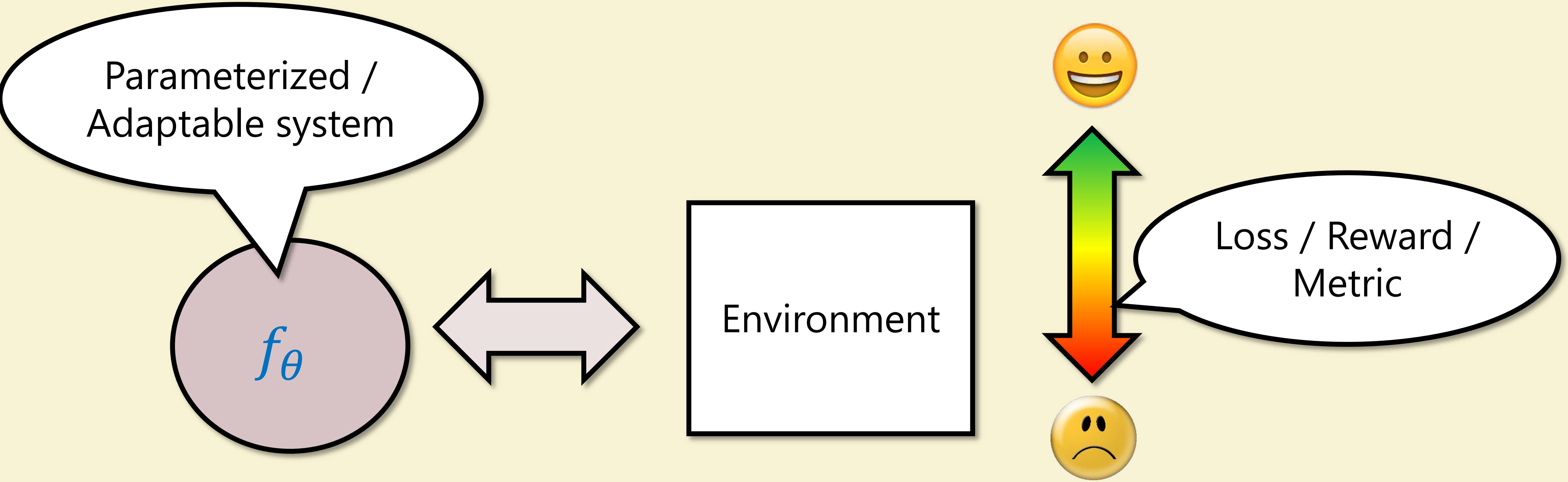
CS 229br: Foundations of Deep Learning

Boaz Barak

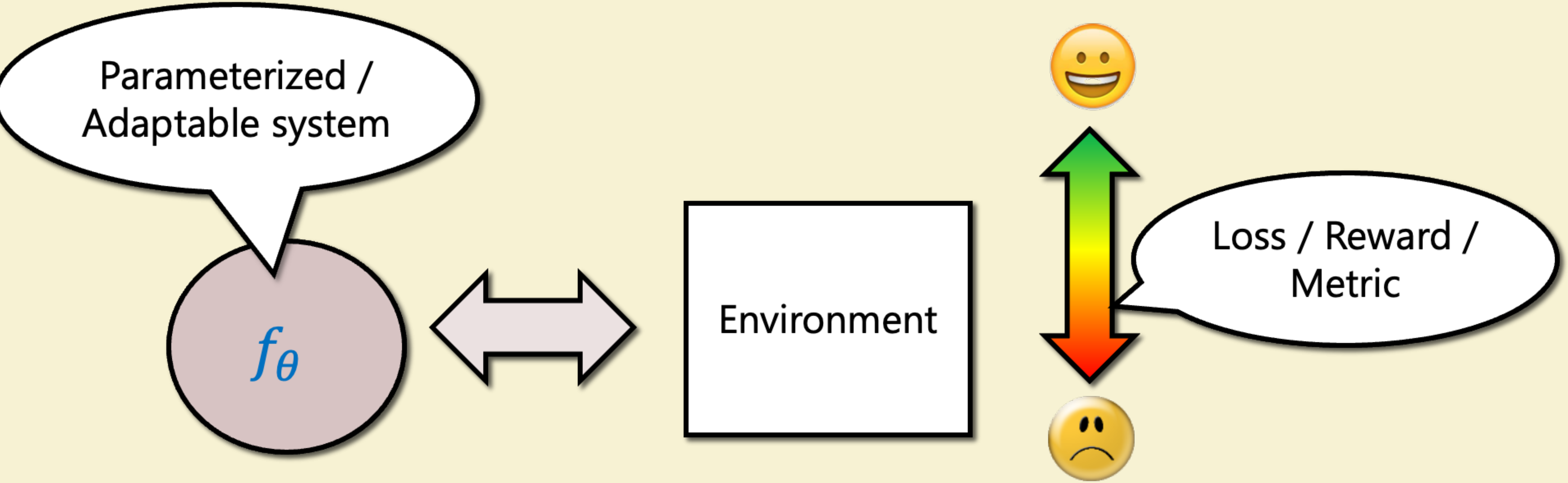
Gustaf Ahdritz

Gal Kaplun

What is learning?



