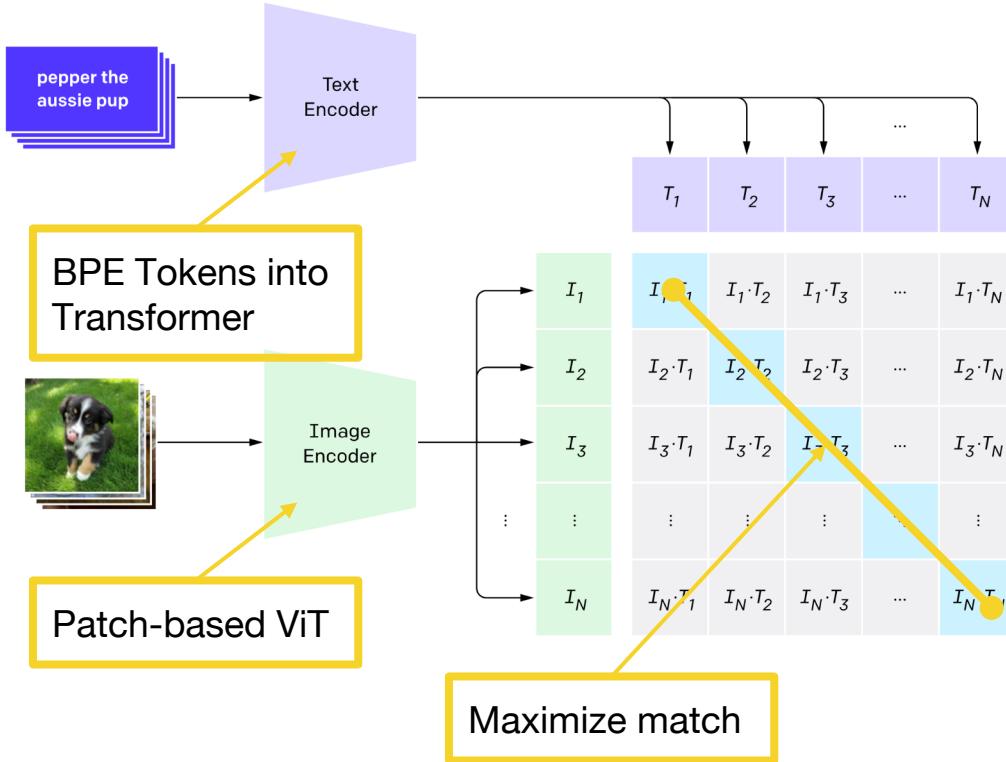


Push distractor

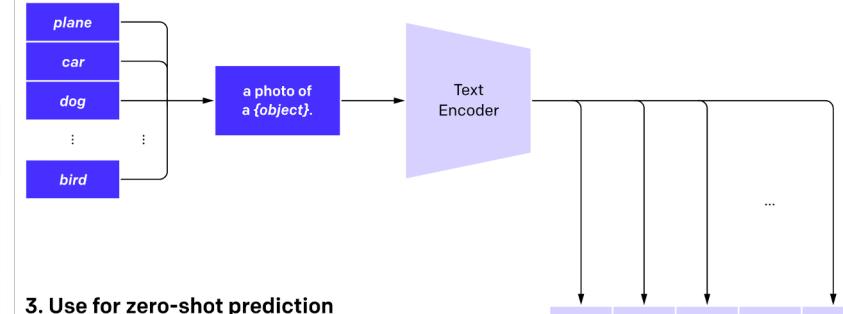
Learned  
Projection  
embeddings

apart.

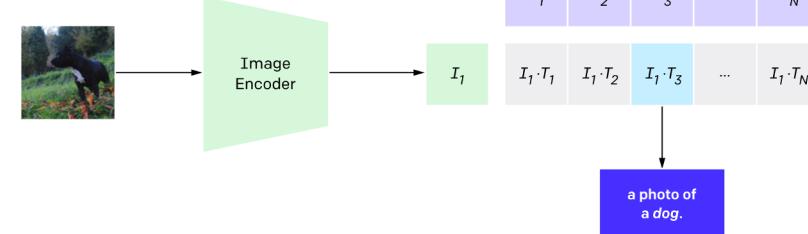
# Contrastive Language–Image Pre-training (CLIP)



