Machine Learning Department
School of Computer Science
Carnegie Mellon University

Special Topics:

Significance Testing
+ Societal Impacts of ML
+ ChatGPT

Matt Gormley
Lecture 28
Apr. 26, 2023

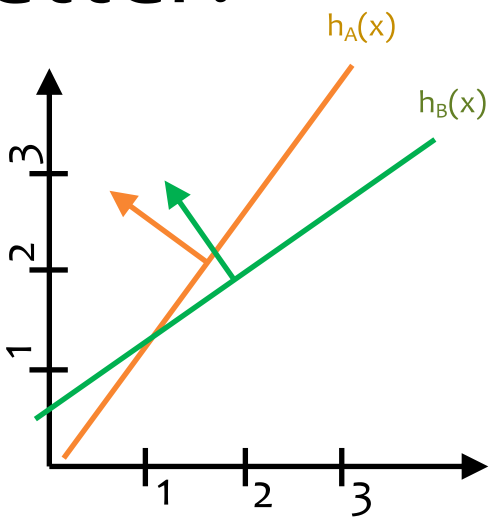
Reminders

- Homework 9: Learning Paradigms
  - Out: Fri, Apr. 21
  - Due: Thu, Apr. 27 at 11:59pm
    (only two grace/late days permitted)
- Exam 3 Practice Problems
  - Out: Tue, Apr 25
- Exam 3
  - Tue, May 2 (5:30pm – 7:30pm)
- Final Exit Poll (after Exam 3)

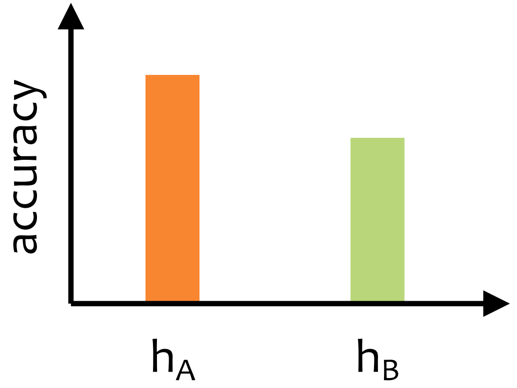
SIGNIFICANCE TESTING

Which classifier is better?

Goal: Given two classifiers: $h_A(x)$ and $h_B(x)$ which is better?



Common Approach: Evaluate each classifier on a test set and report which has higher accuracy.



Two Sources of Variance

1. Randomness in training
2. Randomness in our test data

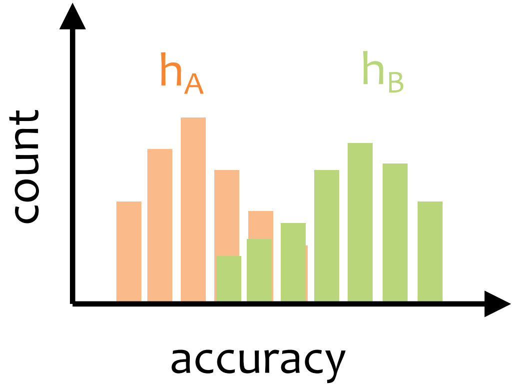
1. Randomness in training

Example: Assume we are training a **deep neural network** with a **nonconvex objective** function via **random restarts**

We collect a sequence of classifiers for R random restarts:

- $h_B(x)^{(1)} \longleftarrow$ train(D, seed = time in ms)
- $h_B(x)^{(2)} \longleftarrow$ train(D, seed = time in ms)
- …
- $h_B(x)^{(R)} \longleftarrow$ train(D, seed = time in ms)

Solution: histogram



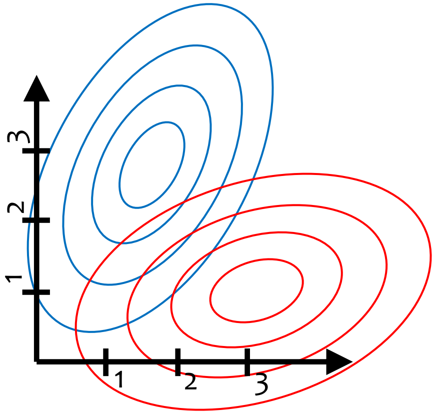
Solution: confidence interval

report variance of h_A and h_B

Ex:

- h_A 45% +/- 5%
- h_B 47% +/- 8%

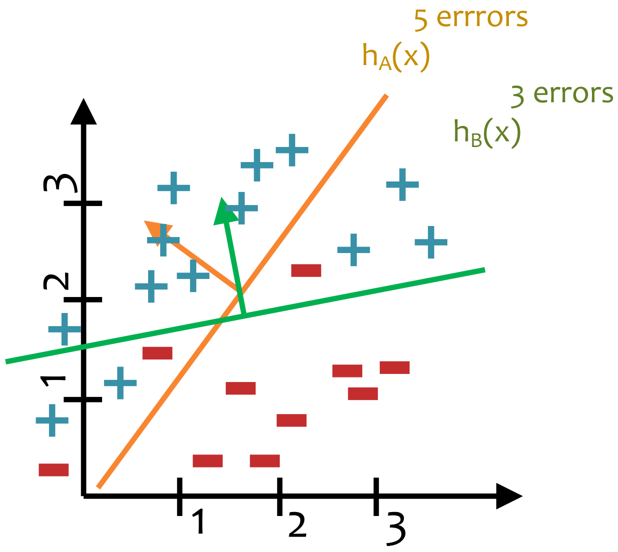
2. Randomness in our test data

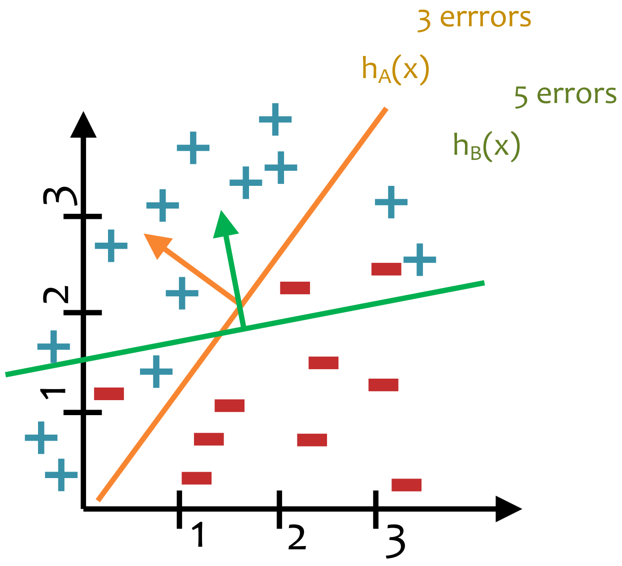


Recall: we assume $x^{(i)} \sim p^*(\cdot)$ and $y^{(i)} = c^*(\mathbf{x}^{(i)})$

or $(x^{(i)}, y^{(i)}) \sim p^*(\cdot, \cdot)$

Data: Assume the data is drawn from a generative distribution $p^*(x|y)p^*(y)$ where $p^*(y)$ is an even coin flip and $p^*(x|y{=}\text{red})$ is the red Gaussian and $p^*(x|y{=}\text{blue})$ is the blue Gaussian.





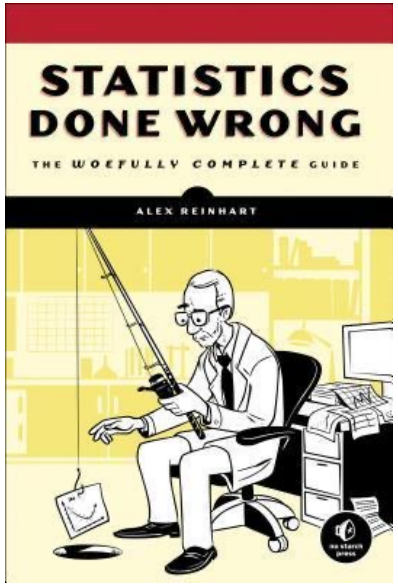
<u>Solution:</u>

significance testing

Significance Testing in ML

"And because any medication or intervention usually has some real effect, you can always get a statistically significant result by collecting so much data that you detect extremely tiny but relatively unimportant differences. As Bruce Thompson wrote, Statistical significance testing can involve a tautological logic in which tired researchers, having collected data on hundreds of subjects, then conduct a statistical test to evaluate whether there were a lot of subjects, which the researchers already know, because they collected the data and know they are tired. This tautology has created considerable damage as regards the cumulation of knowledge."

— Alex Reinhart
Statistics Done Wrong: The Woefully Complete Guide



For machine learning, significance testing is usually still answering an important question:

Did we evaluate our model on enough test data to conclude that our improvement over the baseline is surprising?

Significance Testing in ML

Paired Bootstrap Test

Key Idea: simulate the resampling of many test sets

Algorithm:

1. Draw B bootstrap samples
   $S^{(b)} = \{(\mathbf{x}^{(1)}, y^{(1)}) (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$
   with replacement from test data D_{test}
2. Let v = 0
3. For b = 1,…,B
   if $\delta(S^{(b)}) > 2\delta(D_{test})$:
   v = v + 1
4. Return p-value as v/B

$\delta(D')$ = difference in accuracy between h_A and h_B on D'

H0 = null hypothesis = performance of h_A and h_B is the same

FAIRNESS IN ML

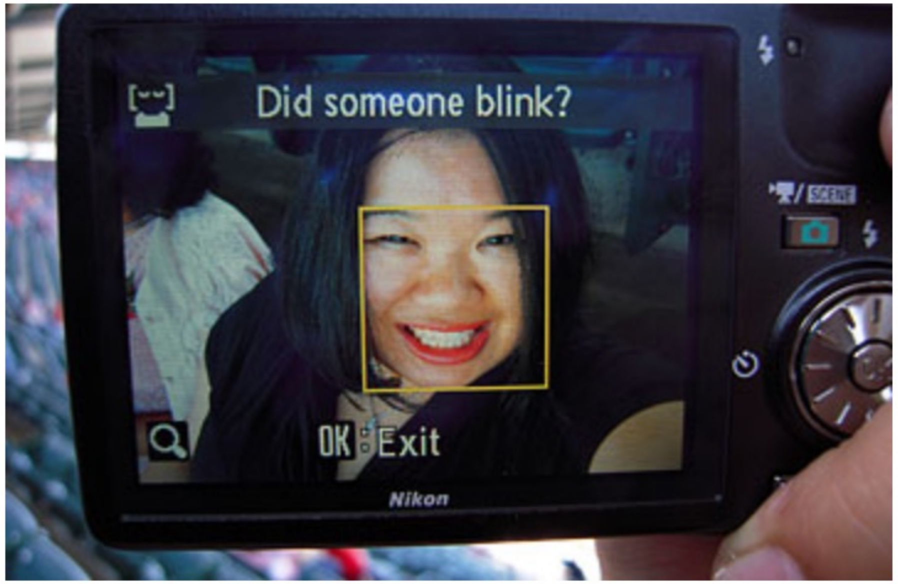
Are Face-Detection Cameras Racist?

By Adam Rose | Friday, Jan. 22, 2010

Tweet | Share | Read Later

When Joz Wang and her brother bought their mom a Nikon Coolpix S630 digital camera for Mother's Day last year, they discovered what seemed to be a malfunction. Every time they took a portrait of each other smiling, a message flashed across the screen asking, "Did someone blink?" No one had. "I thought the camera was broken!" Wang, 33, recalls. But when her brother posed with his eyes open so wide that he looked "bug-eyed," the messages stopped.

Wang, a Taiwanese-American strategy consultant who goes by the Web handle "jozjozjoz," thought it was funny that the camera had difficulties figuring out when her family had their eyes open. So she



Joz Wang

Source: http://content.time.com/time/business/article/0,8599,1954643,00.html

IS THE IPHONE X RACIST? APPLE REFUNDS DEVICE THAT CAN'T TELL CHINESE PEOPLE APART, WOMAN CLAIMS

BY CHRISTINA ZHAO ON 12/18/17 AT 12:24 PM EST

"A Chinese woman [surname Yan] was offered two refunds from Apple for her new iPhone X… [it] was unable to tell her and her other Chinese colleague apart."

"Thinking that a faulty camera was to blame, the store operator gave [Yan] a refund, which she used to purchase another iPhone X. But the new phone turned out to have the same problem, prompting the store worker to offer her another refund … It is unclear whether she purchased a third phone"

Source: https://www.newsweek.com/iphone-x-racist-apple-refunds-device-cant-tell-chinese-people-apart-woman-751263

"As facial recognition systems become more common, Amazon has emerged as a frontrunner in the field, courting customers around the US, including police departments and Immigration and Customs Enforcement (ICE)."

Gender and racial bias found in Amazon's facial recognition technology (again)

Research shows that Amazon's tech has a harder time identifying gender in darker-skinned and female faces

By James Vincent | Jan 25, 2019, 9:45am EST

Source: https://www.theverge.com/2019/1/25/18197137/amazon-rekognition-facial-recognition-bias-race-gender

Healthcare risk algorithm had 'significant racial bias'

It reportedly underestimated health needs for black patients.

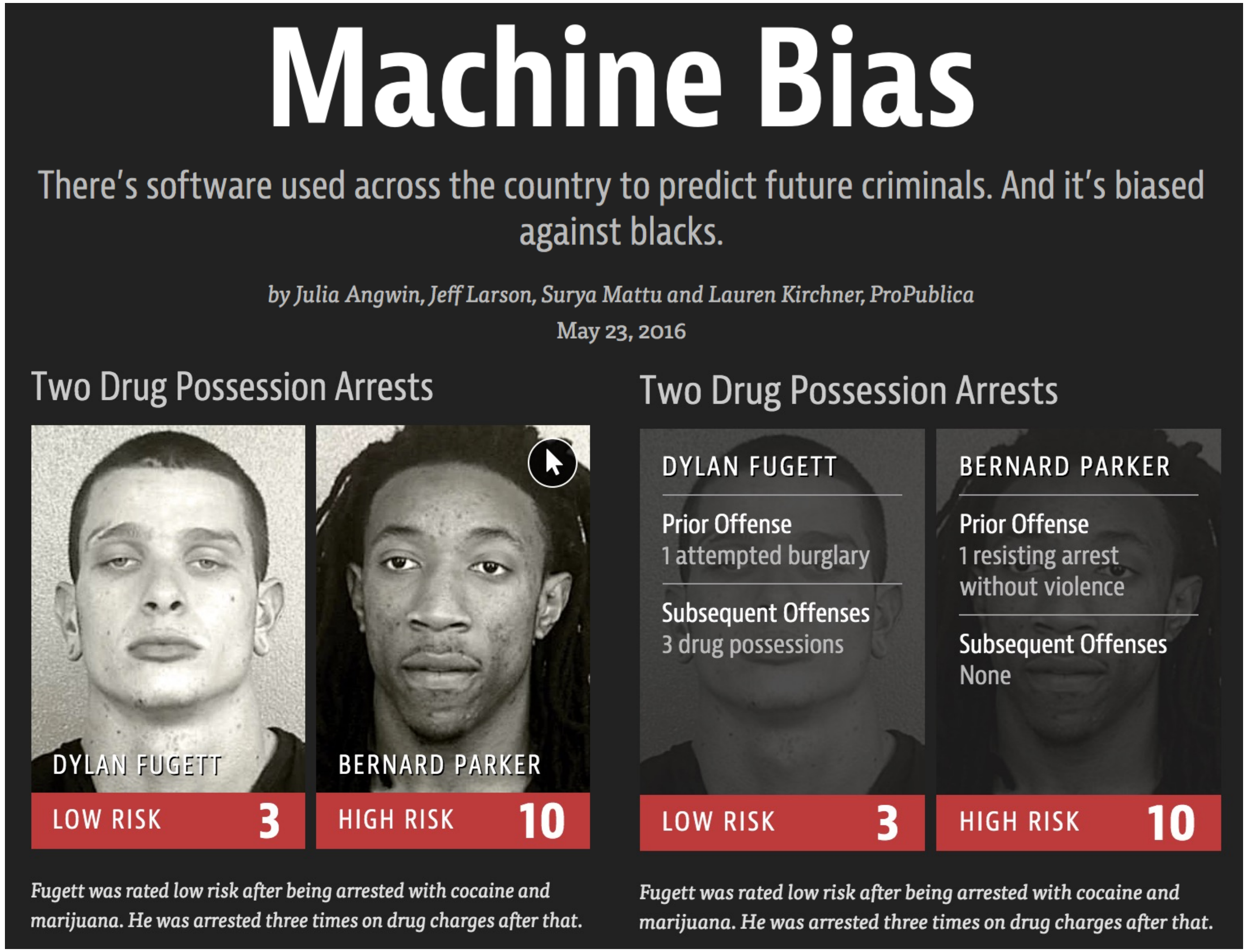
Jon Fingas, @jonfingas
10.26.19 in Medicine

"While it [the algorithm] didn't directly consider ethnicity, its emphasis on medical costs as bellwethers for health led to the code routinely underestimating the needs of black patients. A sicker black person would receive the same risk score as a healthier white person simply because of how much they could spend."

Source: https://science.sciencemag.org/content/366/6464/447

Word embeddings and analogies

- https://lamyiowce.github.io/word2viz/



Source: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

Different Types of Errors

	True label	Predicted label
True positive (TP)	$+1$	$+1$
False positive (FP)	-1	$+1$
True negative (TN)	-1	-1
False negative (FN)	$+1$	-1

How We Analyzed the COMPAS Recidivism Algorithm

by Jeff Larson, Surya Mattu, Lauren Kirchner and Julia Angwin

May 23, 2016

All Defendants	Low	High	Black Defendants	Low	High	White Defendants	Low	High
Survived	2681	1282	Survived	990	805	Survived	1139	349
Recidivated	1216	2035	Recidivated	532	1369	Recidivated	461	505
FP rate: 32.35			FP rate: 44.85			FP rate: 23.45		
FN rate: 37.40			FN rate: 27.99			FN rate: 47.72		

This is one possible definition of unfairness.

We'll explore a few others and see how they relate to one another.