What is learning?



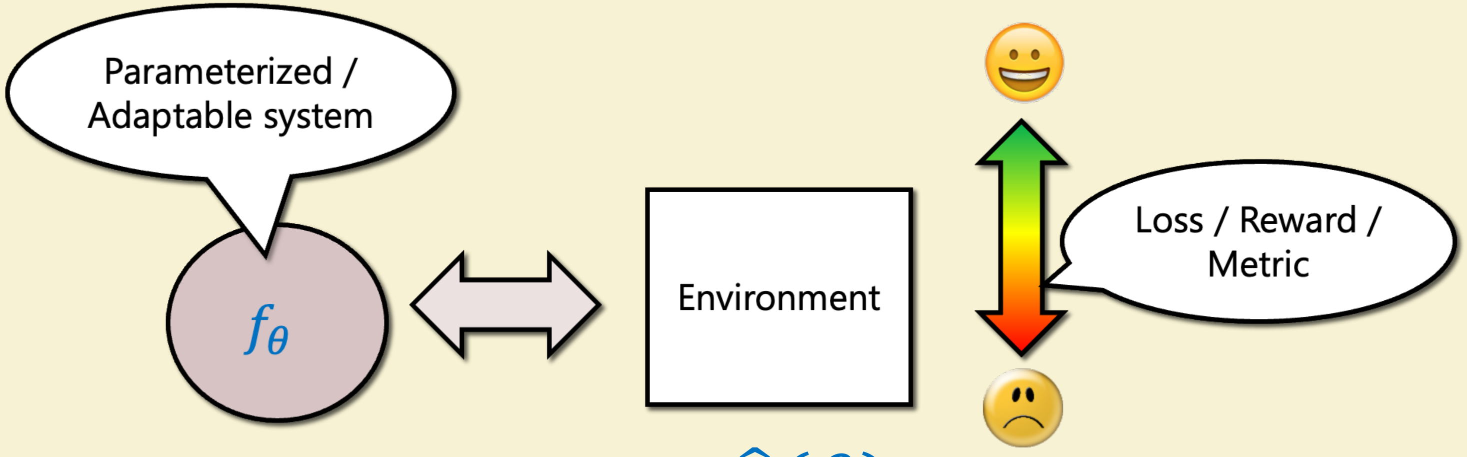
Parameterized / Adaptable system

f_θ

Environment

Loss / Reward / Metric

What is learning?



Traditionally:

Loss: Well specified $L(\theta)$ can compute estimator $\hat{L}(\theta)$

Supervised learning: $L(\theta) = \mathbb{E}[\ell(f_\theta(x), y)]$

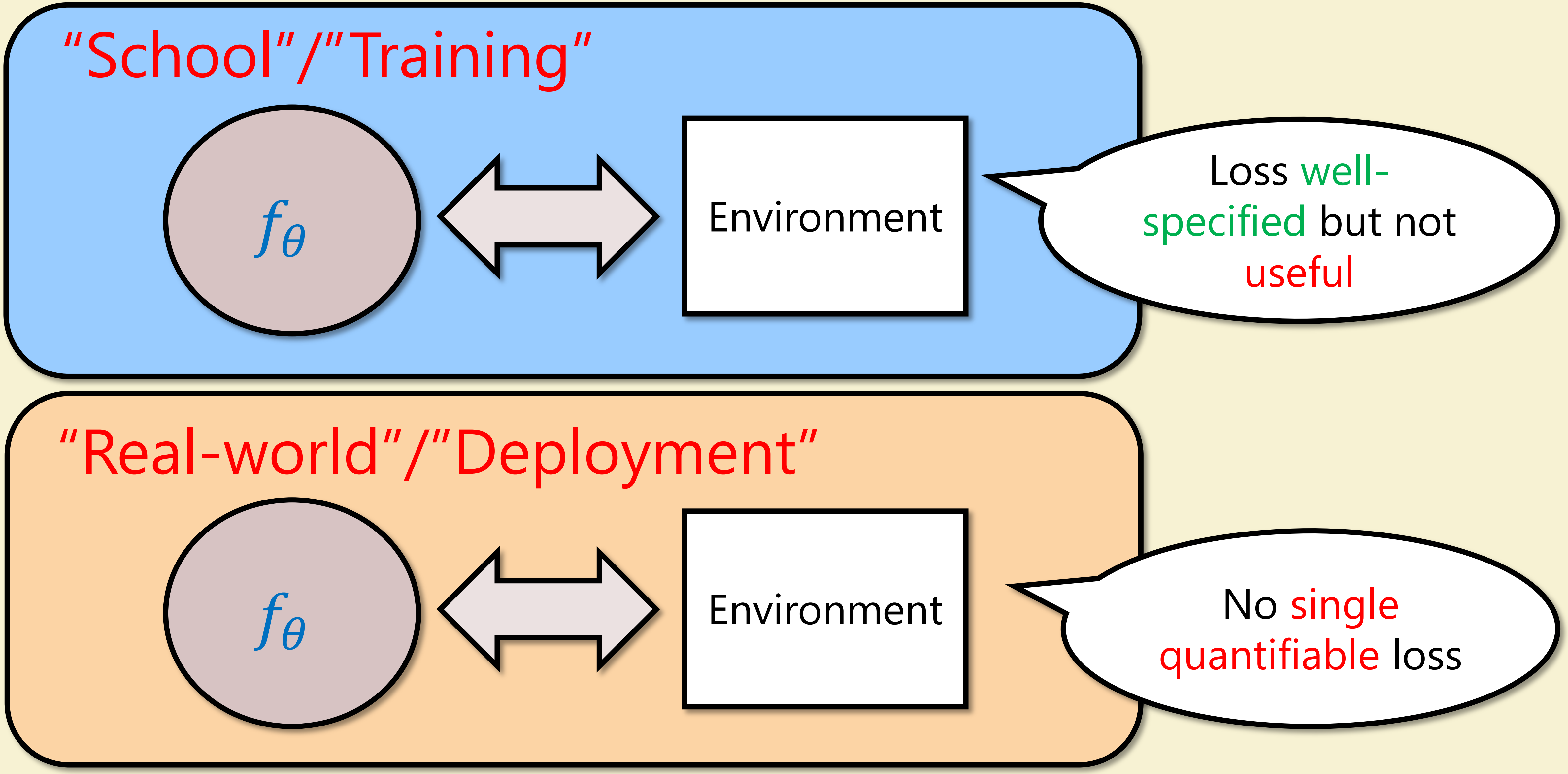
Reinforcement learning: $L(\theta) = -\mathbb{E}\left[\sum_{i=0}^{H} r(s_i)\right]$

Representation: Is there θ with small $\hat{L}(\theta)$?