## 2. Create dataset classifier from label text

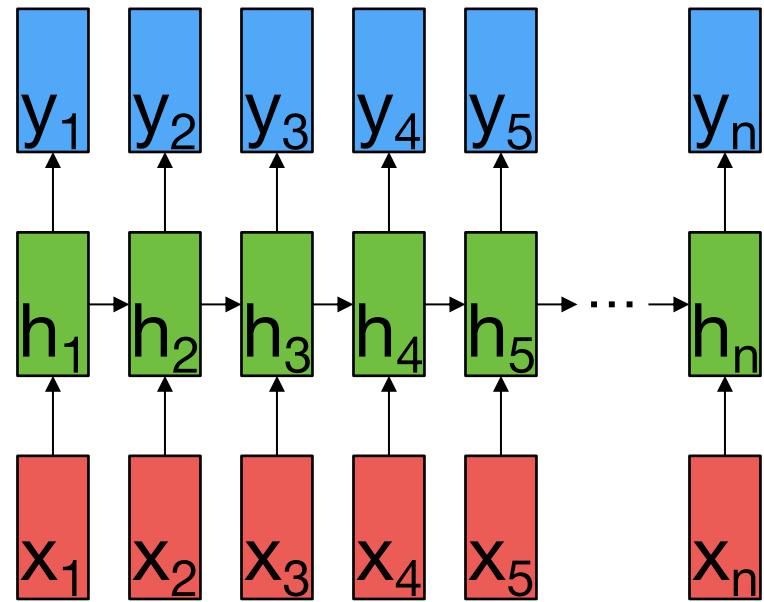
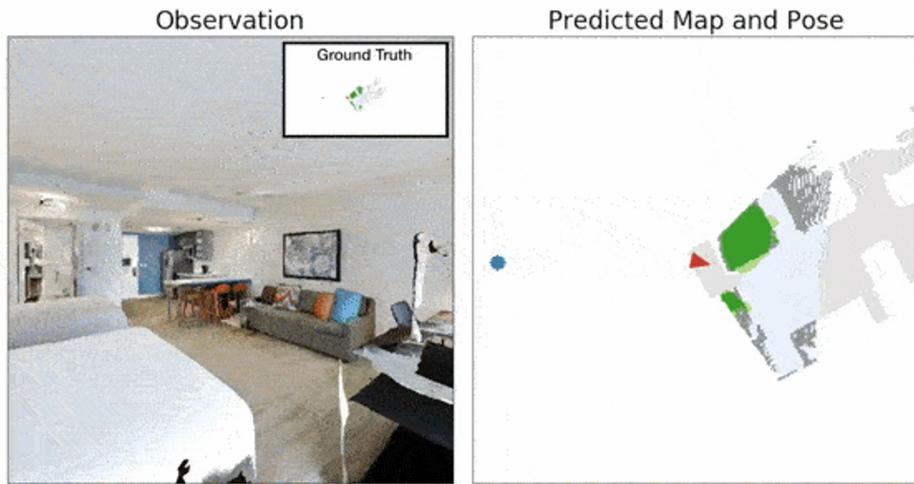


## 3. Use for zero-shot prediction

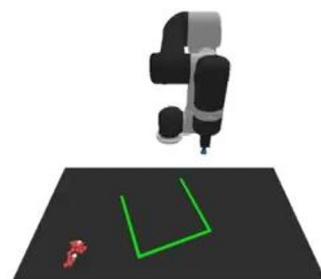


# Classes of Sequence-to-Sequence Problems

- Many-to-many
  - A sequence of inputs produces a sequence of outputs
    - Robot Actions



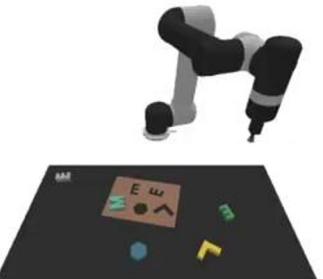
# Deep Learning for Language and Robots: Pick-and-Place



"align the rope from back right corner to back left corner"



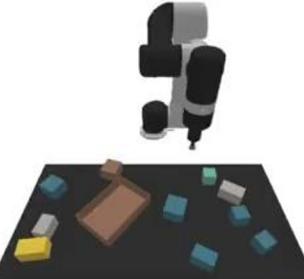
"pack the hexagon in the brown box"



"put the green letter E in the right letter E shape hole"



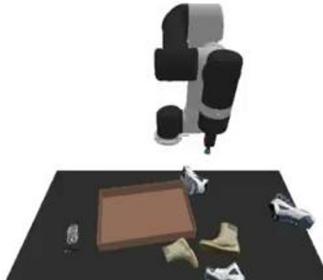
"put the blue blocks in a green bowl"



"pack all the yellow and blue blocks in the brown box"