Source: https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm

Running Example

- Suppose you're an admissions officer for CMU, deciding which applicants to admit to your program
- $\boldsymbol{x}$ are the features of an applicant (e.g., standardized test scores, GPA)
- a is a protected attribute (e.g., gender), usually categorical i.e. $a \in \{a_1, \dots, a_C\}$
- $h(\boldsymbol{x}, a)$ is your model's prediction, which usually corresponds to some decision or action (e.g., $+1 =$ admit to CMU)
- y is the true, underlying target variable, usually thought of as some latent or hidden state (e.g., $+1 =$ this applicant would be "successful" at CMU)

Three Criteria for Fairness

- **Independence**: $h(\boldsymbol{x}, a) \perp a$
  - Probability of being accepted is the same for all genders
- **Separation**: $h(\boldsymbol{x}, a) \perp a \mid y$
  - All "good" applicants are accepted with the same probability, regardless of gender
  - Same for all "bad" applicants
- **Sufficiency**: $y \perp a \mid h(\boldsymbol{x}, a)$
  - For the purposes of predicting y, the information contained in $h(\boldsymbol{x}, a)$ is "sufficient", a becomes irrelevant

Achieving Fairness

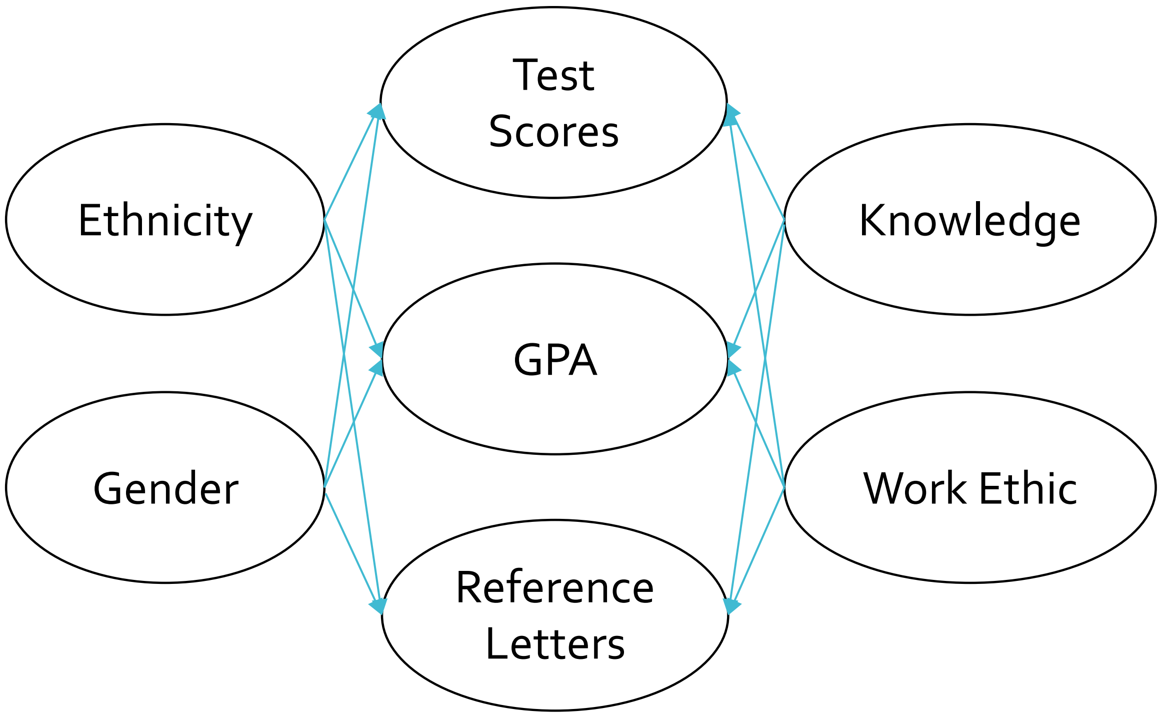
- Pre-processing data
- Additional constraints during training
- Post-processing predictions

Three Criteria for Fairness

- **Independence**: $h(\boldsymbol{x}, a) \perp a$
  - Probability of being accepted is the same for all genders
- **Separation**: $h(\boldsymbol{x}, a) \perp a \mid y$
  - All "good" applicants are accepted with the same probability, regardless of gender
  - Same for all "bad" applicants
- **Sufficiency**: $y \perp a \mid h(\boldsymbol{x}, a)$
  - For the purposes of predicting y, the information contained in $h(\boldsymbol{x}, a)$ is "sufficient", a becomes irrelevant
- Any two of these criteria are mutually exclusive in the general case!

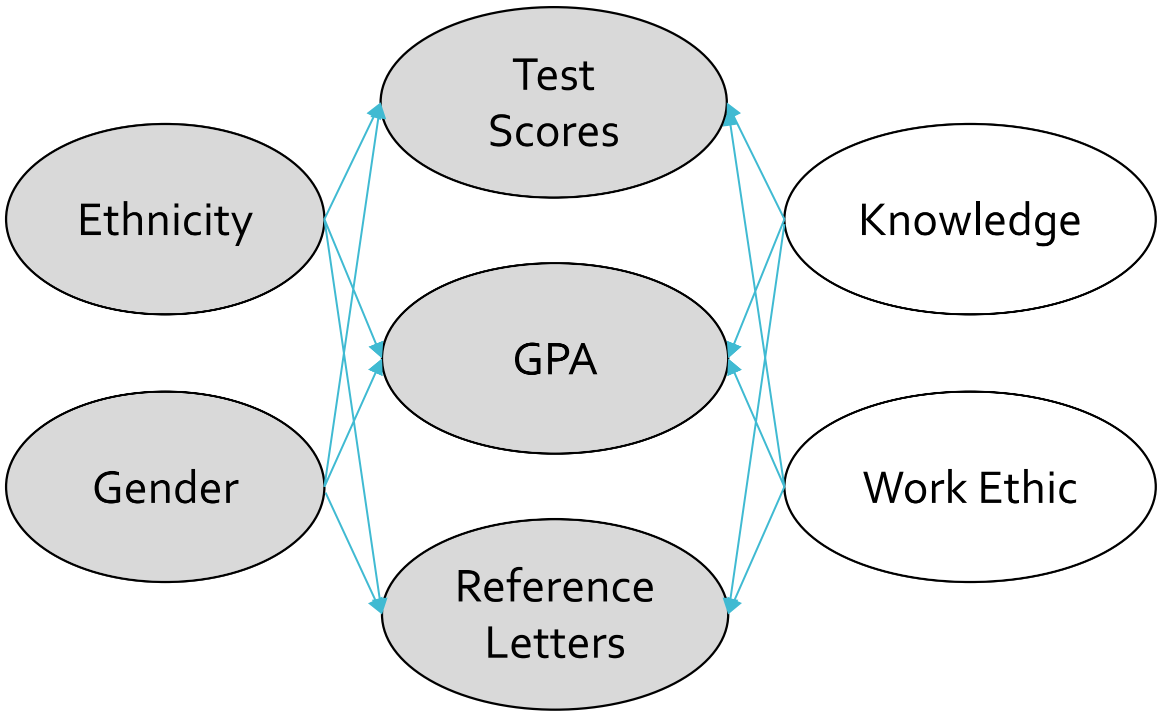
A Fourth Criterion for Fairness

- ~~Causality~~ Bayesian networks to the rescue!



A Fourth Criterion for Fairness

- ~~Causality~~ Bayesian networks to the rescue!



- Counterfactual fairness: how would an applicant's probability of acceptance change if they were a different gender?

Source: Counterfactual fairness, Kusner et al., https://papers.nips.cc/paper/2017/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf

LARGE LANGUAGE MODELS

What is ChatGPT?

- ChatGPT is a large (in the sense of having many parameters) language model, fine-tuned to be a dialogue agent
- The base language model is GPT-3.5 which was trained on a large quantity of text

Outline

- Task: Language Modeling
  - noisy channel models (speech / MT)
  - (historical) Large LMs (n-gram models)
- Model: GPT
  - Attention (computation graph)
  - Transformer-LM (cf. RNN-LM)
- Learning
  - Pre-training (unsupervised learning)
  - Reinforcement Learning with Human Feedback (deep RL)
- Optimization
  - AdamW (cf. SGD)
  - Distributed training
- Societal Impacts

TASK: LANGUAGE MODELING

Recall…

n-Gram Language Model

Question: How can we **define** a probability distribution over a sequence of length T?

The	bat	made	noise	at	night
w_1	w_2	w_3	w_4	w_5	w_6

n-Gram Model (n=3)

$$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, w_{t-2})$$

$p(w_1, w_2, w_3, \ldots, w_6) =$

The						$p(w_1)$
The	bat					$p(w_2 \mid w_1)$
The	bat	made				$p(w_3 \mid w_2, w_1)$
	bat	made	noise			$p(w_4 \mid w_3, w_2)$
		made	noise	at		$p(w_5 \mid w_4, w_3)$
			noise	at	night	$p(w_6 \mid w_5, w_4)$

Recall...

n-Gram Language Model

Question: How can we **define** a probability distribution over a sequence of length T?

The	bat	made	noise	at	night
w_1	w_2	w_3	w_4	w_5	w_6

n-Gram Model (n=3)

$$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, w_{t-2})$$

$p(w_1, w \ldots w_3, \ldots, w_6) =$

The

The

The

$p(w_1)$

Note: This is called a **model** because we made some **assumptions** about how many previous words to condition on (i.e. only n-1 words)

Learning an n-Gram Model

Recall…

Question: How do we **learn** the probabilities for the n-Gram Model?

$p(w_t \mid w_{t-2} = \text{The}, w_{t-1} = \text{bat})$

w_t	$p(\cdot \mid \cdot, \cdot)$
ate	0.015
…	
flies	0.046
…	
zebra	0.000

$p(w_t \mid w_{t-2} = \text{made}, w_{t-1} = \text{noise})$

w_t	$p(\cdot \mid \cdot, \cdot)$
at	0.020
…	
pollution	0.030
…	
zebra	0.000

$p(w_t \mid w_{t-2} = \text{cows}, w_{t-1} = \text{eat})$

w_t	$p(\cdot \mid \cdot, \cdot)$
corn	0.420
…	
grass	0.510
…	
zebra	0.000

Recall...

Learning an n-Gram Model

<u>Question</u>: How do we **learn** the probabilities for the n-Gram Model?
<u>Answer</u>: From data! Just **count** n-gram frequencies

$p(w_t \mid w_{t-2} = \text{cows}, w_{t-1} = \text{eat})$

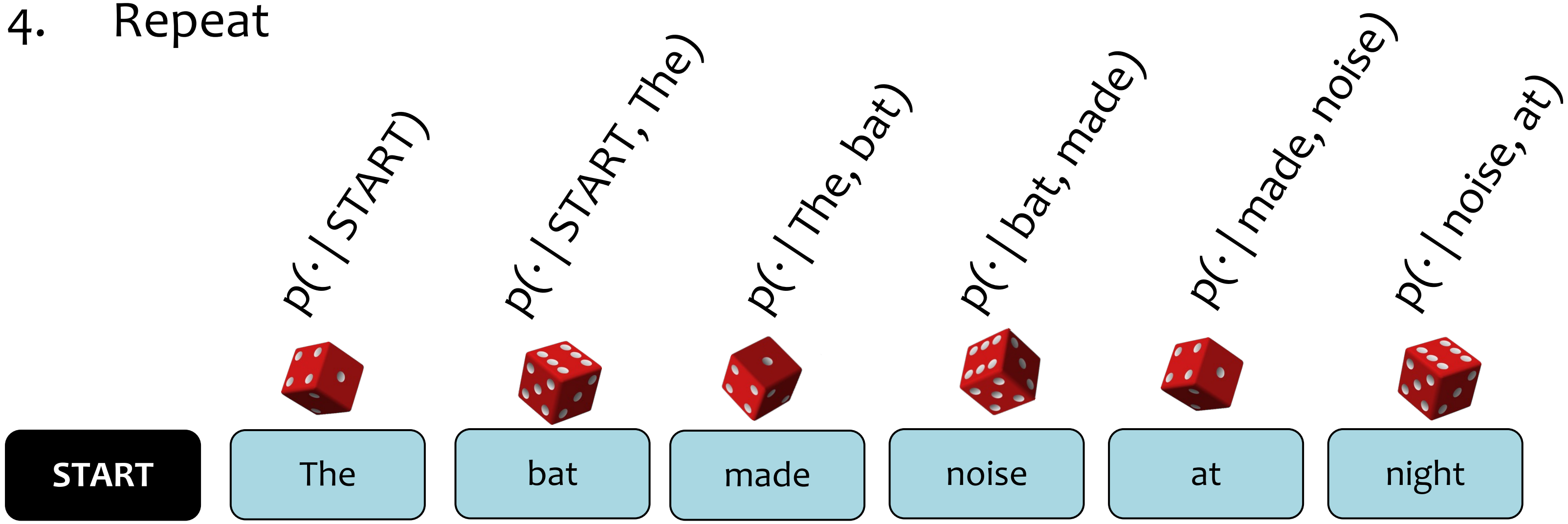
…the **cows eat** grass…
…our **cows eat hay** daily…
…factory-farm **cows eat** corn…
…on an organic farm, **cows eat hay** and…
…do your **cows eat** grass or corn?...
…what do **cows eat if** they have…
…**cows eat** corn when there is no…
…which **cows eat which** foods depends…
…if **cows eat** grass…
…when **cows eat** corn their stomachs…
…should we let **cows eat** corn?...

w_t	$p(\cdot \mid \cdot, \cdot)$
corn	4/11
grass	3/11
hay	2/11
if	1/11
which	1/11

Sampling from a Language Model

Recall…

Question: How do we sample from a Language Model?
Answer:
1. Treat each probability distribution like a (50k-sided) weighted die
2. Pick the die corresponding to $p(w_t \mid w_{t-2}, w_{t-1})$
3. Roll that die and generate whichever word w_t lands face up
4. Repeat



Noisy Channel Models

- Prior to 2017, two tasks relied heavily on language models:
  - speech recognition
  - machine translation
- Definition: a **noisy channel model** combines a *transduction model* (probability of converting **y** to **x**) with a *language model* (probability of **y**)

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}\, p(\mathbf{y} \mid \mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmax}}\, \underbrace{p(\mathbf{x} \mid \mathbf{y})}_{\text{transduction model}} \underbrace{p(\mathbf{y})}_{\text{language model}}$$

- **Goal:** to recover **y** from **x**
  - *For speech:* **x** is acoustic signal, **y** is transcription
  - *For machine translation:* **x** is sentence in source language, **y** is sentence in target language

Large (n-Gram) Language Models

- The earliest (truly) large language models were n-gram models
- Google n-Grams:
  - 2006: first release, English n-grams
    - trained on **1 trillion tokens** of web text (95 billion sentences)
    - included 1-grams, 2-grams, 3-grams, 4-grams, and 5-grams
  - 2009 – 2010: n-grams in Japanese, Chinese, Swedish, Spanish, Romanian, Portuguese, Polish, Dutch, Italian, French, German, Czech

English n-gram model is ~**3 billion parameters**

Number of unigrams:	13,588,391
Number of bigrams:	314,843,401
Number of trigrams:	977,069,902
Number of fourgrams:	1,313,818,354
Number of fivegrams:	1,176,470,663

```
serve as the incoming 92
serve as the incubator 99
serve as the independent 794
serve as the index 223
serve as the indication 72
serve as the indicator 120
serve as the indicators 45
serve as the indispensable 111
serve as the indispensible 40
serve as the individual 234
serve as the industrial 52
serve as the industry 607
```

```
accessoire Accessoires </S> 515
accessoire Accord i-CTDi 65
accessoire Accra accu 312
accessoire Acheter cet 1402
accessoire Ajouter au 160
accessoire Amour Beauté 112
accessoire Annuaire LOEIL 49
accessoire Architecture artiste 531
accessoire Attention : 44
```

```
惯例 为 电影 创作 52
惯例 为 的 是 95
惯例 为 目标 职位 49
惯例 为 确保 合作 69
惯例 为 确保 重组 213
惯例 为 科研 和 55
惯例 为 统称 </s> 183
惯例 为 维 和 50
惯例 为 自己 的 43
惯例 为 艺术类 学院 44
惯例 为 避免 侵权 148
```

Large (n-Gram) Language Models

- The earliest (truly) large language models were n-gram models
- Google n-Grams:
  - 2006: first release, English n-grams
    - trained on **1 trillion tokens** of web text (95 billion sentences)
    - included 1-grams, 2-grams, 3-grams, 4-grams, and 5-grams
  - 2009 – 2010: n-grams in Japanese, Chinese, Swedish, Spanish, Romanian, Portuguese, Polish, Dutch, Italian, French, German, Czech

English n-gram model is ~**3 billion parameters**

Number of unigrams:	13,588,391
Number of bigrams:	314,843,401
Number of trigrams:	977,069,902
Number of fourgrams:	1,313,818,354
Number of fivegrams:	1,176,470,663

Q: Is this a large training set?

A: Yes!

oire Accessoir
oire Accord i-
oire Accra acc
oire Acheter c
oire Ajouter a
oire Amour Bea
oire Annuaire
oire Architect
oire Attention

Q: Is this a large model?

A: Yes!

serve as the individual 234
serve as the industrial 52
serve as the industry 607

惯例 为 艺术类 学院 44
惯例 为 避免 侵权 148

How large are LLMs?

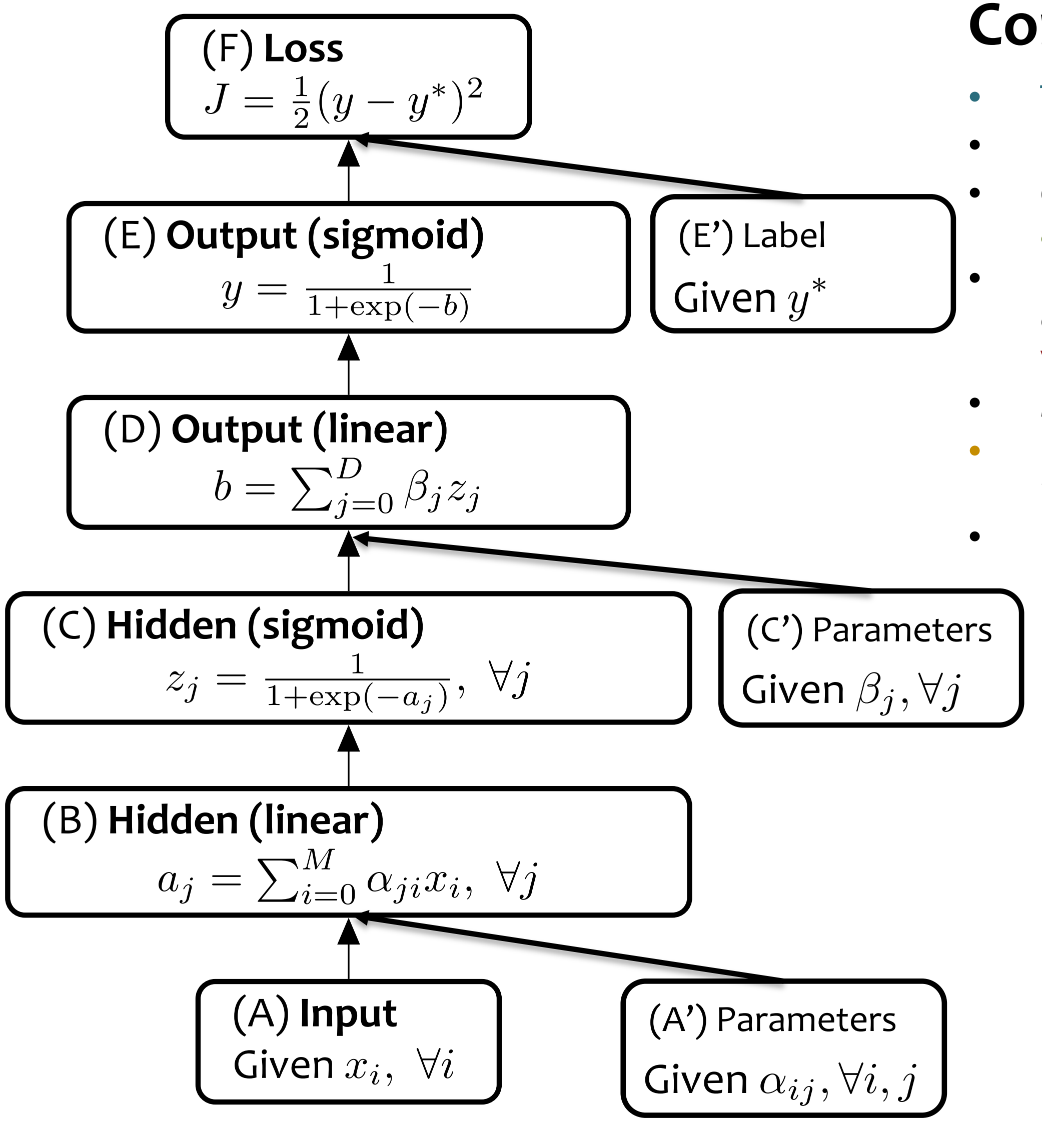
Comparison of some recent **large language models** (LLMs)

Model	Creators	Year of release	Training Data (# tokens)	Model Size (# parameters)
GPT-2	OpenAI	2019	~10 billion (40Gb)	1.5 billion
GPT-3 (cf. ChatGPT)	OpenAI	2020	300 billion	175 billion
PaLM	Google	2022	780 billion	540 billion
Chinchilla	DeepMind	2022	1.4 trillion	70 billion
LaMDA (cf. Bard)	Google	2022	1.56 trillion	137 billion
LLaMA	Meta	2023	1.4 trillion	65 billion
GPT-4	OpenAI	2023	?	?