Optimization: Can we find such θ?

Generalization: Can we guarantee connection of $L(\theta)$ vs $\hat{L}(\theta)$?

What is learning? Modern:



"School"/"Training"

f_θ

Environment

Loss well-specified but not useful

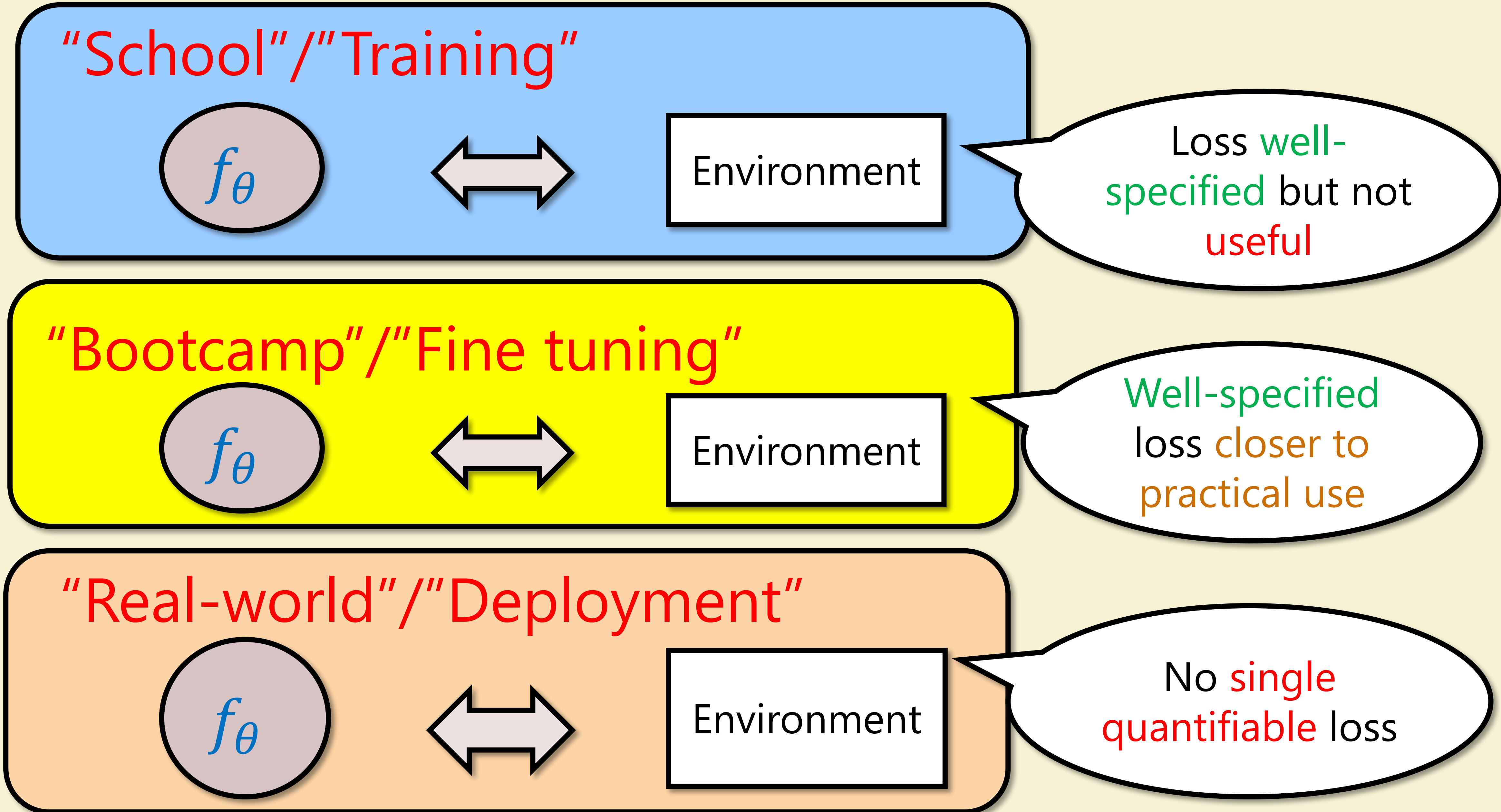
"Real-world"/"Deployment"