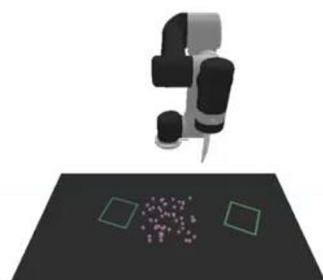
"pack the yoshi figure in the brown box"



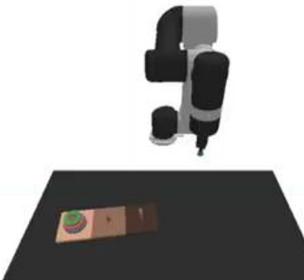
"pack all the blue and black sneaker objects in the brown box"



"put the blue block on the lightest brown block"



"push the pile of purple blocks into the green square"



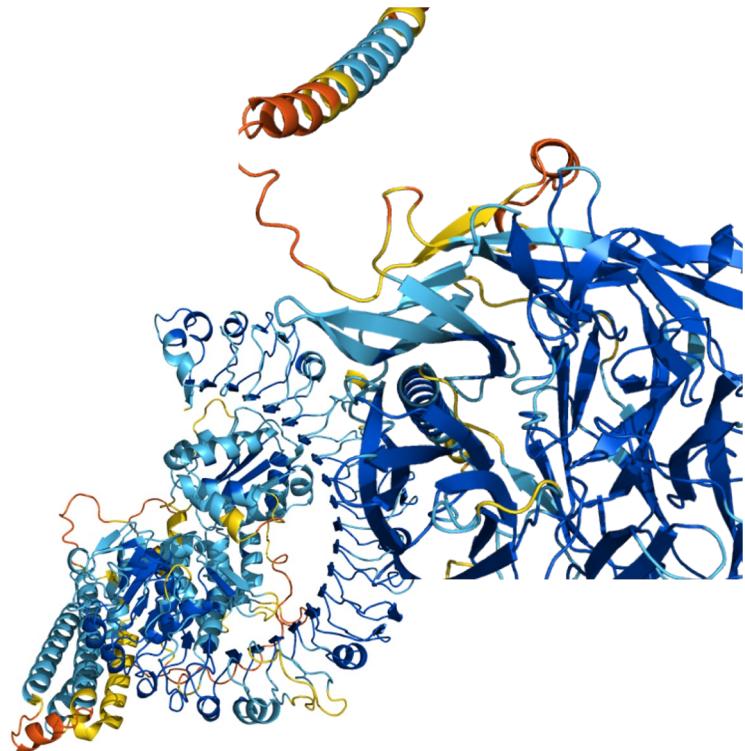
"move the green ring to the darker brown side"

# Some Things We Did Not Cover

- Graph neural networks (GNNs)
  - Learn *node embeddings* over which we run operations like *graph convolutions*
  - Luckily you can basically imagine this because you know the intuitions for embeddings (compression) and taking advantage of locality information (e.g., CNN layer)
- Fast fine-tuning and continual learning: Adapter networks, memory relation networks, in-context learning
- Diffusion models and the wider space of unsupervised learning based on noising objectives (e.g., masking/cloze objectives)