MODEL: GPT

Ways of Drawing Neural Networks

Recall...

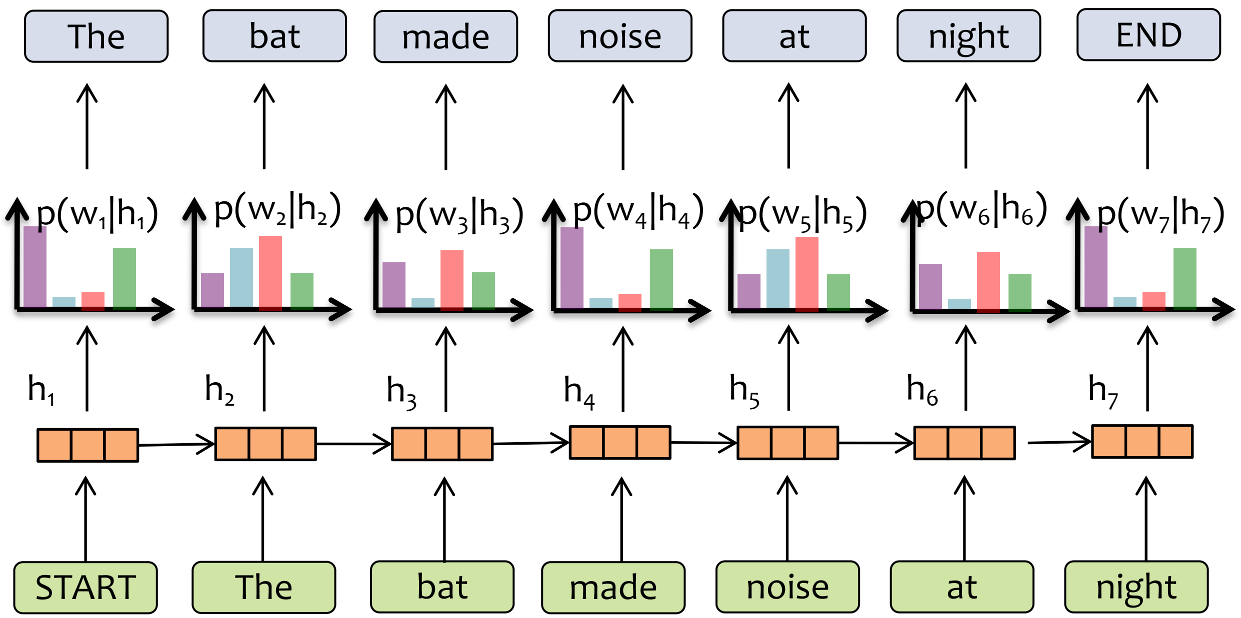


Computation Graph

- The diagram represents an algorithm
- Nodes are **rectangles**
- One node per intermediate variable in the algorithm
- Node is labeled with the function that it computes (inside the box) and also the variable name (outside the box)
- *Edges are directed*
- Edges do not have labels (since they don't need them)
- For neural networks:
  - Each intercept term should appear as a node (if it's not folded in somewhere)
  - Each parameter should appear as a node
  - Each constant, e.g. a true label or a feature vector should appear in the graph
  - It's perfectly fine to include the loss

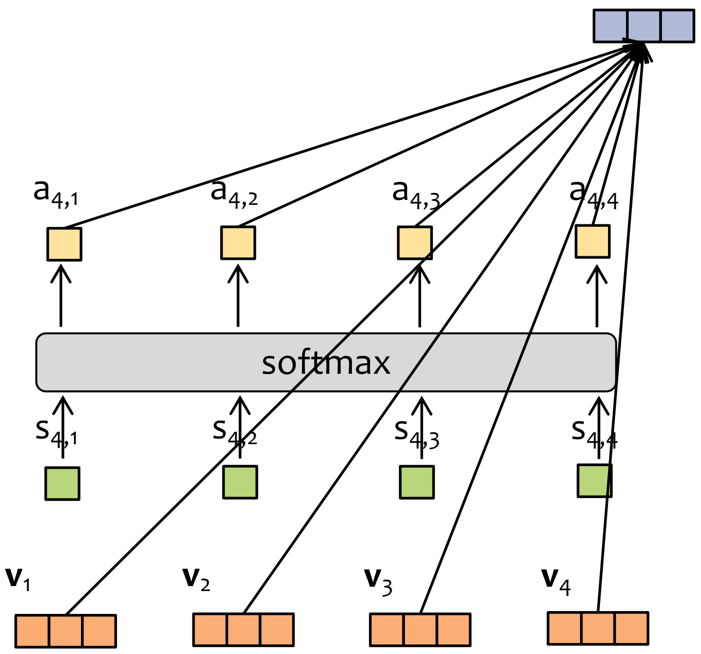
RNN Language Model

Recall...



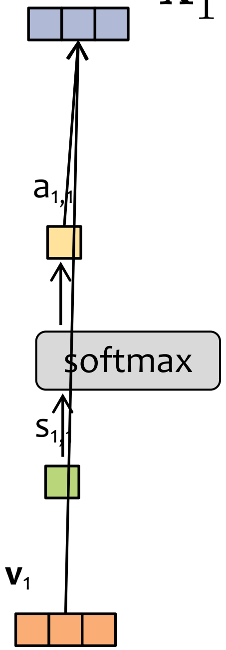
<u>*Key Idea*</u>:
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \dots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \dots, w_1)$

Attention



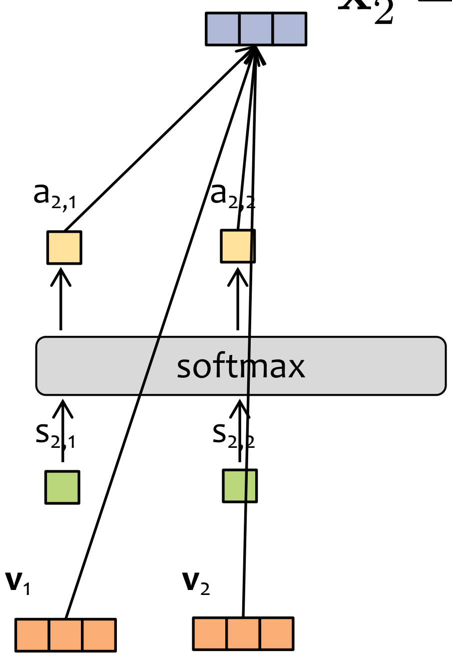
$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