f_θ

Environment

No single quantifiable loss

What is learning? Modern:



"School"/"Training"

f_θ

Environment

Loss well-specified but not useful

"Bootcamp"/"Fine tuning"

f_θ

Environment

Well-specified loss closer to practical use

"Real-world"/"Deployment"

f_θ

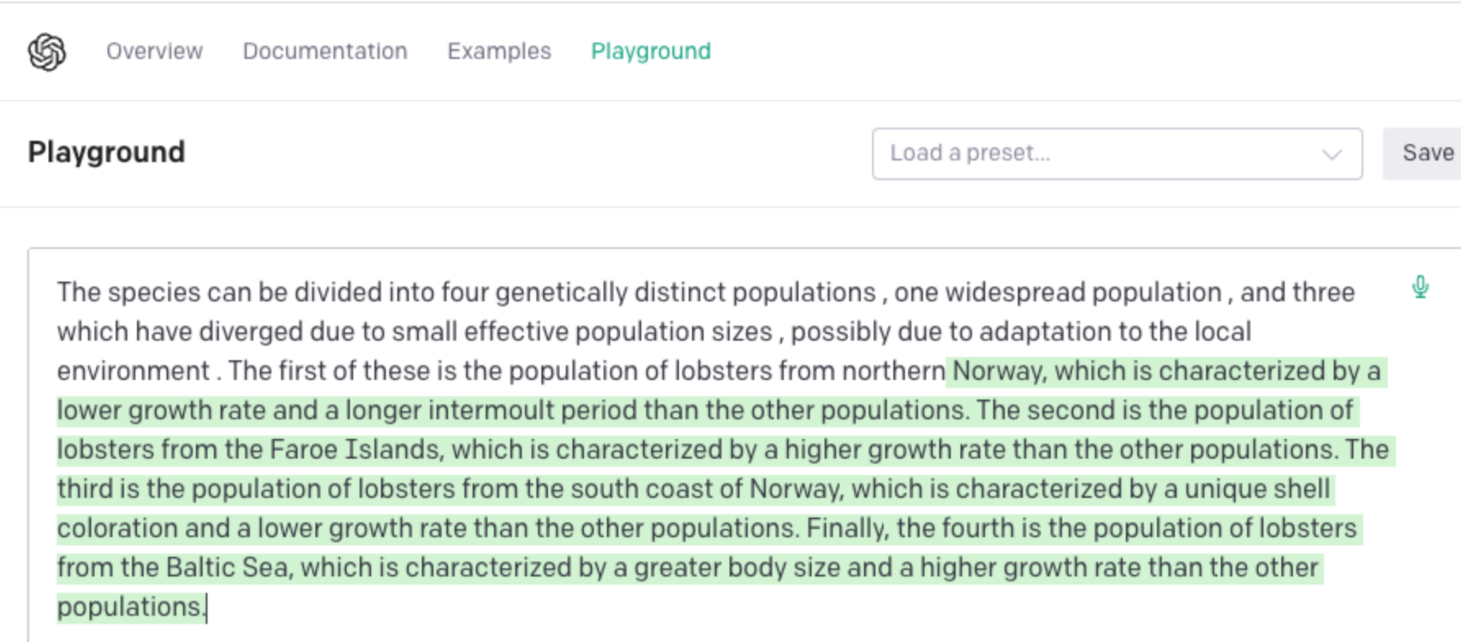
Environment

No single quantifiable loss

Examp

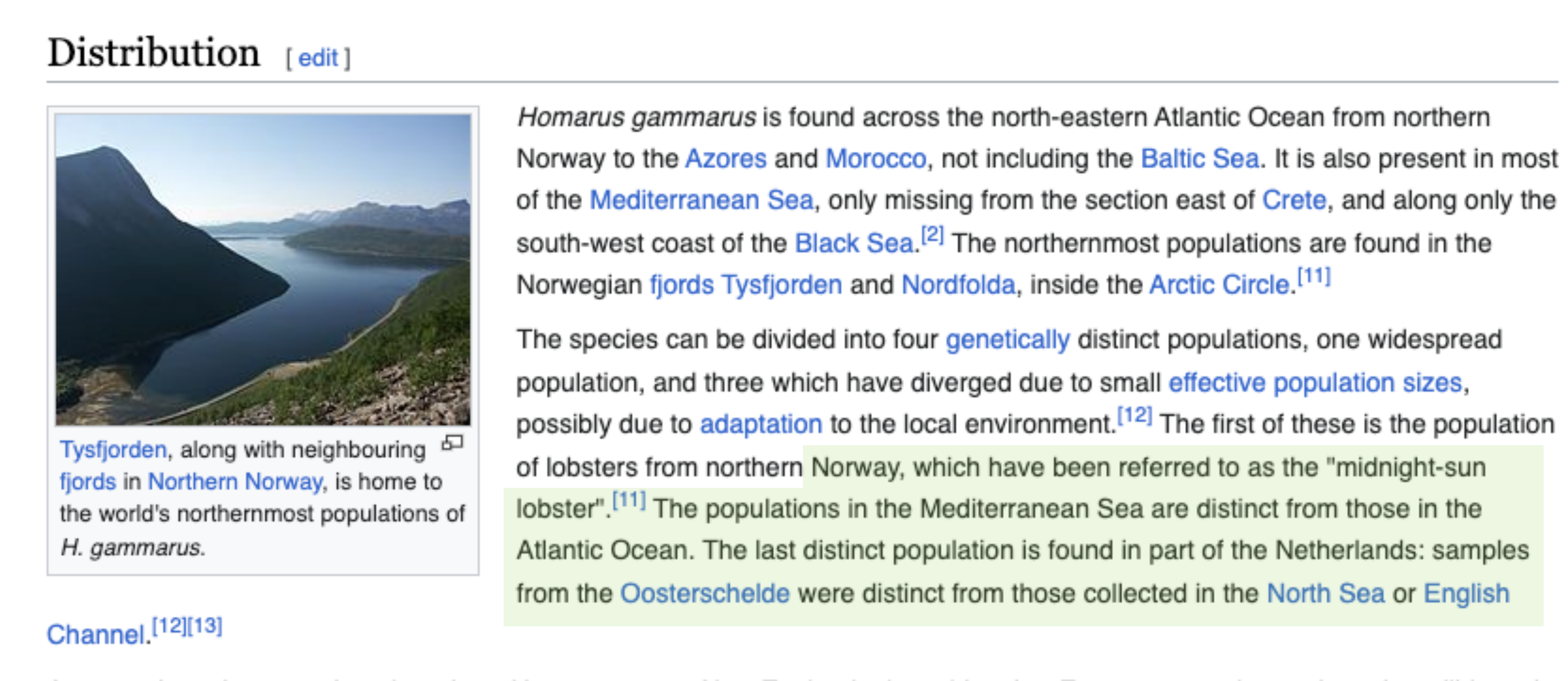
“Basic s

net



The species can be divided into four genetically distinct populations , one widespread population , and three which have diverged due to small effective population sizes , possibly due to adaptation to the local environment . The first of these is the population of lobsters from northern Norway, which is characterized by a lower growth rate and a longer intermoult period than the other populations. The second is the population of lobsters from the Faroe Islands, which is characterized by a higher growth rate than the other populations. The third is the population of lobsters from the south coast of Norway, which is characterized by a unique shell coloration and a lower growth rate than the other populations. Finally, the fourth is the population of lobsters from the Baltic Sea, which is characterized by a greater body size and a higher growth rate than the other populations.

The sp
distin , and
three
popula the
local
popula



Distribution [edit]

Tysfjorden, along with neighbouring fjords in Northern Norway, is home to the world's northernmost populations of *H. gammarus*.