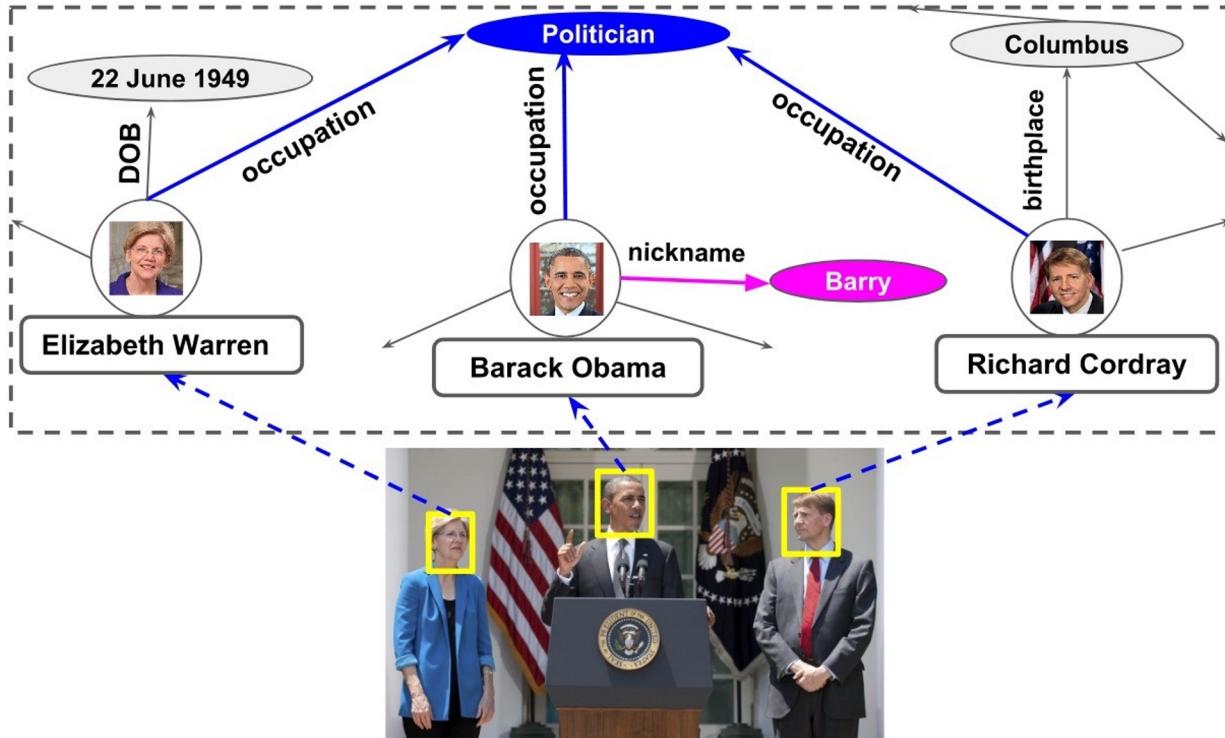
# Graph Neural Networks

AlphaFold is an AI system developed by DeepMind that predicts a protein's 3D structure from its amino acid sequence. It regularly achieves accuracy competitive with experiment.



# Graph Neural Networks

## Knowledge Graph



# Deep Learning and The World

**AI suggested 40,000 new possible chemical weapons in just six hours**



/ ‘For me, the concern was just how easy it was to do’

# Deep Learning and The World

Write a python function to check if -

ChatGPT  
privacy

① 1 April

d  
if  
re  
else:  
return False

logically or mathema

• **Take over**

Now, Stelladia's art has gone viral for all the wrong reasons. Over the last few weeks, AI image generators like Lensa and MyHeritage Time Machine, which have been trained using Stable Diffusion, have trended on [TikTok](#) and [Instagram](#).

ARTIFICIAL INTELLIGENCE / TECH / LAW

**Getty Images sues AI art generator Stable Diffusion in the US for copyright infringement**

Machine cost money to use, this means

off of these artists' own work. And the artists don't see a cent.

# Deep Learning and The World

The doctor asked the nurse to help him

ENGLISH - DETECTED

The doctor asked for the scalpel.

ENGLISH SPANISH FRENCH

ENGLISH SPANISH ARABIC

Translations are gender-specific. [Learn more](#)

La doctora pidió el bisturí. (feminine)

El doctor

ENGLISH - DETECTED

The doctor as procedure.

The doctor asked for the scalpel.

Facebook translates 'good morning' into 'attack them', leading to arrest

Palestinian man questioned by Israeli police after embarrassing mistranslation of caption under photo of him leaning against bulldozer

# Deep Learning and The World

Prompt: ceo;

Date: April 6, 2022



Prompt: nurse;

Date: April 6, 2022



Prompt: a photo of a personal assistant;  
Date: April 1, 2022



[Submitted on 20 Mar 2023]

## Stable Bias: Analyzing Societal Representations in Diffusion Models

Alexandra Sasha Luccioni, Christopher Akiki, Margaret Mitchell, Yacine Jernite



# Deep Learning and The World

LAPD could soon add Boston Dynamics' \$300K robot dog to its arsenal

- These  
get cool



NYPD

## NYPD Adds 'Spot,' the Robot Dog, to Force – and More Tech

Digidog is just one of the three new technologies that the NYPD is planning to use as part of its overall plan to use innovative devices to increase safety. Two of the new devices are pilot programs.

By NBC New York Staff • Published April 11, 2023 • Updated on April 11, 2023 at 11:38 pm