Attention



$$\mathbf{x}'_1 = \sum_{j=1}^{1} a_{1,j}\mathbf{v}_j$$

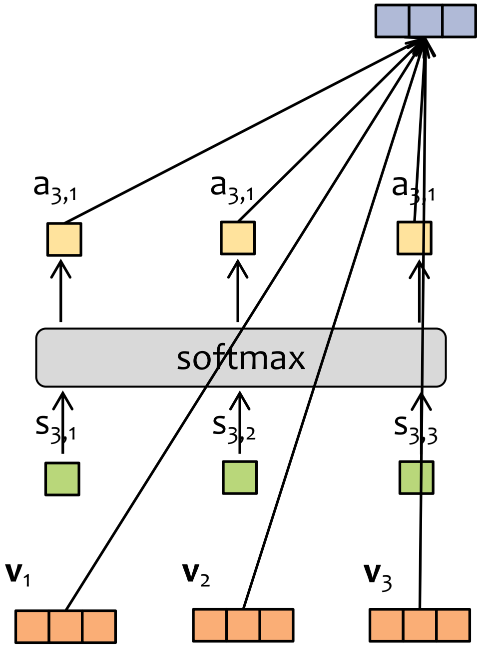
Attention



$$\mathbf{x}'_2 = \sum_{j=1}^{2} a_{2,j}\mathbf{v}_j$$

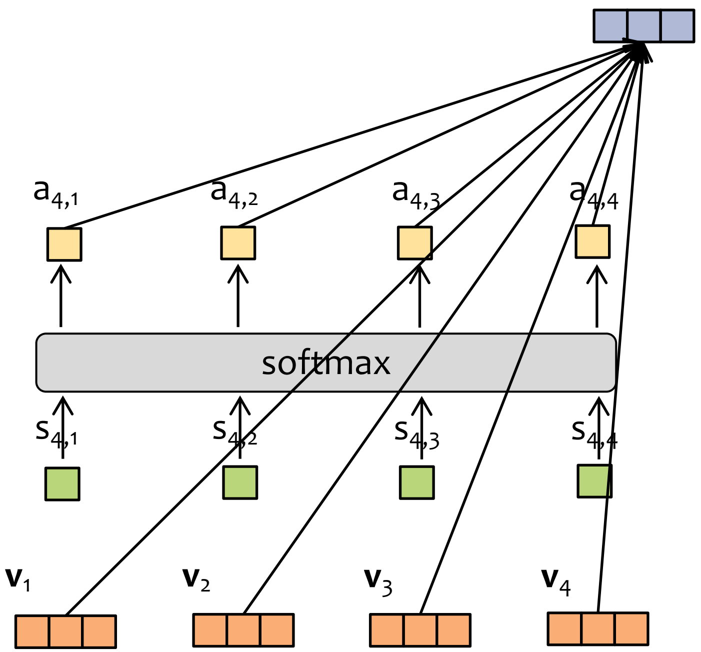
Attention

$$\mathbf{x}'_3 = \sum_{j=1}^{3} a_{3,j}\mathbf{v}_j$$

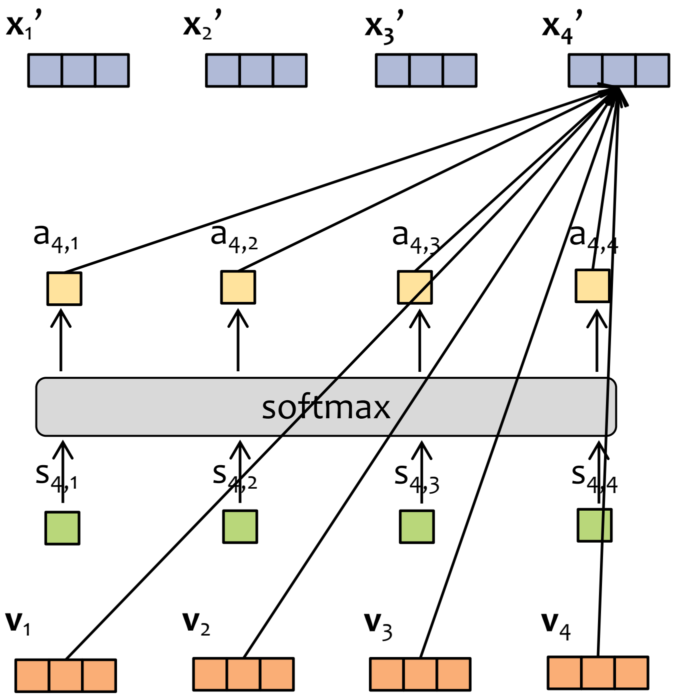


Attention

$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$



Attention



$$\mathbf{x}'_t = \sum_{j=1}^{t} a_{t,j} \mathbf{v}_j$$

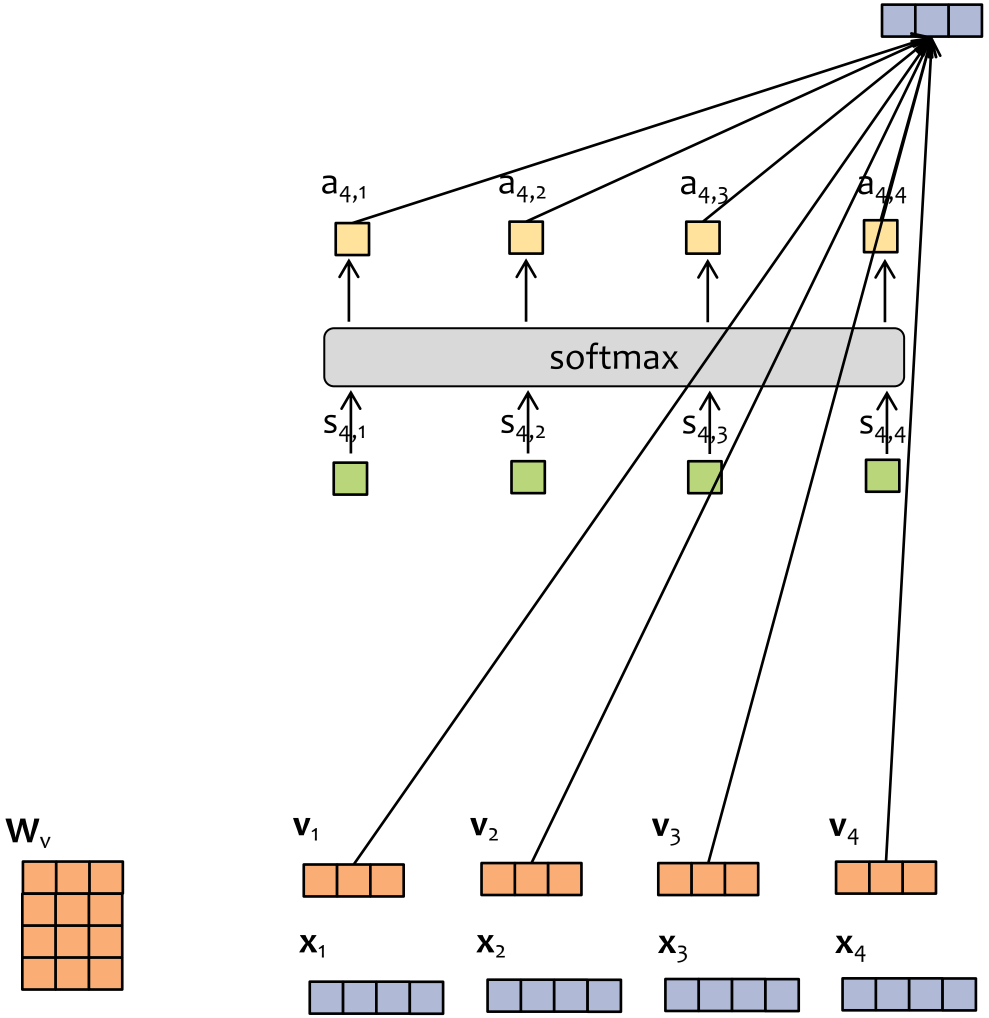
attention weights

scores

values

Scaled Dot-Product Attention

$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

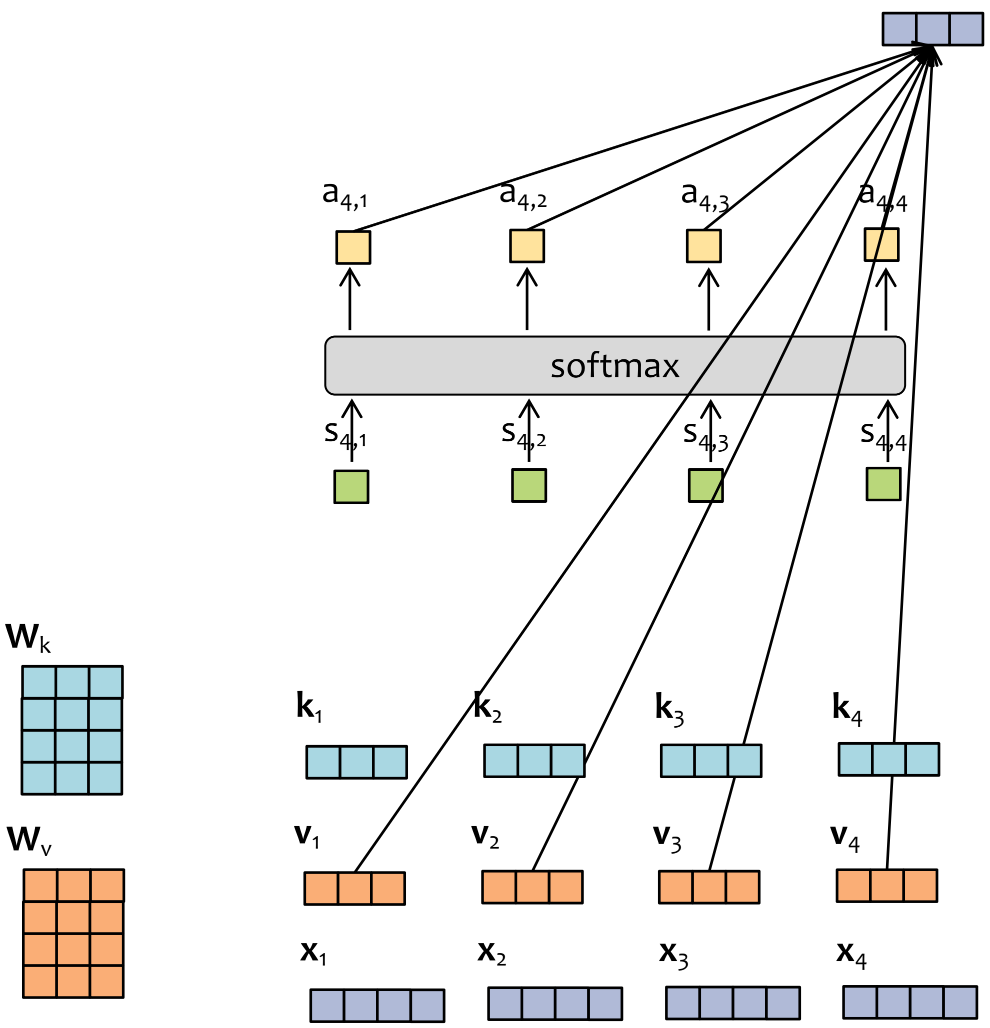


$$\mathbf{v}_j = \mathbf{W}_v^T\mathbf{x}_j$$

values

Scaled Dot-Product Attention

$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

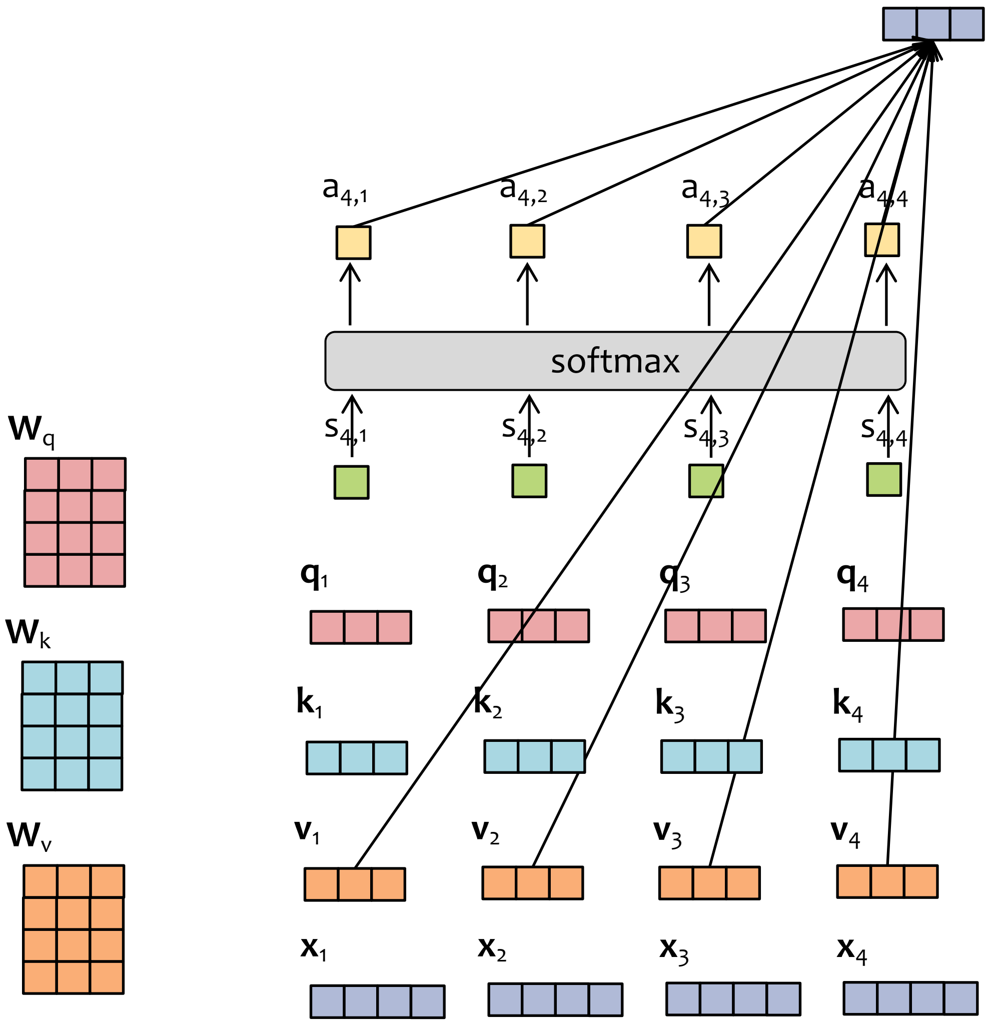


$$\mathbf{k}_j = \mathbf{W}_k^T\mathbf{x}_j$$ keys

$$\mathbf{v}_j = \mathbf{W}_v^T\mathbf{x}_j$$ values

Scaled Dot-Product Attention

$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

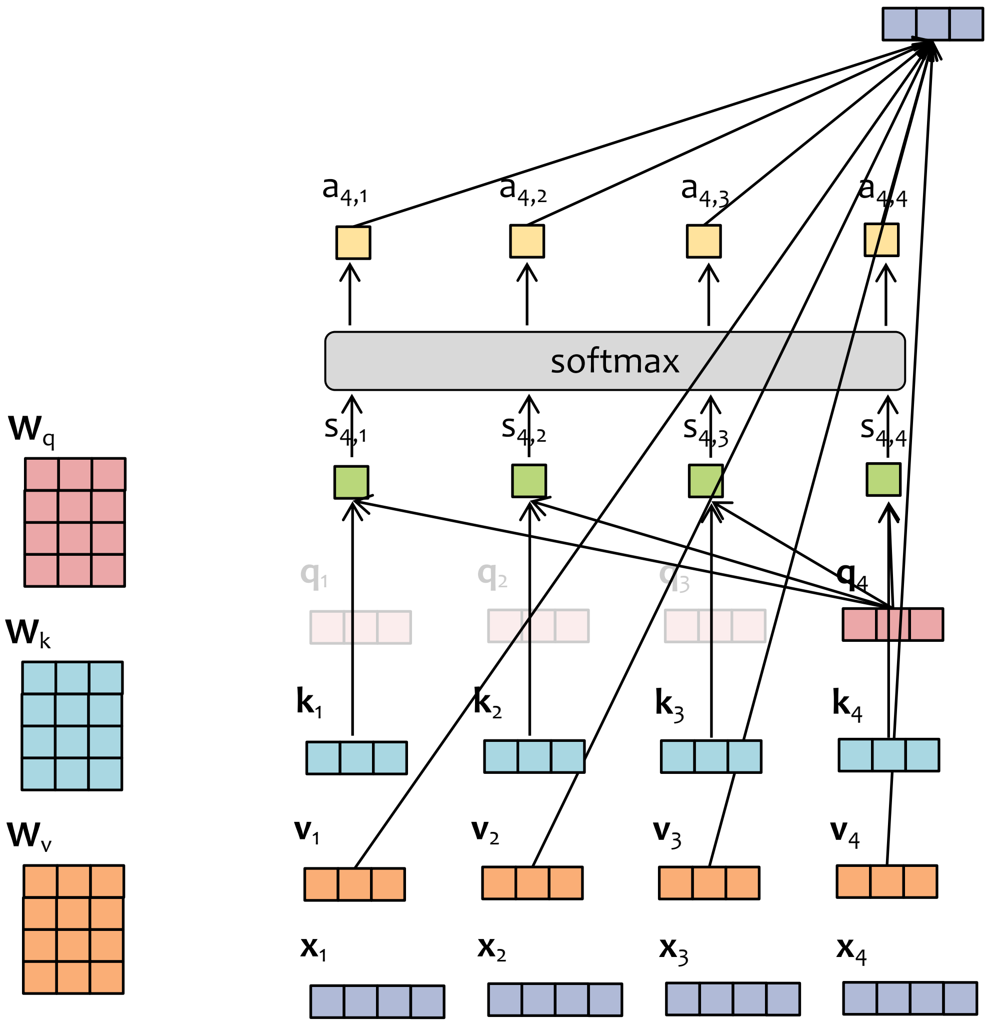


$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j$ queries

$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j$ keys

$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$ values

Scaled Dot-Product Attention



$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j} \mathbf{v}_j$$

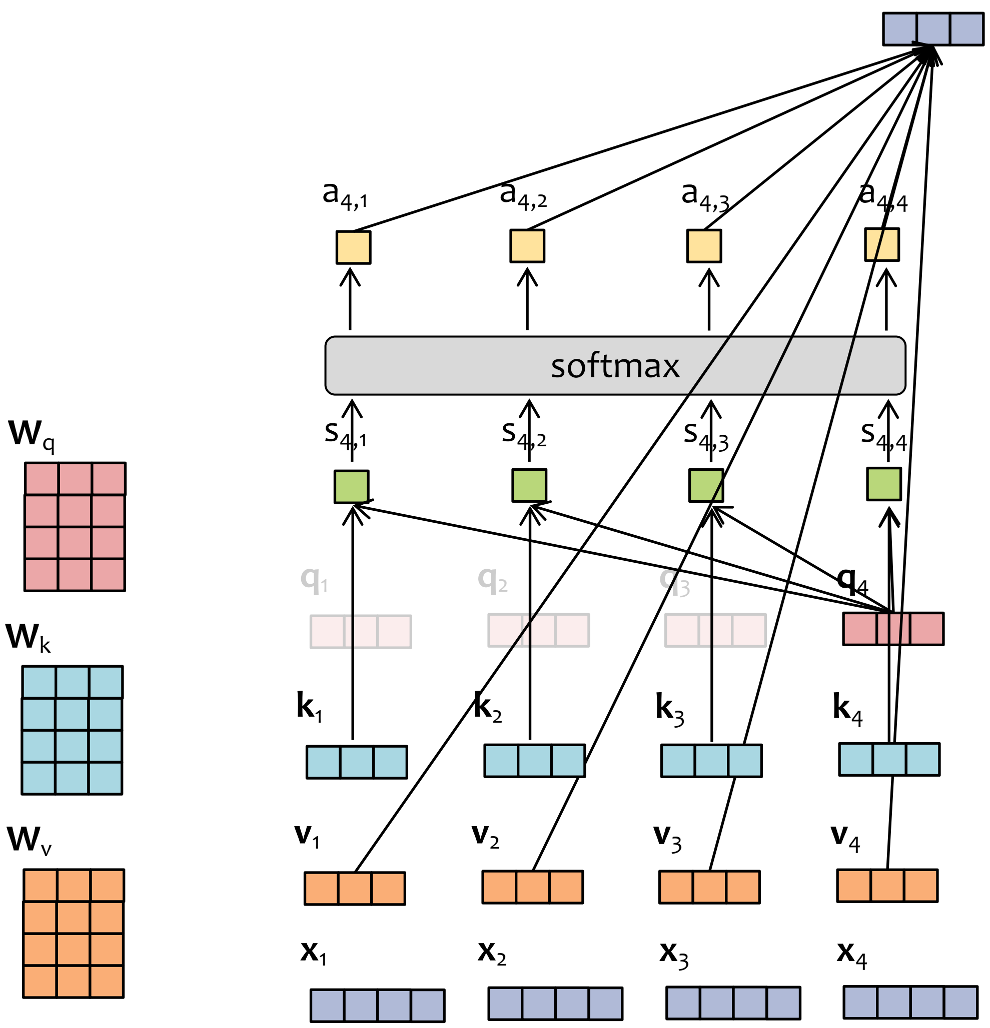
$$s_{4,j} = \mathbf{k}_j^T \mathbf{q}_4 / \sqrt{d_k}$$ scores

$$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j$$ queries

$$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j$$ keys

$$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$$ values

Scaled Dot-Product Attention



$$\mathbf{x}'_4 = \sum_{j=1}^{4} a_{4,j}\mathbf{v}_j$$

$\mathbf{a}_4 = \mathrm{softmax}(\mathbf{s}_4)$ attention weights

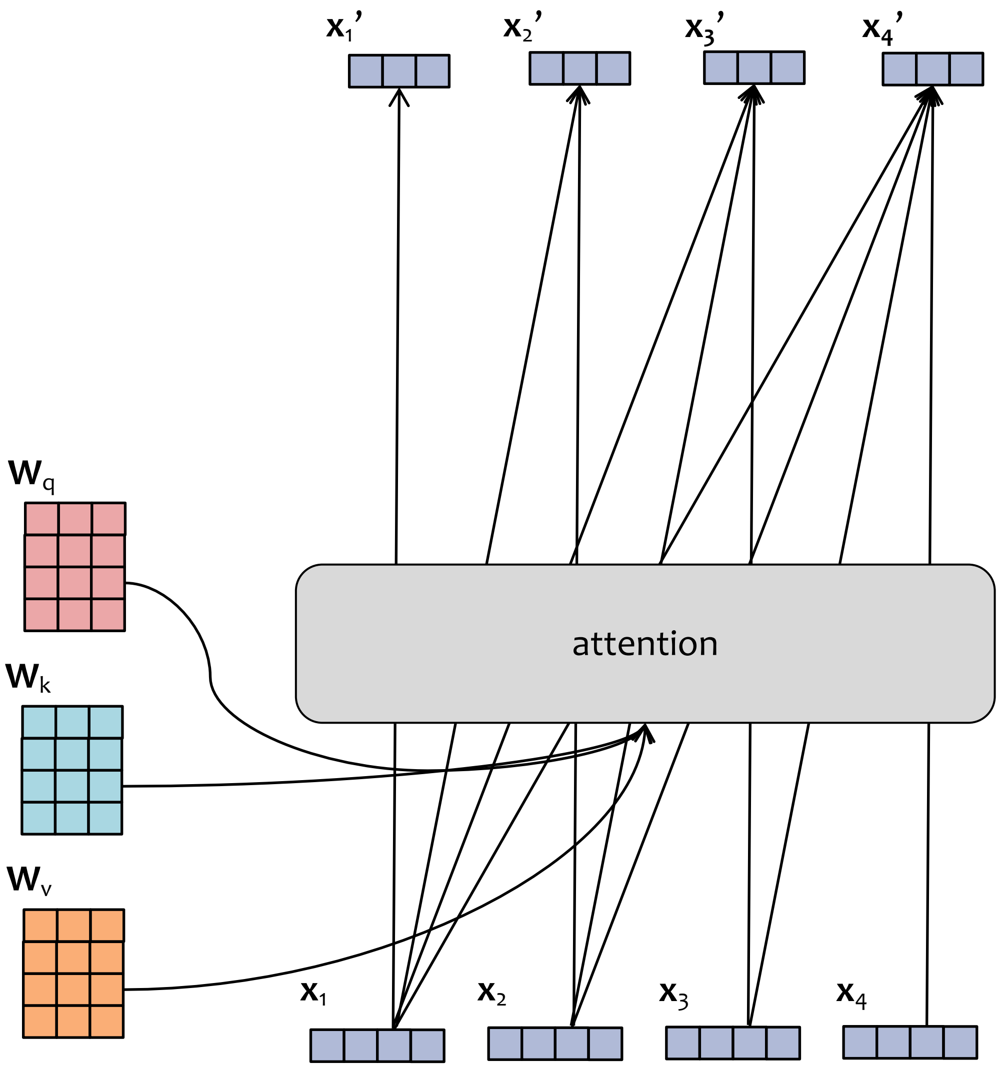
$s_{4,j} = \mathbf{k}_j^T \mathbf{q}_4 / \sqrt{d_k}$ scores

$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j$ queries

$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j$ keys

$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$ values

Scaled Dot-Product Attention



$$\mathbf{x}'_t = \sum_{j=1}^{t} a_{t,j}\mathbf{v}_j$$

$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t)$ attention weights

$s_{t,j} = \mathbf{k}_j^T \mathbf{q}_t / \sqrt{d_k}$ scores

$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j$ queries

$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j$ keys

$\mathbf{v}_j = \mathbf{W}_v^T \mathbf{x}_j$ values

Animation of 3D Convolution

Recall...

http://cs231n.github.io/convolutional-networks/

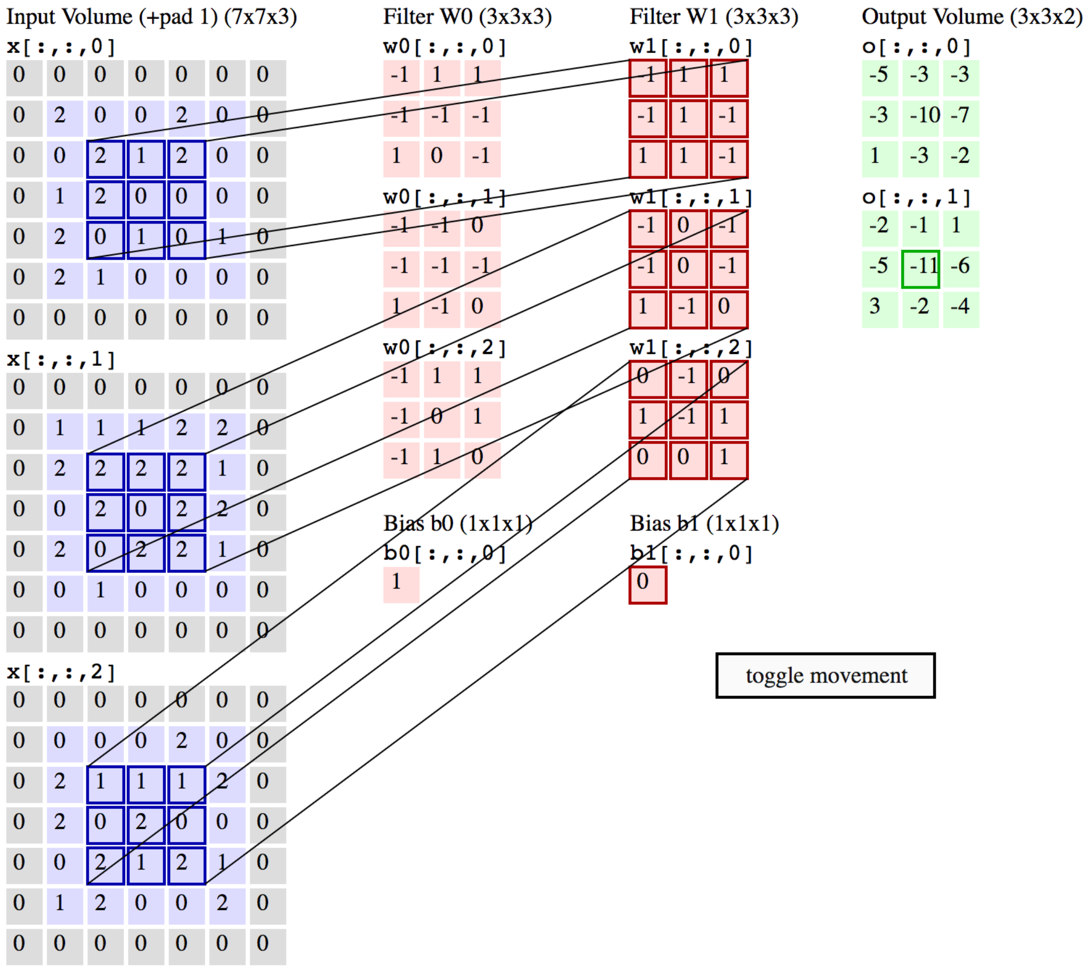
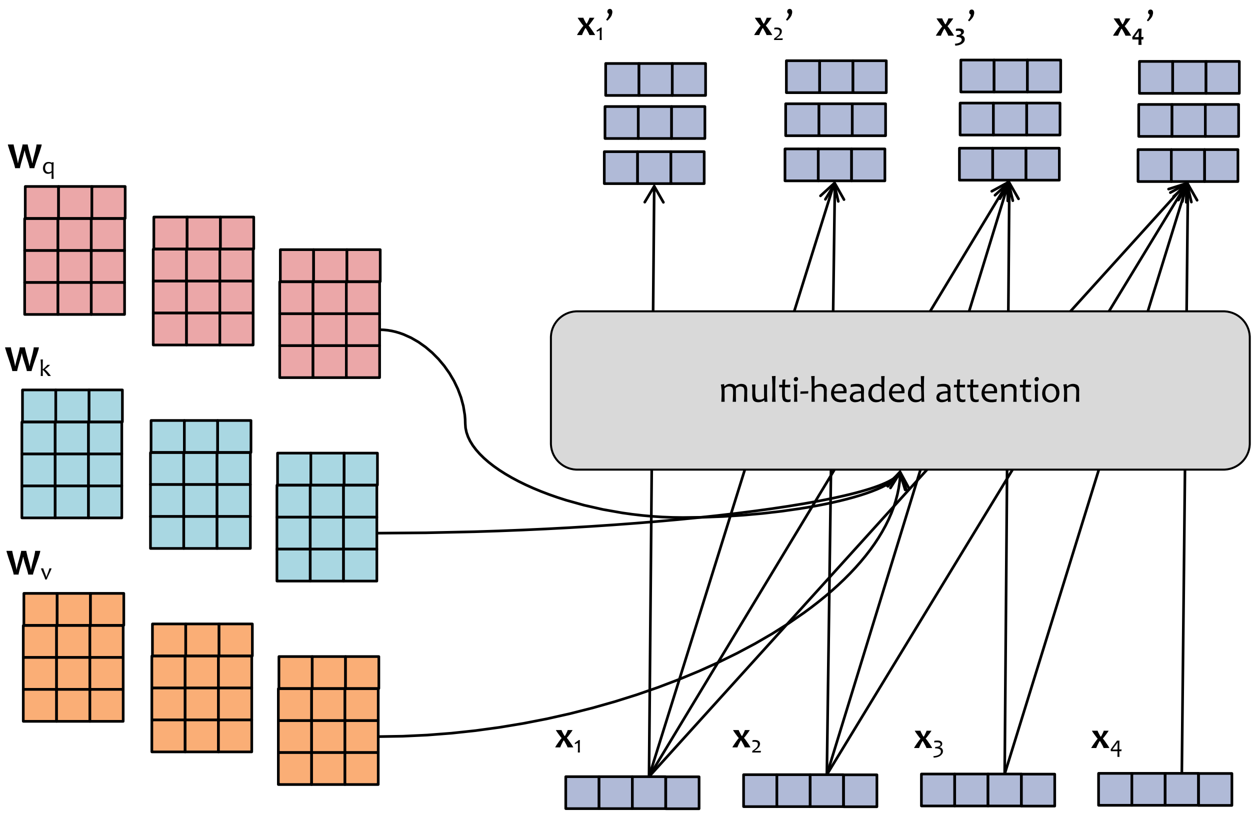


Figure from Fei-Fei Li & Andrej Karpathy & Justin Johnson (CS231N)