Homarus gammarus is found across the north-eastern Atlantic Ocean from northern Norway to the Azores and Morocco, not including the Baltic Sea. It is also present in most of the Mediterranean Sea, only missing from the section east of Crete, and along only the south-west coast of the Black Sea.[2] The northernmost populations are found in the Norwegian fjords Tysfjorden and Nordfolda, inside the Arctic Circle.[11]

The species can be divided into four genetically distinct populations, one widespread population, and three which have diverged due to small effective population sizes, possibly due to adaptation to the local environment.[12] The first of these is the population of lobsters from northern Norway, which have been referred to as the "midnight-sun lobster".[11] The populations in the Mediterranean Sea are distinct from those in the Atlantic Ocean. The last distinct population is found in part of the Netherlands: samples from the Oosterschelde were distinct from those collected in the North Sea or English Channel.[12][13]

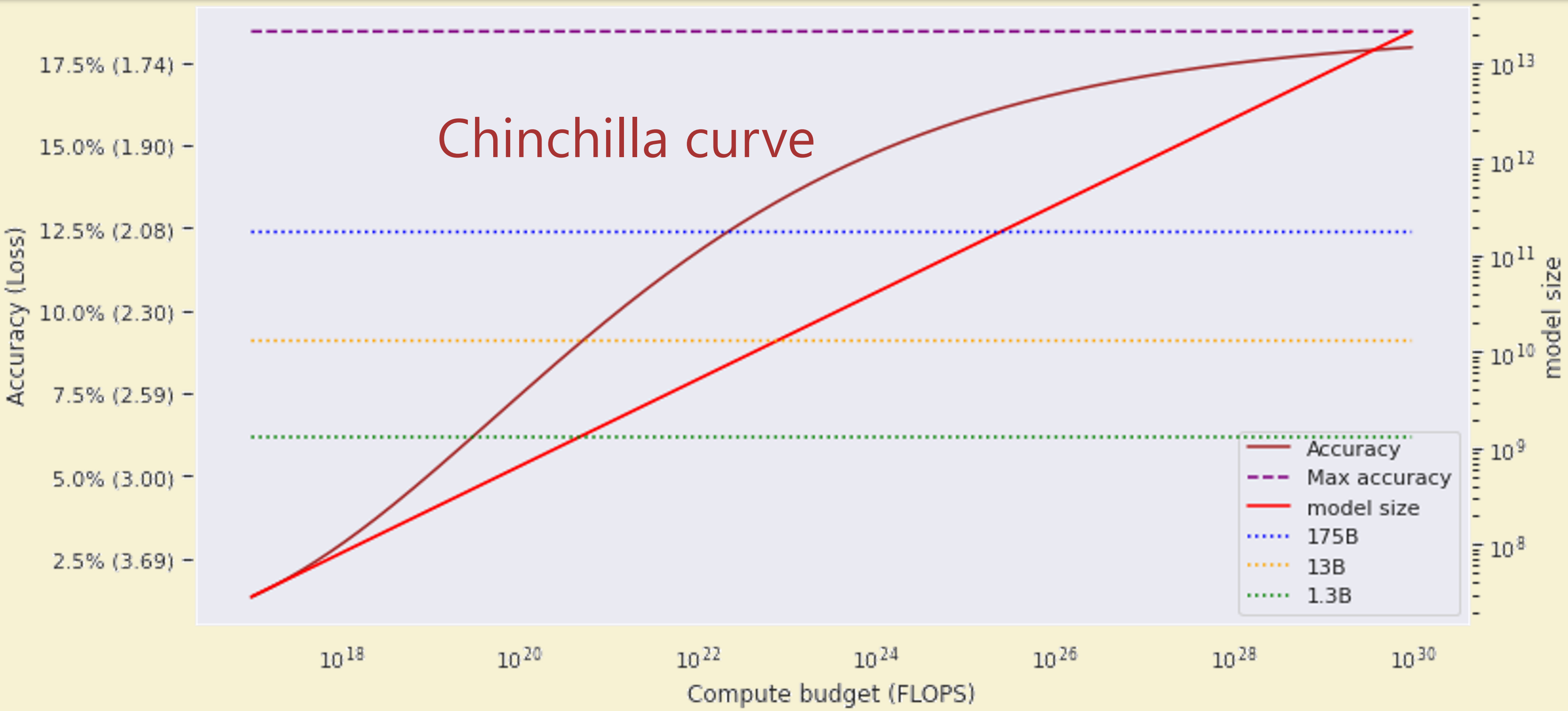
Example: ChatGPT

"Basic schooling": Next-Token Prediction

Input: Random text $(x_1, \ldots, x_n)$ from Internet

Output: (Prob distribution over) token $\hat{x}$

Loss: $-\mathbb{E}[\log \Pr[\hat{x} = x_{n+1}]]]$



Chinchilla curve

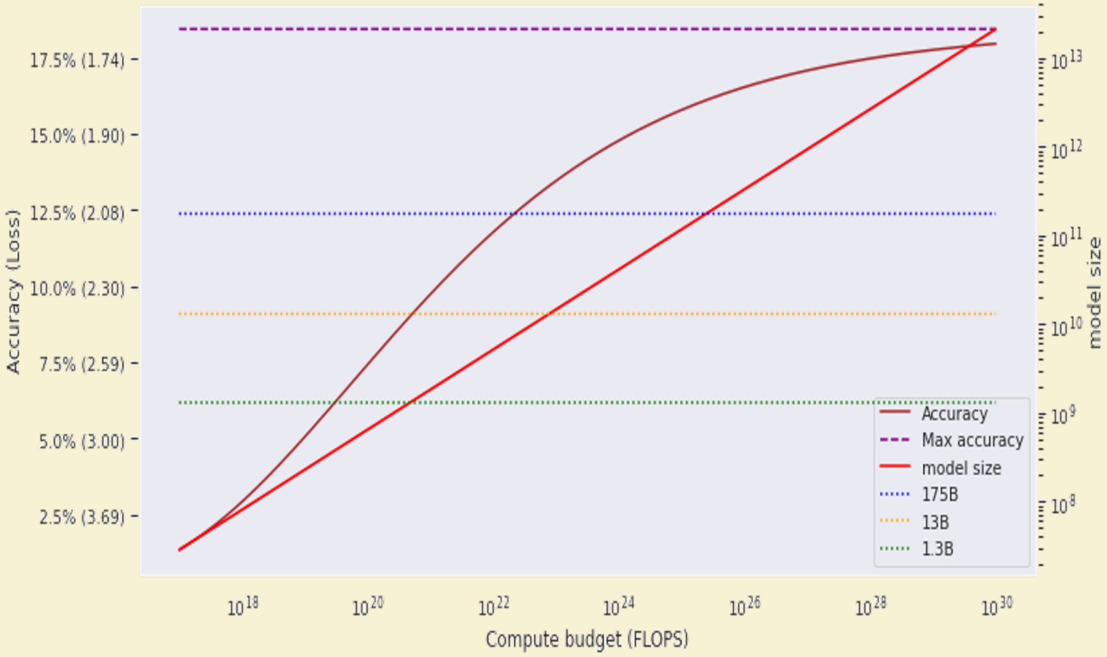
Example: ChatGPT

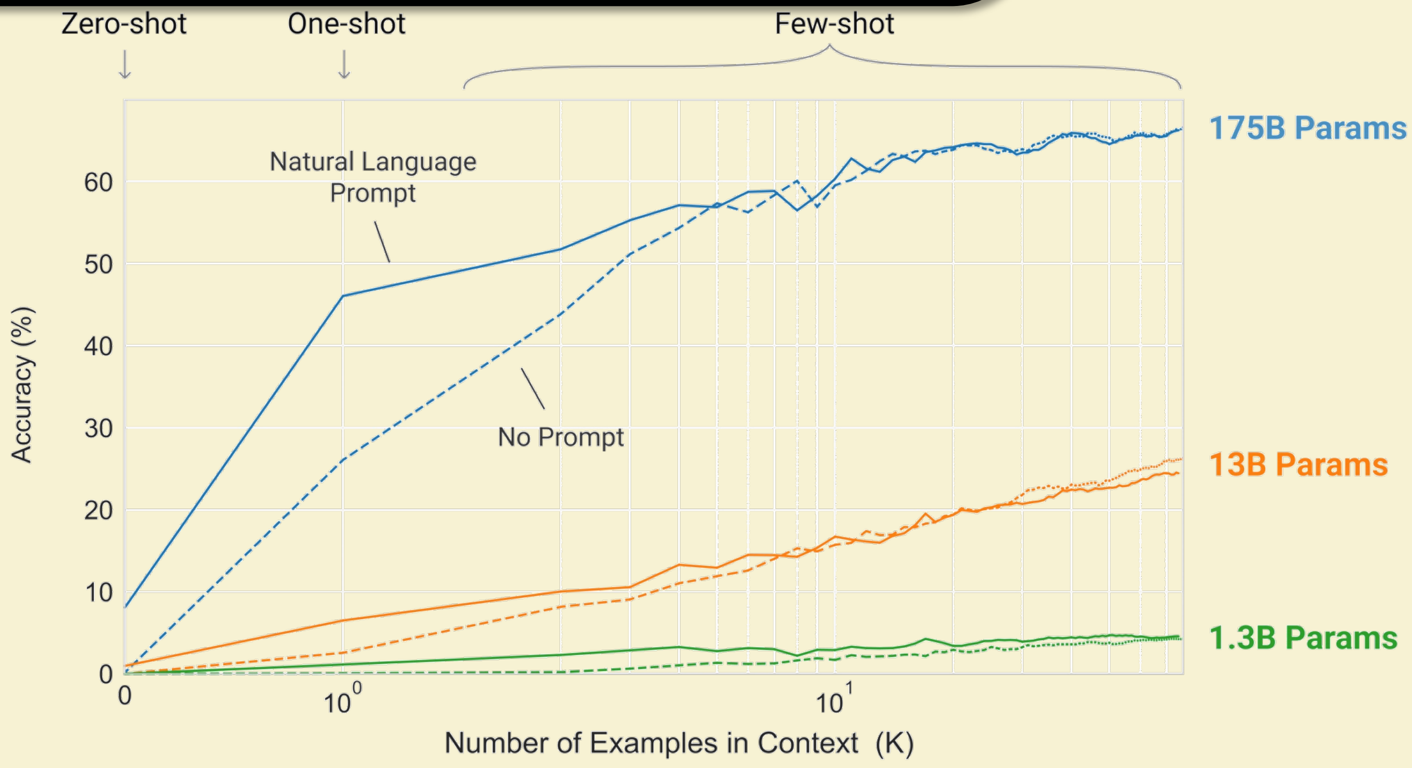
"Basic schooling": Next-Token Prediction