### MOST POPULAR

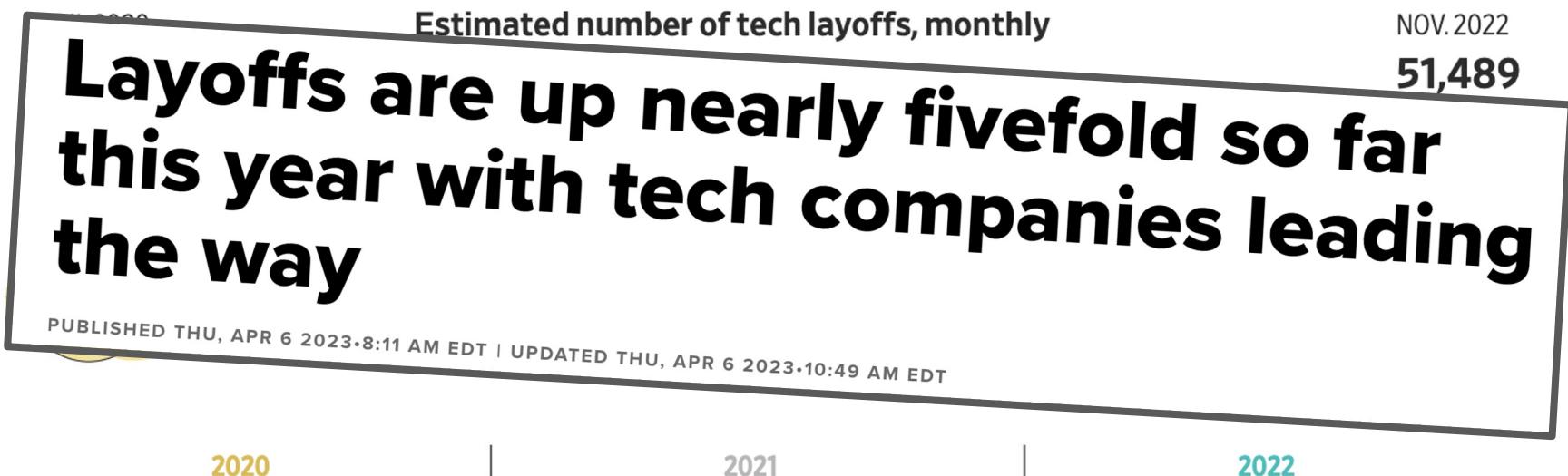
- 1 Southern California braces for heavy rain as 2023 continues its showery start
- 2 Southern California storm map: Track where the rain is now
- 3 Sepulveda Basin closes due to flooding

# A (Short) History of AI

- ...
- 1990–2012: Statistical approaches + subfield expertise; hand-engineered feature extraction
  - Resurgence of probability, focus on uncertainty
  - General increase in technical depth
  - Agents and learning systems... “AI Spring”?
- 2012–2019: Excitement: representation learning
  - Big data, big compute, neural networks
  - Some re-unification of subfields
  - AI used in many industries
- 2019–...: Overhype and reckless deployment
  - Massive compute; uncurated, Internet-scraped data at incredible scale
  - Black-box models behind API paywalls; closed source development
  - “Move fast and hurt people” philosophy seen across AI-enabled areas

# Overhype

- 2019—...: Overhype and reckless deployment
  - What's a concrete indicator of overhype?



# Deep Learning and Its Applications

- When I took my first computer science class in 2008, my instructor ended the course by saying:
  - “I have taught you enough to make you dangerous”
- That attitude has been common in the tech industry for decades, and is evidenced in mottos like:
  - “move fast and break things” - Mark Zuckerberg
- The AI toolbox is not a toy chest anymore, it’s an arsenal.
- I have taught you enough to make you dangerous.
- I hope I have also taught you enough to make you cautious.

## Action Items for You

- For those doing a Paper Roleplaying Breakout session today, turn in your 2-page paper role report by 11:59pm tonight through the gDrive submission form
- If your *final project presentation* is on April 21st (one week from today), turn in your project question + multiple choice answers by tonight; your submitted question is part of your final presentation grade
- Decide within your group how you want to divide the presentation; fine for one person to give, fine for all four to share, etc. Time limit will be strictly enforced

# Acknowledgements

- Thank you for your patience and aplomb; you've been great
- We started this course with:
  - a green instructor (me)
  - half the number of seats for enrollment
  - half the TA support
  - a crappy lecture hall with loud construction outside
- We end the course with final presentations from you across an exciting and versatile range of DL applications and topics
- It's possible this is the last offering of Deep Learning and Its Applications, and I'm grateful to have shared it with you

# Paper Roleplaying Breakouts: April 14 [[Schedule](#)]

- Emergence of Maps in the Memories of Blind Navigation Agents (**Jesse, outside THH 201**)
- Learning to Explore using Active Neural SLAM (**Virtual with Bingjie**)
- Attention Is All You Need (**Virtual with Deqing**)
- Playing Atari with Deep Reinforcement Learning (**Jesse, front of THH 201**)
- The Hardware Lottery (**Tejas, back of THH 201**)

# CSCI 566: Deep Learning and Its Applications

Jesse Thomason

Lecture 11: Deep Learning and The World