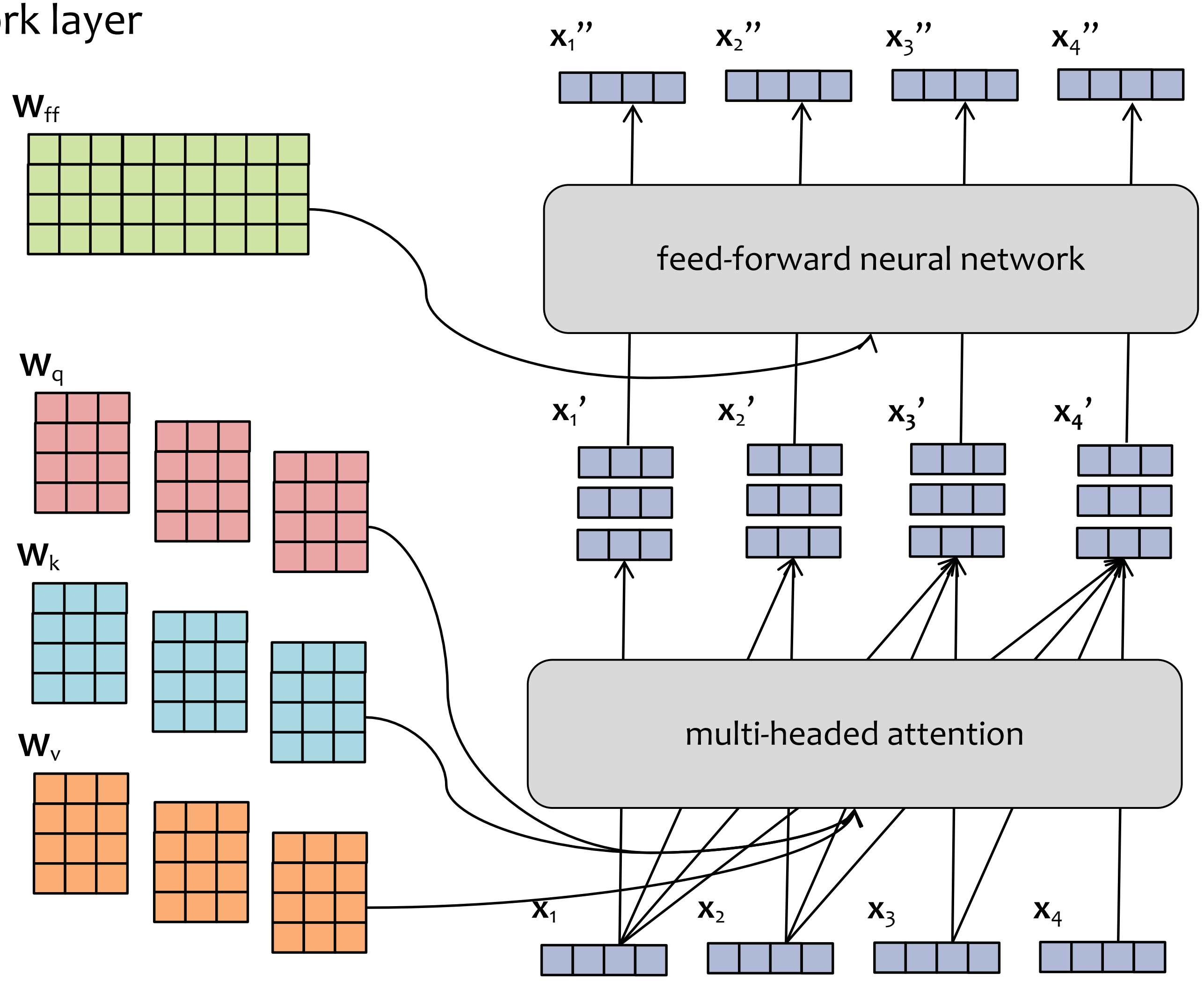
Multi-headed Attention



- Just as we can have **multiple channels** in a **convolution** layer, we can use **multiple heads** in an **attention** layer
- Each head gets **its own parameters**
- We can **concatenate** all the outputs to get a single vector for each time step

Multi-headed Attention

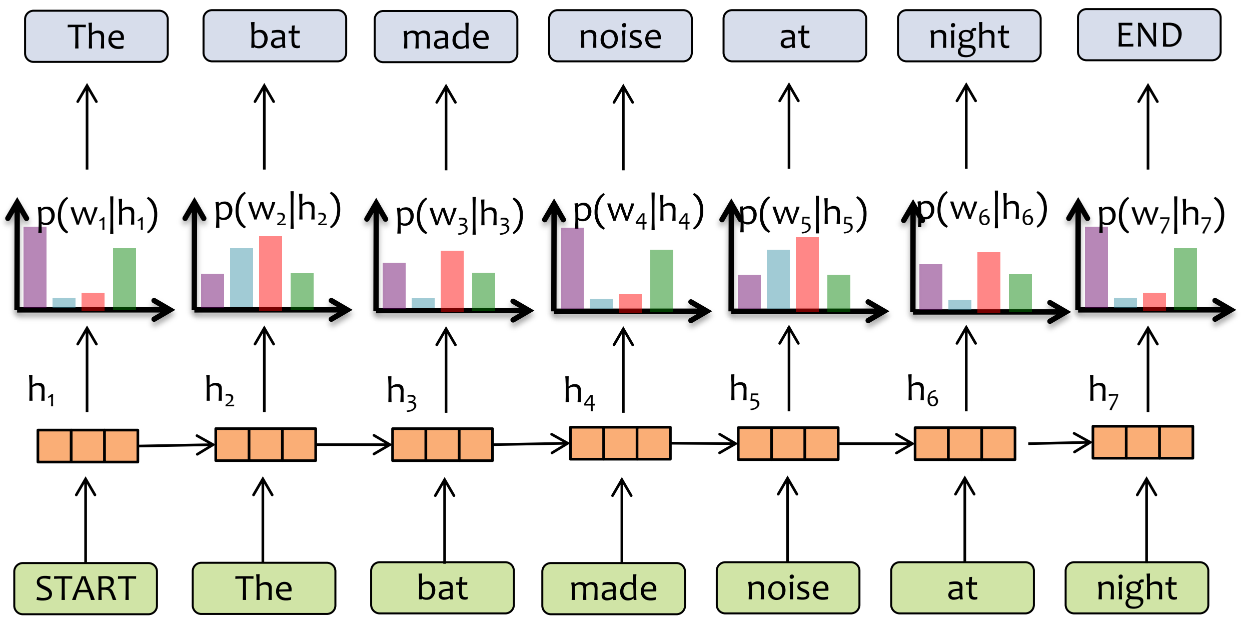
- To ensure the dimension of the **input** embedding $\mathbf{x}_t$ is the same as the **output** embedding $\mathbf{x}_t''$, add a feed-forward neural network layer



- Just as we can have **multiple channels** in a **convolution** layer, we can use **multiple heads** in an **attention** layer
- Each head gets **its own parameters**
- We can **concatenate** all the outputs to get a single vector for each time step

RNN Language Model

Recall...



<u>*Key Idea*</u>:

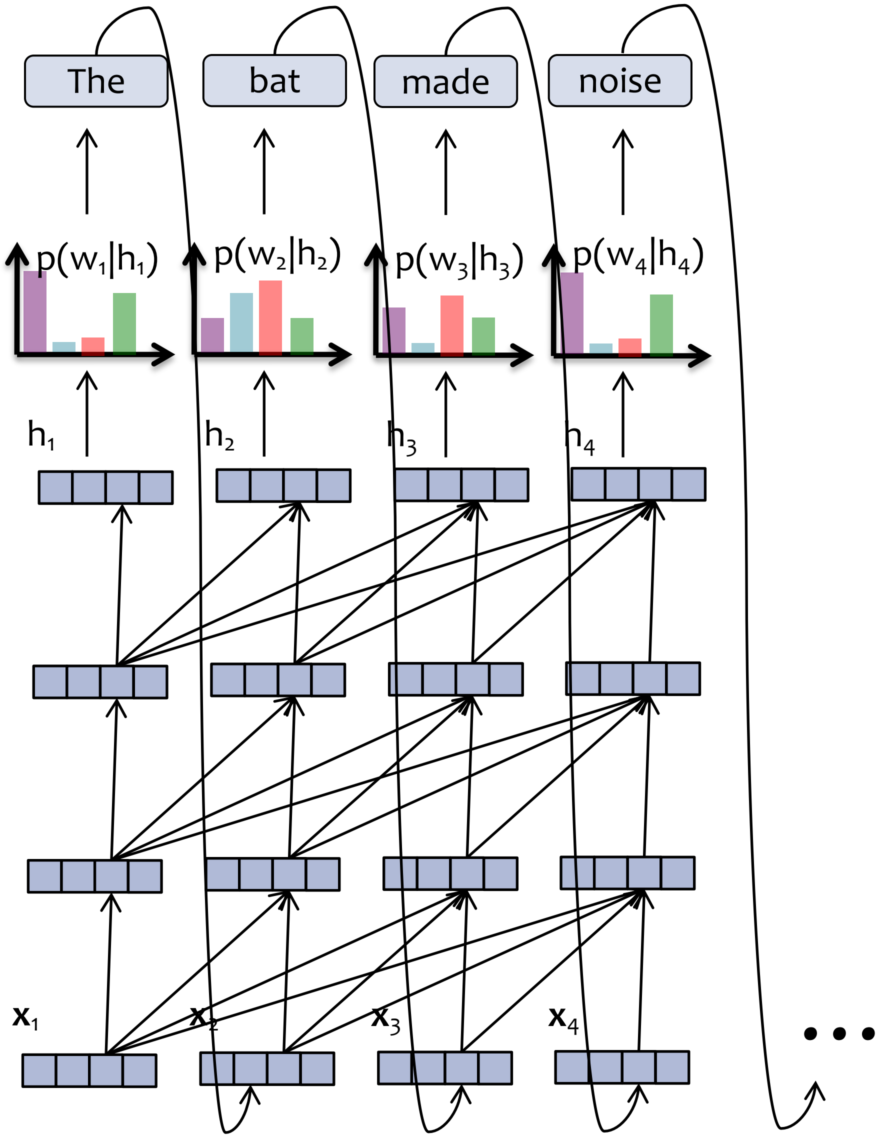
(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

Transformer Language Model

Important!

- RNN computation graph grows **linearly** with the number of input tokens
- Transformer-LM computation graph grows **quadratically** with the number of input tokens



Each layer of a Transformer LM consists of several **sublayers**:

1. attention
2. feed-forward neural network
3. layer normalization
4. residual connections

Each hidden vector looks back at the hidden vectors of the **current and previous timesteps in the previous layer.**

The language model part is just like an RNN-LM, except that a **position embedding** is attached to each word embeddings.

GPT-3

- GPT stands for Generative Pre-trained Transformer
- GPT is just a Transformer LM, but with a huge number of parameters

Model	# layers	dimension of states	dimension of inner states	# attention heads	# params
GPT (2018)	12	768	3072	12	117M
GPT-2 (2019)	48	1600	--	--	1542M
GPT-3 (2020)	96	12288	4*12288	96	175000M

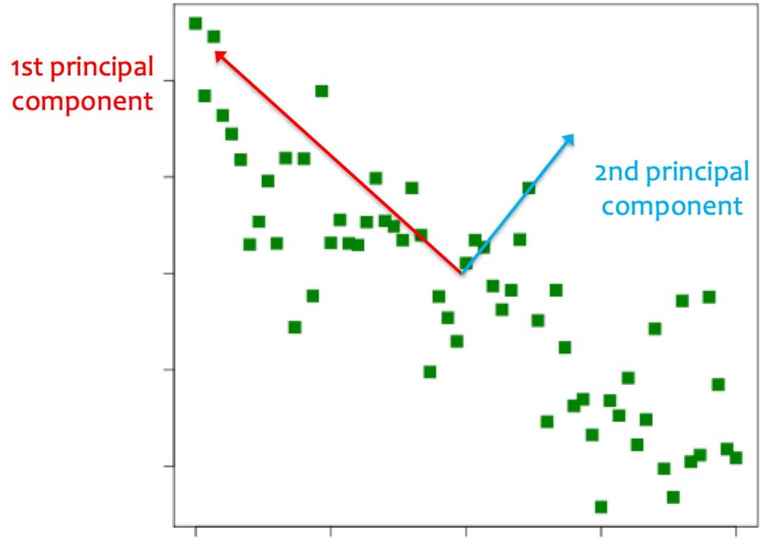
LEARNING FOR LLMS

Unsupervised Dimensionality Reduction

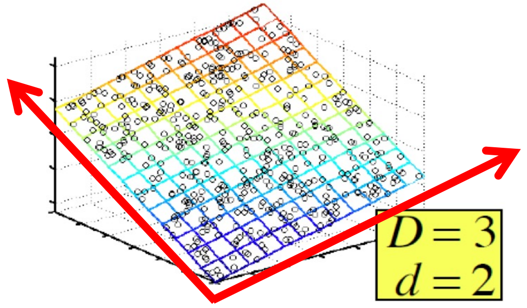
Recall...

Principal Component Analysis (PCA)

- **Assumption:** the data lies on a low K-dimensional linear subspace
- **Goal:** identify the axes of that subspace, and project each point onto hyperplane
- **Algorithm:** find the K eigenvectors with largest eigenvalue using classic matrix decomposition tools



PCA Example: 2D Gaussian Data



https://commons.wikimedia.org/wiki/File:Scatter_diagram_for_quality_characteristic_XXX.svg

The Start of Deep Learning

- The architectures of modern deep learning have a long history:
  - 1960s: Rosenblatt's 3-layer multi-layer perceptron, ReLU)
  - 1970-80s: RNNs and CNNs
  - 1990s: linearized self-attention
- The spark for deep learning came in 2006 thanks to **pre-training** (e.g., Hinton & Salakhutdinov, 2006)

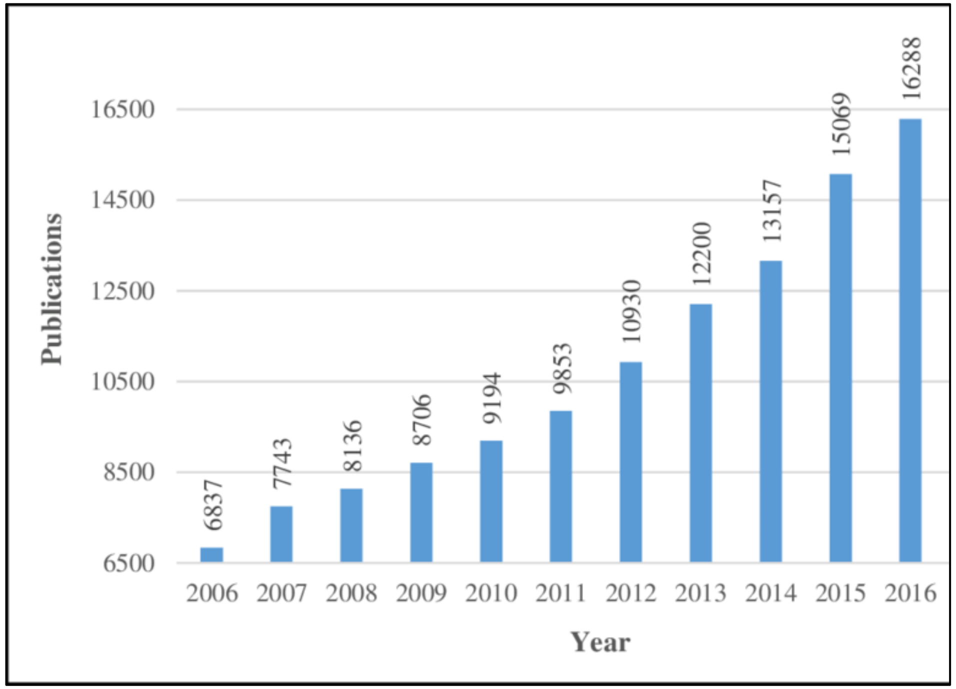
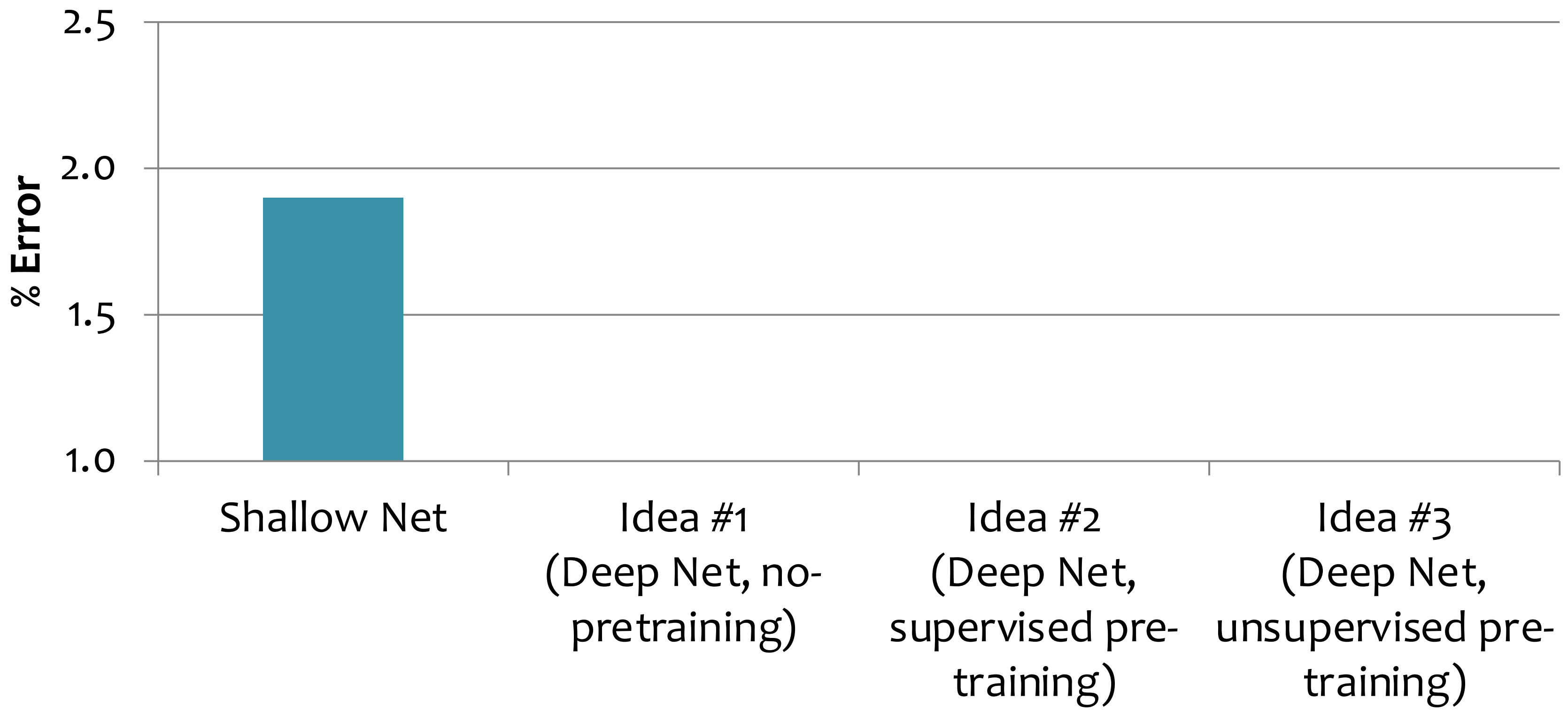


Figure from Vargas et al. (2017)

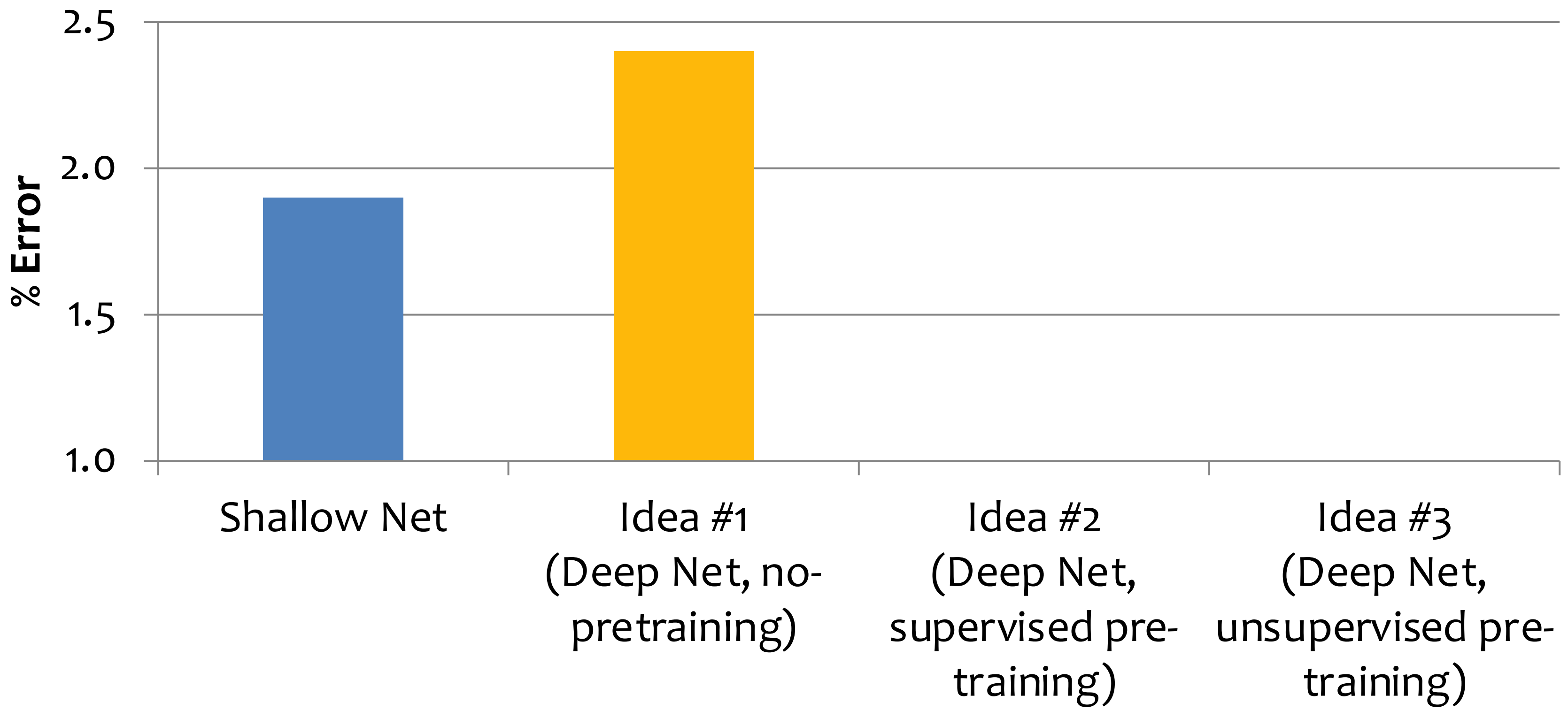
Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



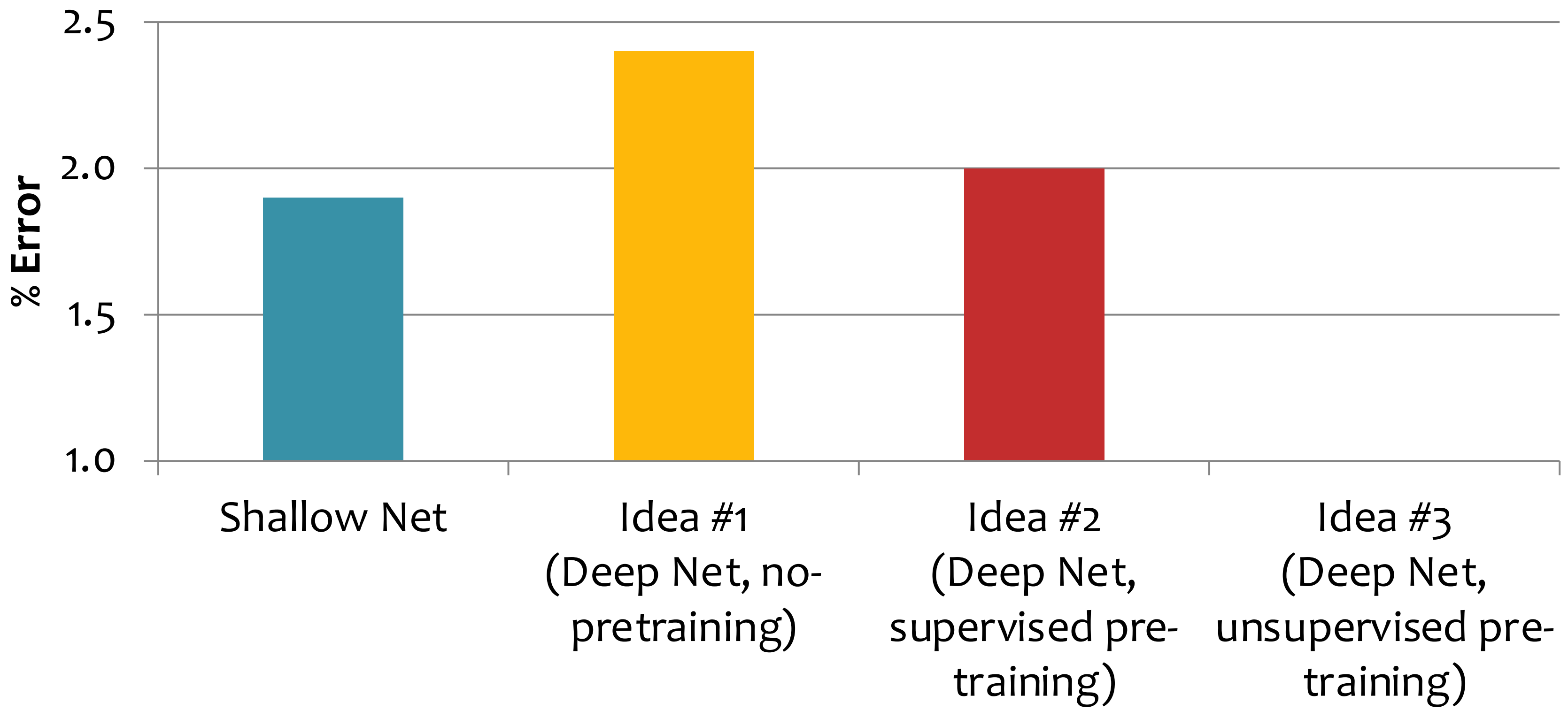
Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



Idea #3: Unsupervised Pre-training

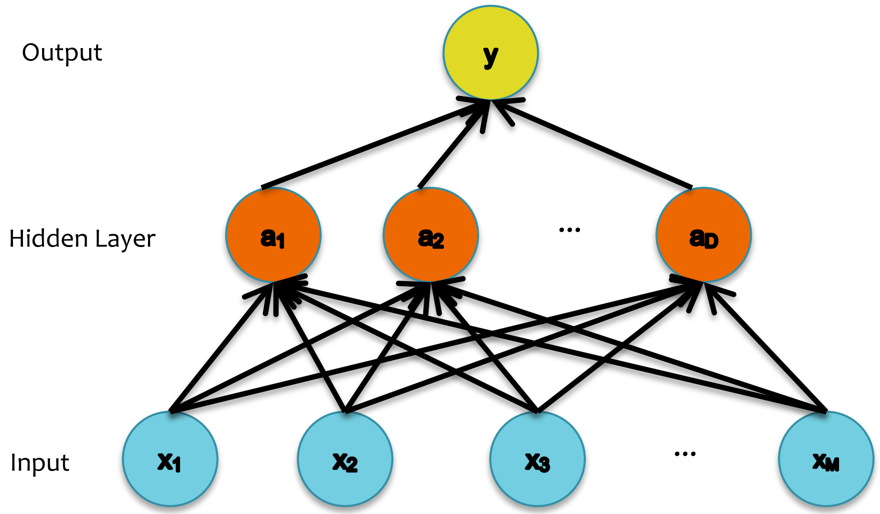
- **Idea #3: (Two Steps)**
  - Use our original idea, but **pick a better starting point**
  - **Train each level** of the model in a **greedy** way

1. Unsupervised Pre-training
   - Use **unlabeled** data
   - Work bottom-up
     - Train hidden layer 1. Then fix its parameters.
     - Train hidden layer 2. Then fix its parameters.
     - …
     - Train hidden layer n. Then fix its parameters.
2. Supervised Fine-tuning
   - Use **labeled** data to train following “Idea #1”
   - Refine the features by backpropagation so that they become tuned to the end-task

The solution:
Unsupervised pre-training

Unsupervised pre-training of the first layer:

- What should it predict?
- What else do we observe?
- **The input!**

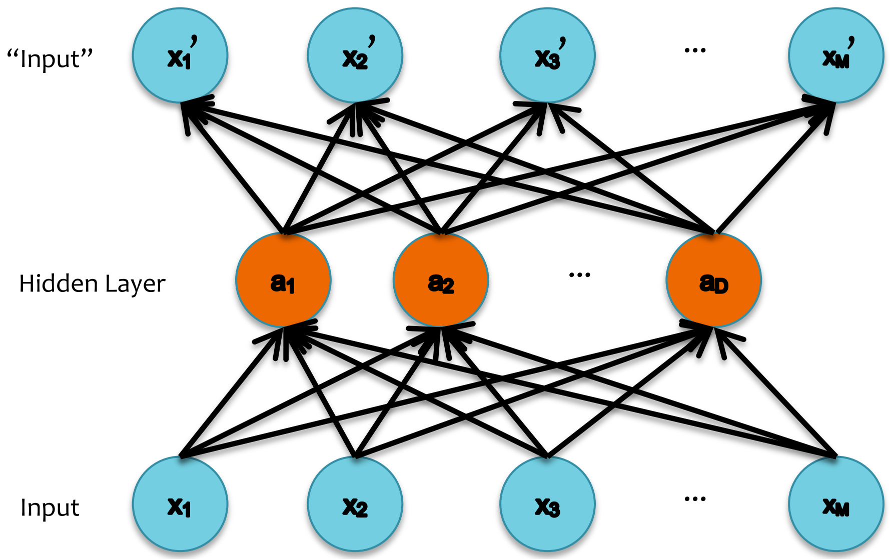


The solution:
Unsupervised pre-training

Unsupervised pre-training of the first layer:

- What should it predict?
- What else do we observe?
- **The input!**

This topology defines an Auto-encoder.



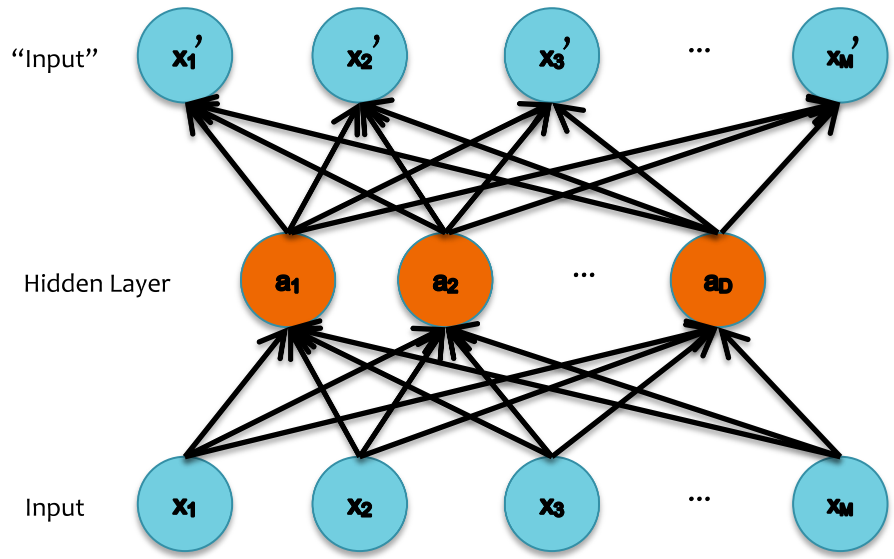
Auto-Encoders

Key idea: Encourage z to give small reconstruction error:

- x' is the *reconstruction* of x
- Loss = || x – DECODER(ENCODER(x)) ||²
- Train with the same backpropagation algorithm for 2-layer Neural Networks with x_m as both input and output.

DECODER: x' = h(W'z)

ENCODER: z = h(Wx)

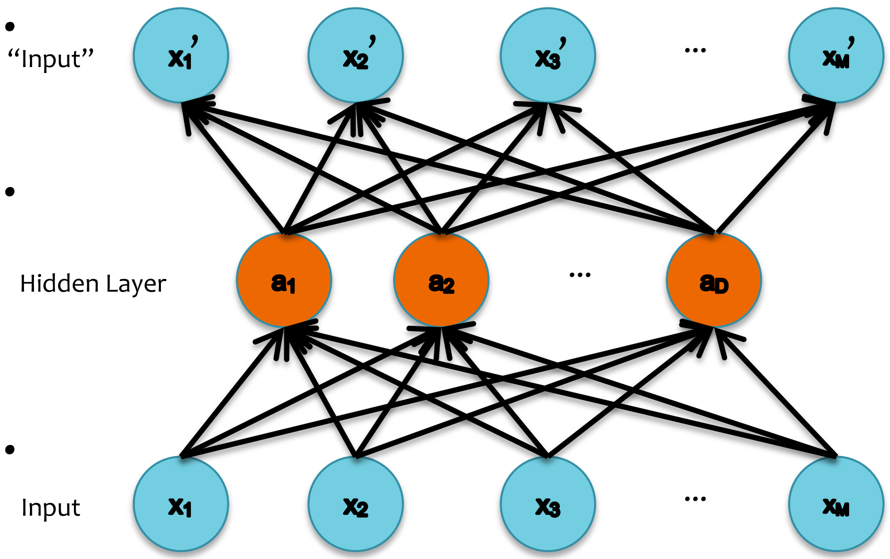


Slide adapted from Raman Arora

The solution:
Unsupervised pre-training

Unsupervised pre-training

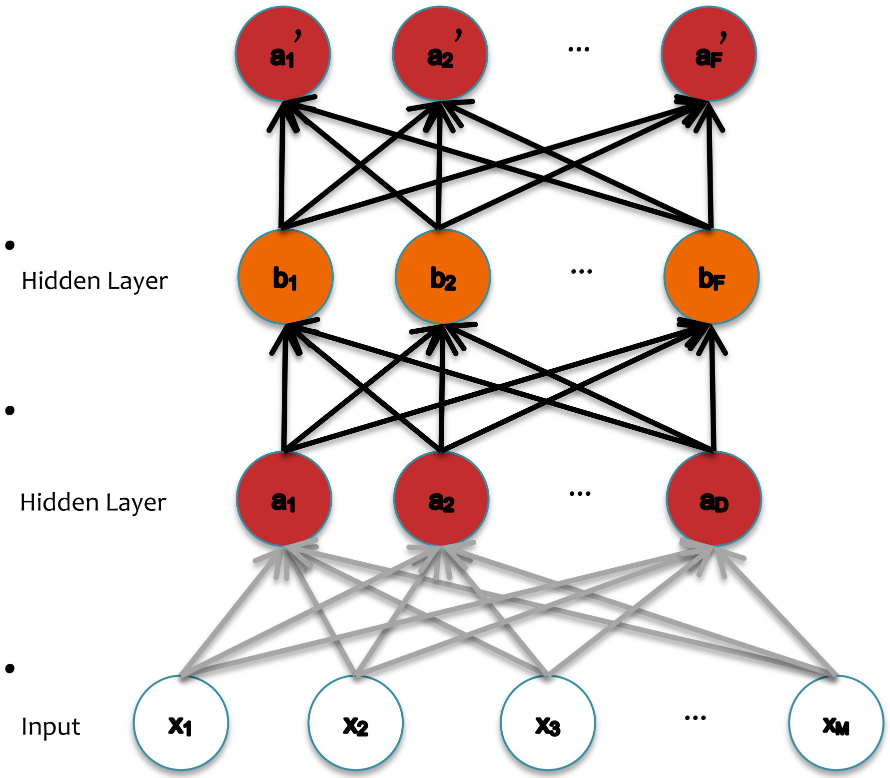
- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.



The solution:
Unsupervised pre-training

Unsupervised pre-training

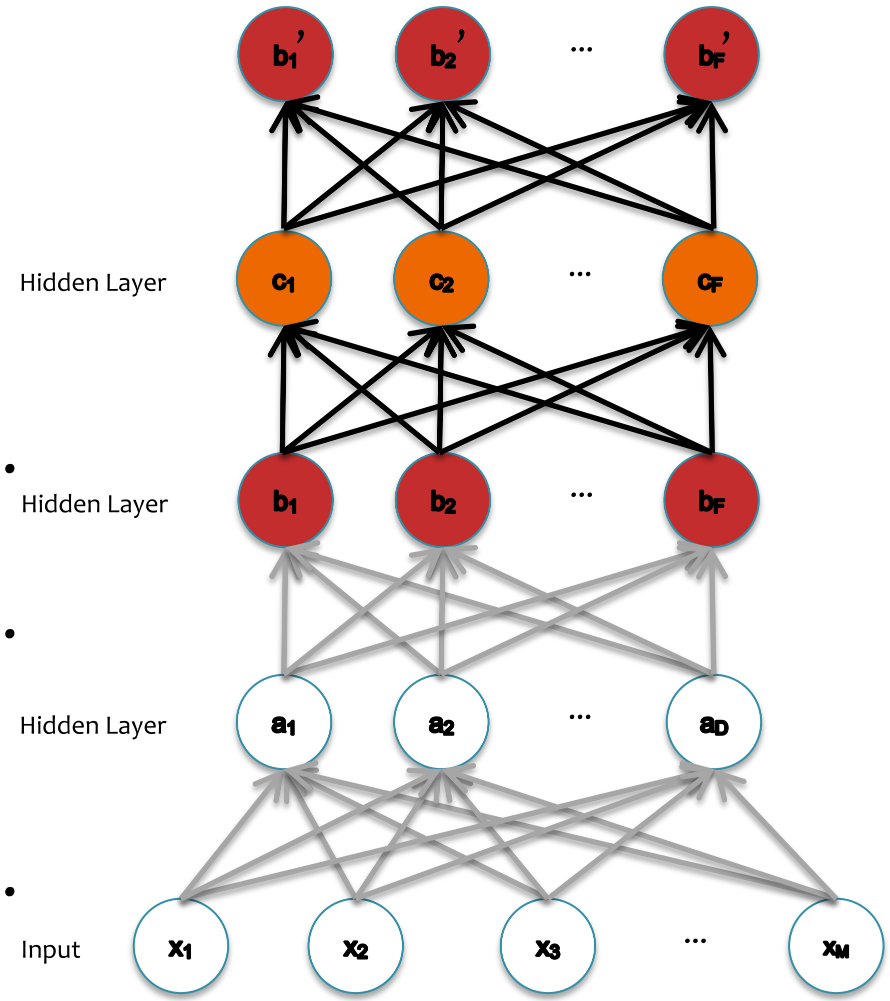
- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.



The solution:
Unsupervised pre-training

Unsupervised pre-training

- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.

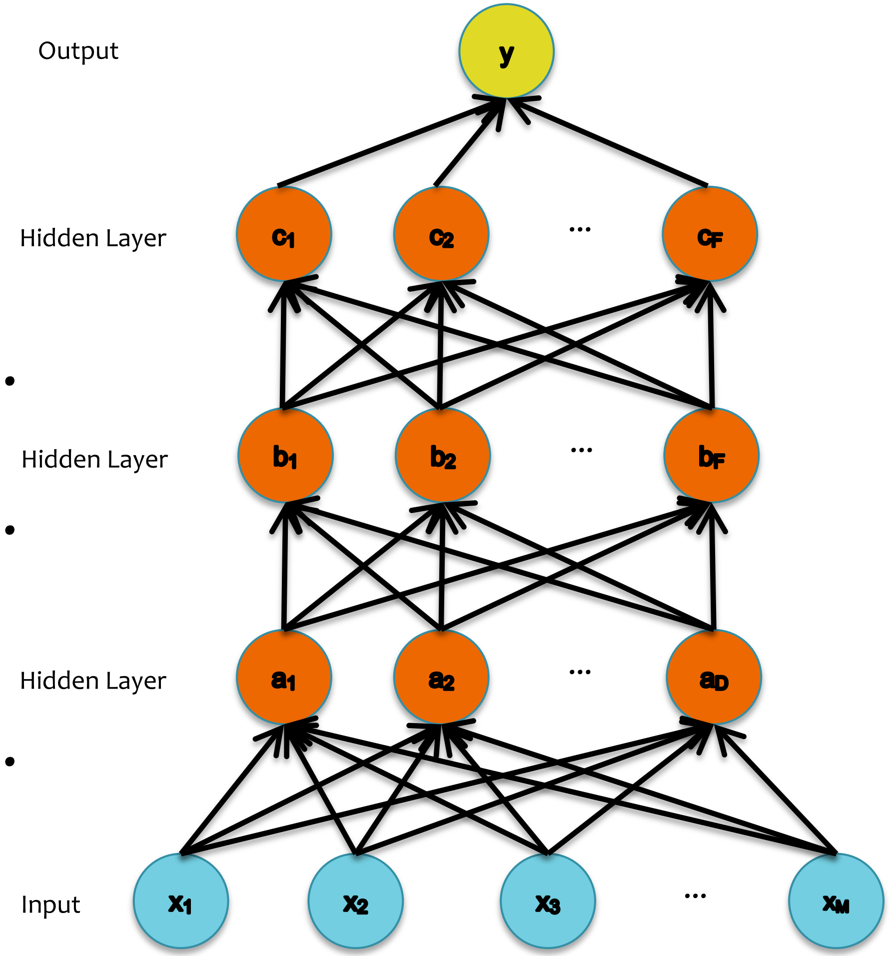


The solution:
Unsupervised pre-training

Unsupervised pre-training

- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.

Supervised fine-tuning
Backprop and update all parameters



Deep Network Training

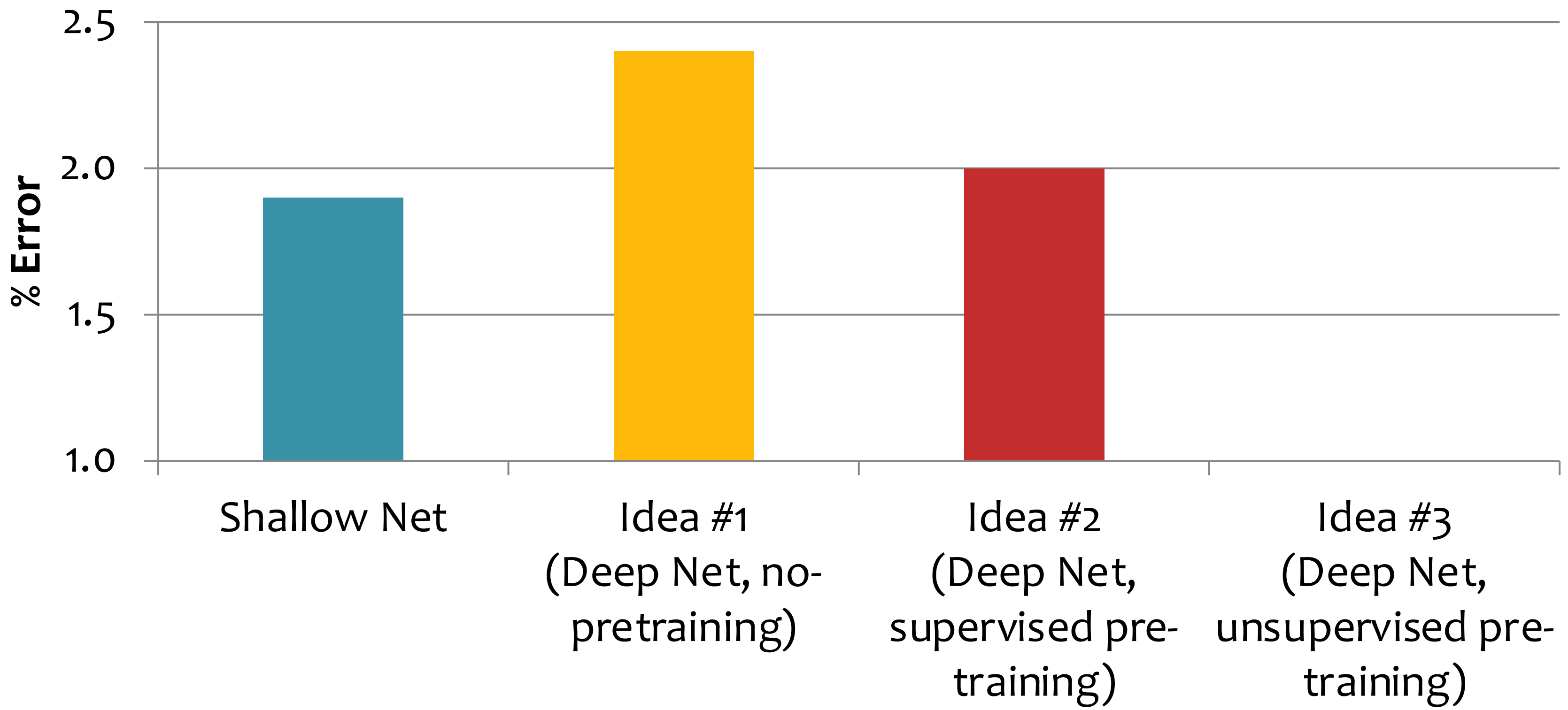
- **Idea #1:**
  1. Supervised fine-tuning only

- **Idea #2:**
  1. Supervised layer-wise pre-training
  2. Supervised fine-tuning

- **Idea #3:**
  1. Unsupervised layer-wise pre-training
  2. Supervised fine-tuning

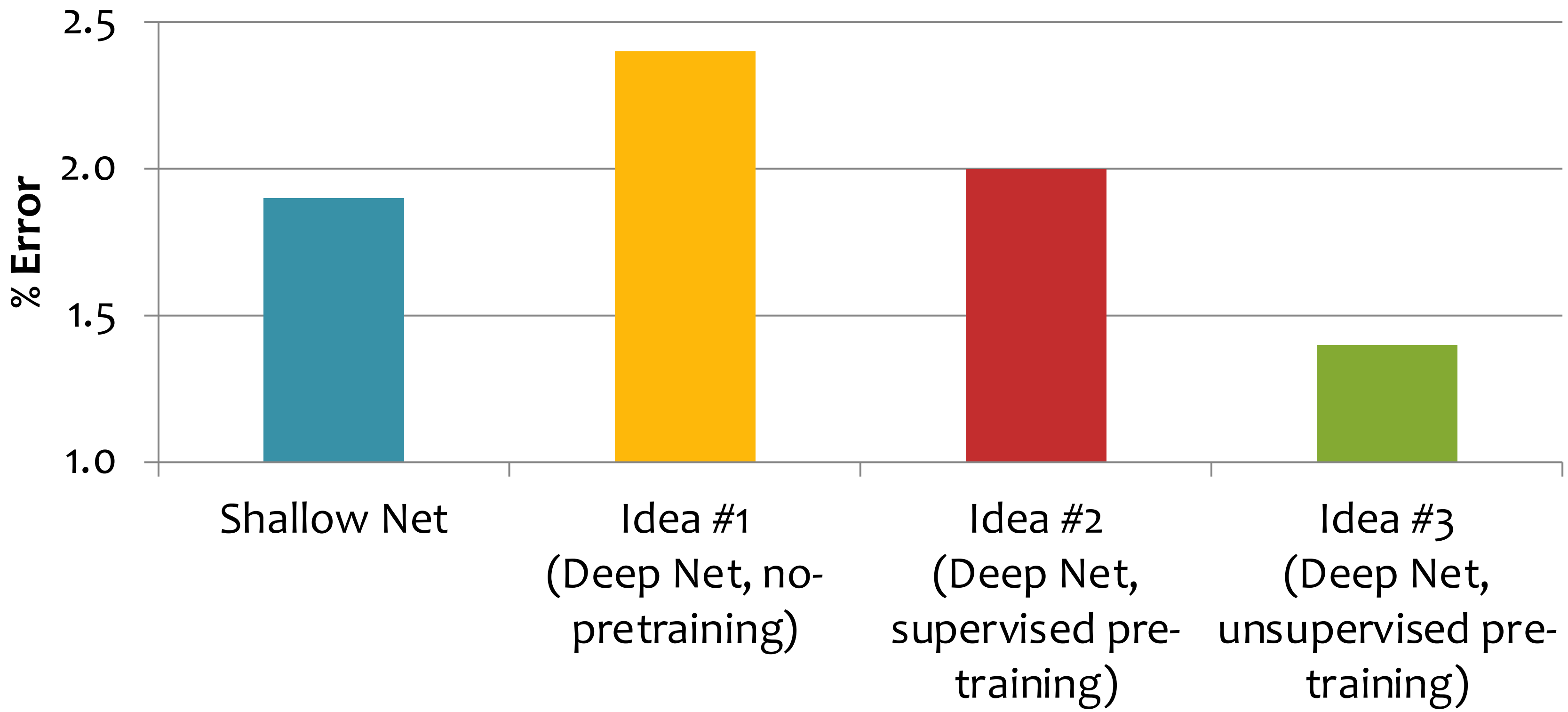
Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)

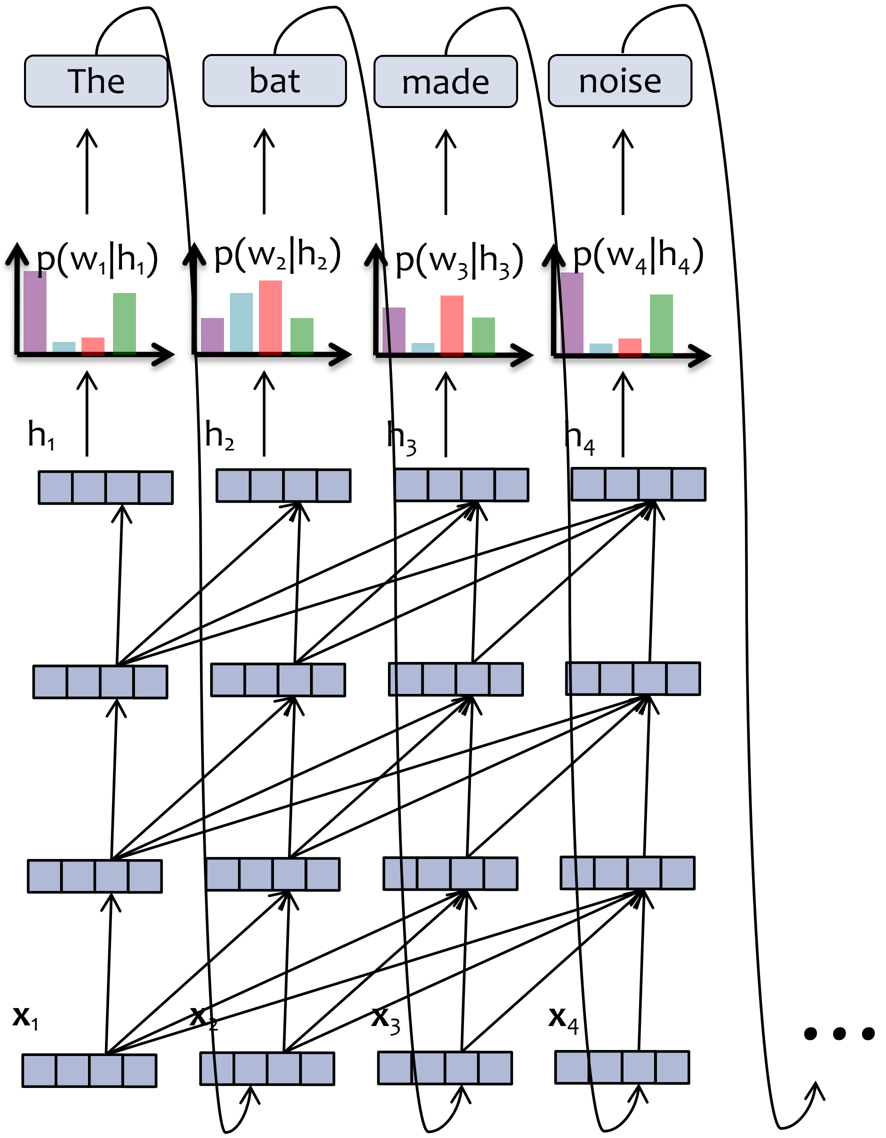


Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



Transformer Language Model



Generative pre-training for a deep language model:

- each training example is an (unlabeled) sentence
- the objective function is the likelihood of the observed sentence

Practically, we can **batch** together many such training examples to make training more efficient

Training Data for LLMs

GPT-3 Training Data:

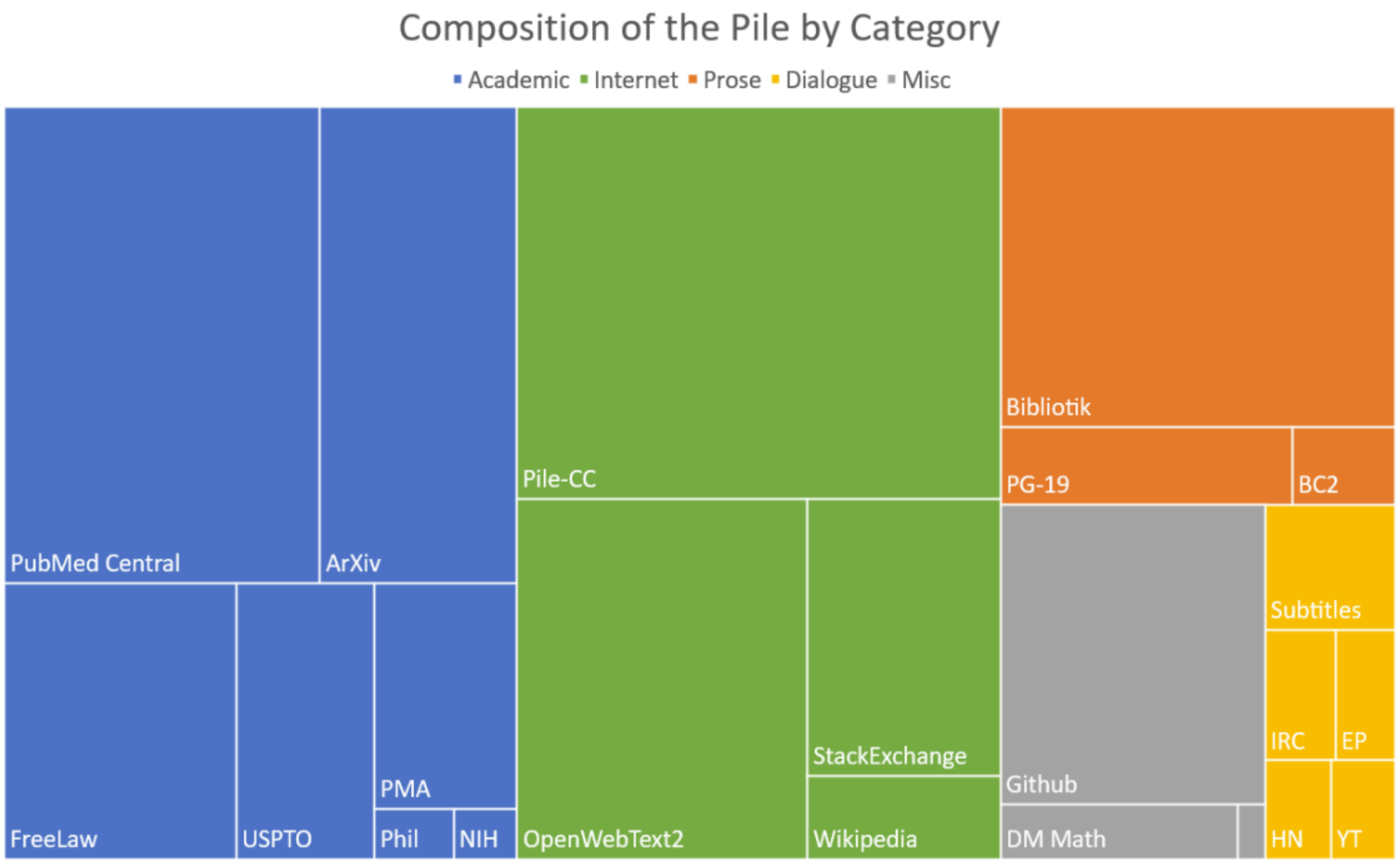
Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table from http://arxiv.org/abs/2005.14165

Training Data for LLMs

The Pile:

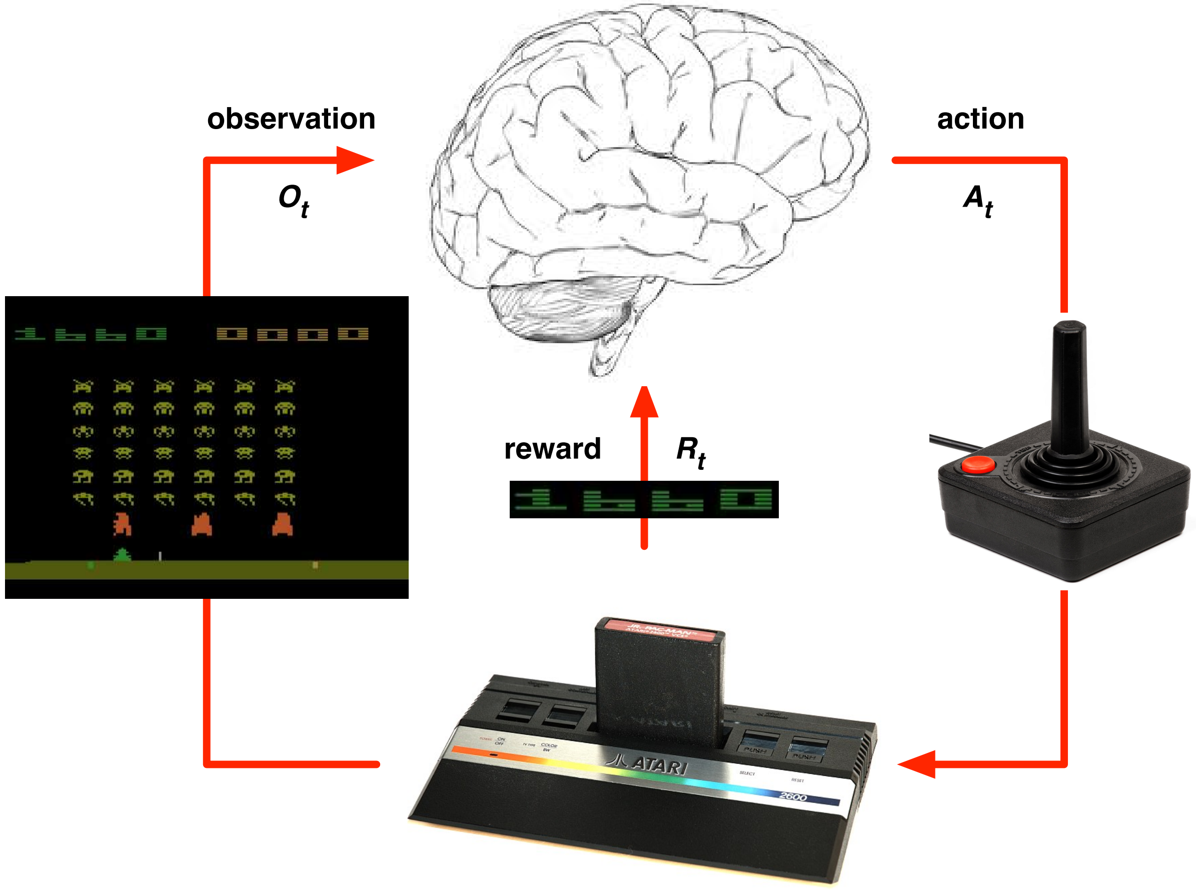
- An open source dataset for training language models
- Comprised of 22 smaller datasets
- Favors high quality text
- 825 Gb ≈ 1.2 trillion tokens



Recall...

Playing Atari with Deep RL

- Setup: RL system observes the pixels on the screen
- It receives rewards as the game score
- Actions decide how to move the joystick / buttons



Figures from David Silver (Intro RL lecture)

Deep Q-learning

Recall…

- Algorithm 4: Online learning of Q^* (parametric form)
  - Inputs: discount factor γ,
    an initial state s_0,
    learning rate α
  - Initialize parameters $\Theta^{(0)}$
  - For $t = 0, 1, 2, \ \ldots$
    - Gather training sample (s_t, a_t, r_t, s_{t+1})
    - Update $\Theta^{(t)}$ by taking a step opposite the gradient
    - $\Theta^{(const)} \leftarrow \Theta^{(t)}$
      $\Theta^{(t+1)} \leftarrow \Theta^{(t)} - \alpha \nabla_{\Theta^{(t)}} \ell\left(\Theta^{(const)}, \Theta^{(t)}\right)$

where

$$\nabla_{\Theta} \ell\left(\Theta^{(const)}, \Theta^{(t)}\right) = 2\left(y - Q\left(s, a; \Theta^{(t)}\right)\right) \nabla_{\Theta^{(t)}} Q\left(s, a; \Theta^{(t)}\right)$$

$$= 2\left(r + \gamma \max_{a'} Q\left(s', a'; \Theta^{(const)}\right) - Q\left(s, a; \Theta^{(t)}\right)\right) \nabla_{\Theta^{(t)}} Q\left(s, a; \Theta^{(t)}\right)$$

RLHF

- **InstructGPT** uses Reinforcement Learning with Human Feedback (RLHF) to **fine-tune** a **pre-trained** GPT model
- From the paper: “In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters.”

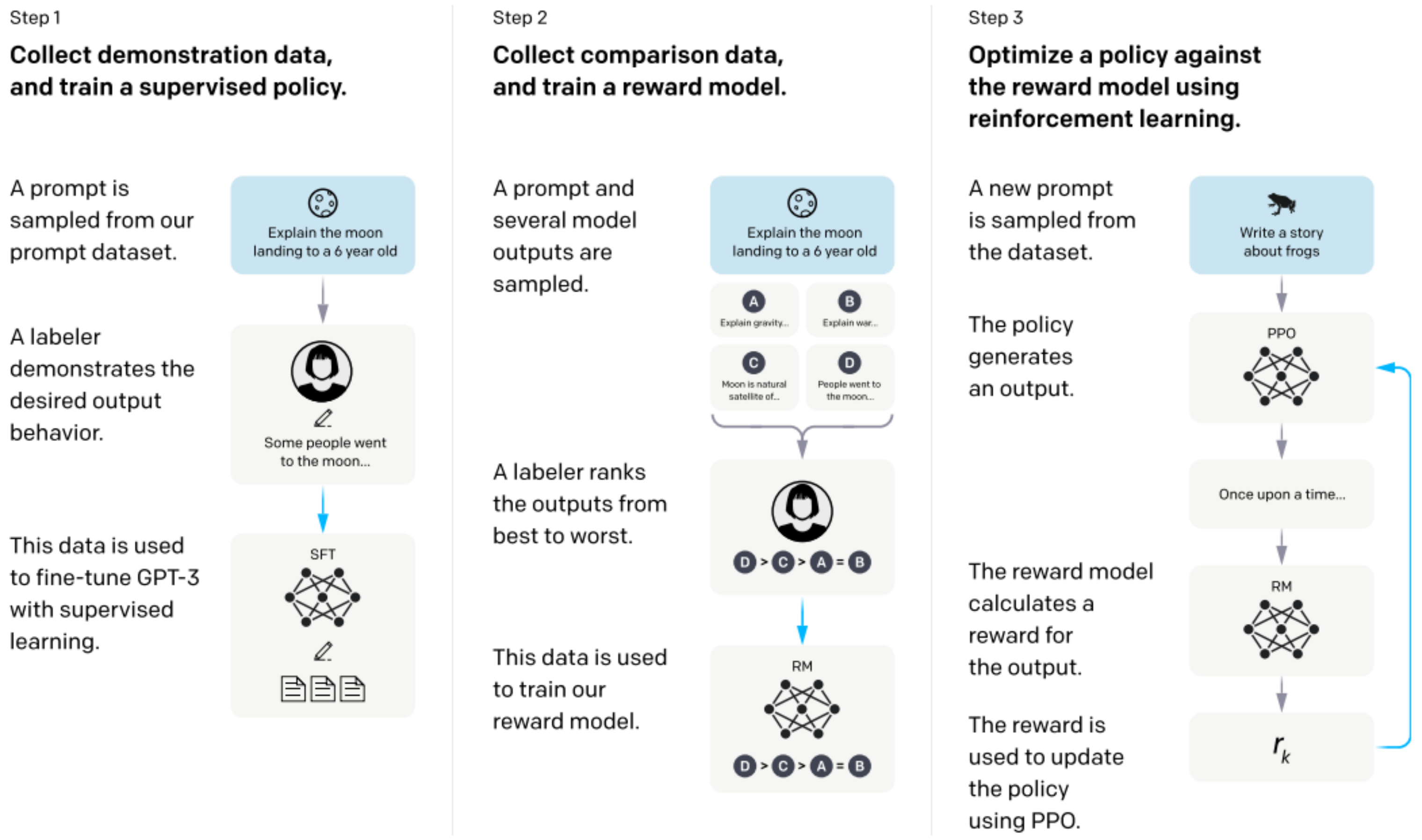


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers. See Section 3 for more details on our method.

Figure from https://arxiv.org/pdf/2203.02155.pdf

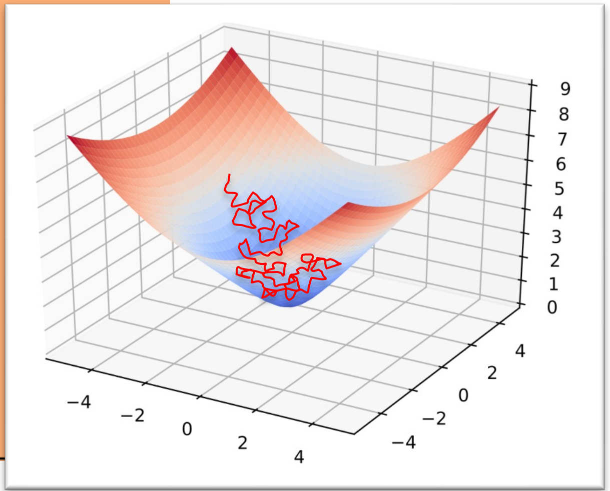
OPTIMIZATION FOR LLMS

Recall…

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

procedure SGD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
    while not converged **do**
        for $i \in$ shuffle($\{1, 2, \ldots, N\}$) **do**
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
    return $\boldsymbol{\theta}$



In practice, it is common to implement SGD using sampling **without** replacement (i.e. shuffle({1,2,… N}), even though most of the theory is for sampling **with** replacement (i.e. Uniform({1,2,… N}).

per-example objective:

$$J^{(i)}(\boldsymbol{\theta})$$

original objective:

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

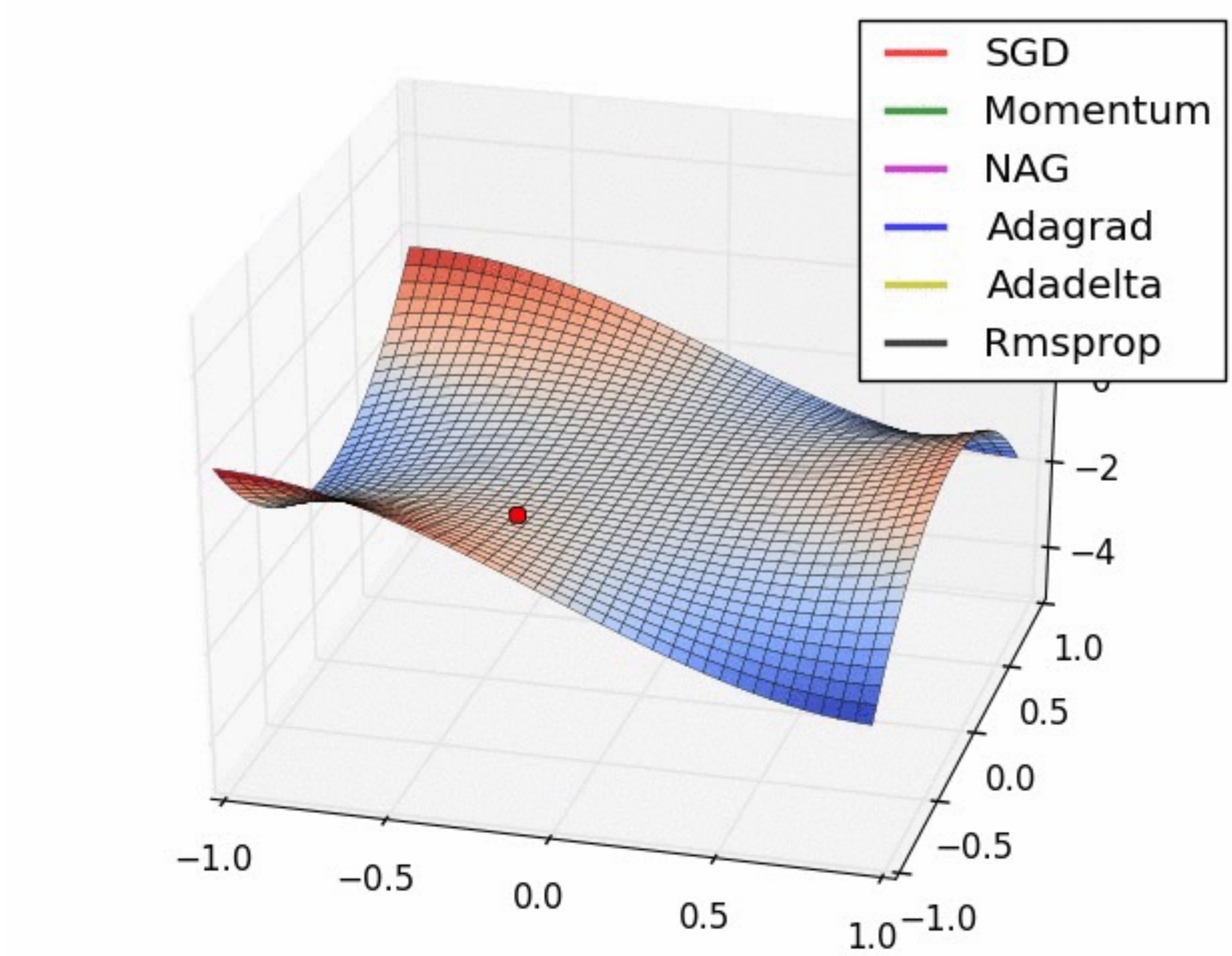


Figure from https://imgur.com/a/Hqolp#2dKCQHh

Adam

- Adam combines elements of two popular algorithms:
  1. **AdaGrad**
     each parameter gets its own learning rate
  2. **RMSProp**
     keeps a moving average of recent gradients

Adam

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Memory Usage of LLMs

How to store a large language model in memory?

- **full precision:** 32-bit floats
- **half precision:** 16-bit floats
- Using half precision not only **reduces memory,** it also **speeds up** GPU computation
- *"Peak float16 matrix multiplication and convolution performance is 16x faster than peak float32 performance on A100 GPUs."* from Pytorch docs

Model	Megatron-LM	GPT-3
# parameters	8.3 billion	175 billion
full precision	30 Gb	651 Gb
half precision	15 Gb	325 Gb

GPU / TPU	Max Memory
TPU v2	16 Gb
TPU v3/v4	32 Gb
Tesla V100 GPU	32 Gb
NVIDIA RTX A6000	48 Gb
Tesla A100 GPU	80 Gb

Two Types of Distributed Training

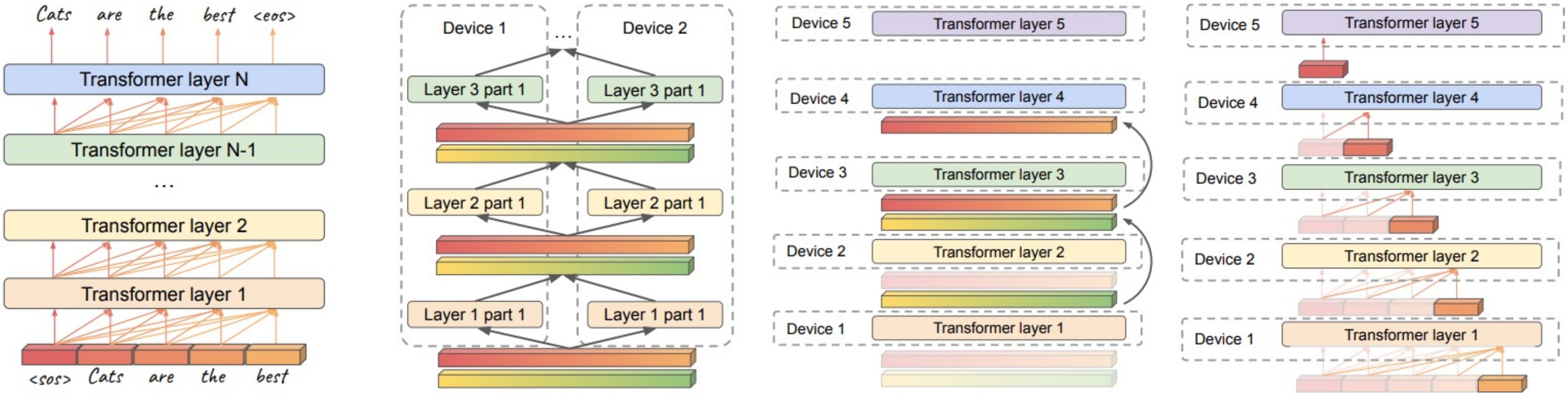
Data Parallel

- **key idea:** (almost trivial) parallelism achieved by distributing the batches across multiple GPUs
- **key challenge:** sharing / updating a single set of parameters across all devices

Model Parallel

- **key idea:** (very tricky) parallelism achieved by dividing the model parameters/computation across multiple GPUs
- **key challenge:** maintaining high speedup even though some of the model computation *must* be done sequentially (e.g. the backward computation must happen *after* the forward computation)

Distributed Training: Model Parallel



(a) Transformer-based LM

(b) Operation partitioning (Megatron-LM)

(c) Microbatch-based pipeline parallelism (GPipe)

(d) Token-based pipeline parallelism (TeraPipe)

There are a variety of different options for how to distribute the model computation / parameters across multiple devices.

Matrix multiplication comprises most Transformer LM computation and can be divided along rows/columns of the respective matrices.

The most natural division is by layer: each device computes a subset of the layers, only that device stores the parameters and computation graph for those layers.

A more efficient solution is to divide computation by token *and* layer. This requires careful division of work and is specific to the Transformer LM.

Figure from https://arxiv.org/pdf/2102.07988.pdf

Cost to train

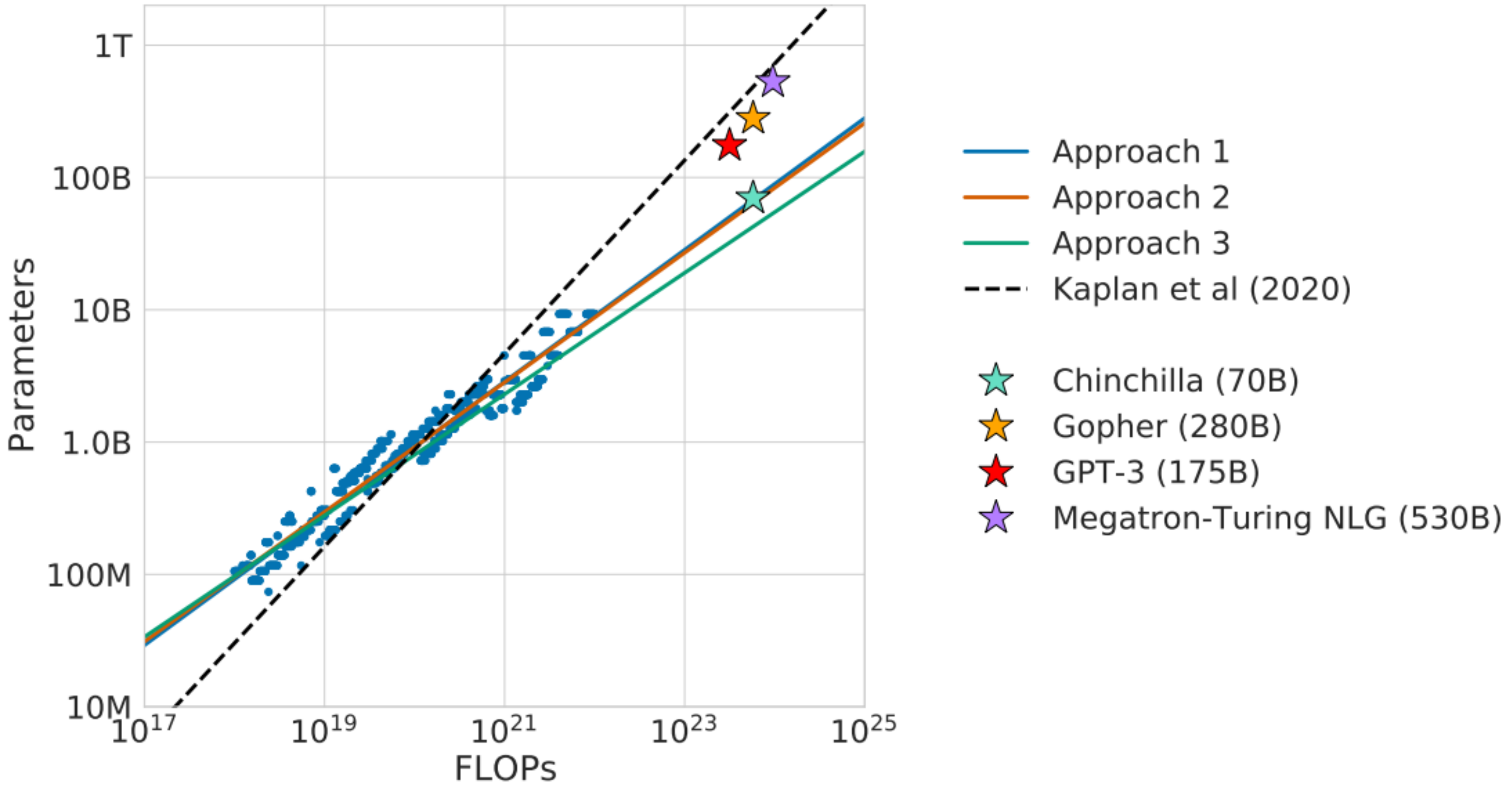


Figure from https://arxiv.org/pdf/2203.15556.pdf

SOCIETAL IMPACTS OF LLMS

Societal Impacts of ChatGPT

In-class exercise:
What are the potential societal impacts of ChatGPT?

Summary

- Task: Language Modeling
  - noisy channel models (speech / MT)
  - (historical) Large LMs (n-gram models)
- Model: GPT
  - Attention (computation graph)
  - Transformer-LM (cf. RNN-LM)
- Learning for LLMs
  - Pre-training (unsupervised learning)
  - Reinforcement Learning with Human Feedback (deep RL)
- Optimization for LLMs
  - Adam (cf. SGD)
  - Distributed training
- Societal Impacts of LLMs