Input: Random text $(x_1, \dots, x_n)$ from Internet

Output: (Prob distribution over) token $\hat{x}$

Loss: $-\mathbb{E}[\log \Pr[\hat{x} = x_{n+1}]]]$





Example: ChatGPT

"Basic schooling": Next-Token Prediction

500B

Input: Random text $(x_1, \dots, x_n)$ from Internet

Output: (Prob distribution over) token $\hat{x}$

Loss: $-\mathbb{E}[\log \Pr[\hat{x} = x_{n+1}]]]$

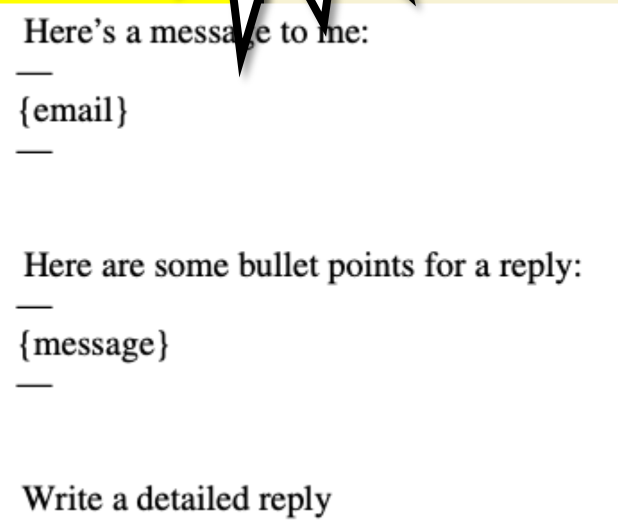
"Bootcamp": Instruct Tuning

40K

Input: Random human-produced prompt

Output: Response

Loss: (learned version of) human rating



Here's a message to me:

—

{email}

—

Here are some bullet points for a reply:

—

{message}

—

Write a detailed reply

This course

- Taste of research results, questions, experiments, and more
- Goal: Get to state of art research:
    - Most lectures include paper from last 2 years
    - Though some also "wisdom of the ancients"
- Very experimental "rough around the edges"
- Lot of learning on your own and from each other
- Hope: Very interactive – in lectures and on slack

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

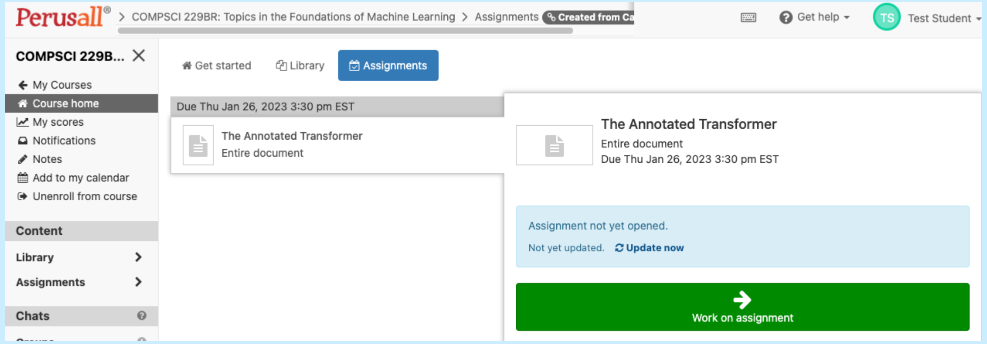
Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Student expectations



Due Thu Jan 26, 2023 3:30 pm EST

The Annotated Transformer
Entire document

The Annotated Transformer
Entire document
Due Thu Jan 26, 2023 3:30 pm EST

Assignment not yet opened.

Not yet updated.

Not set in stone but will include:

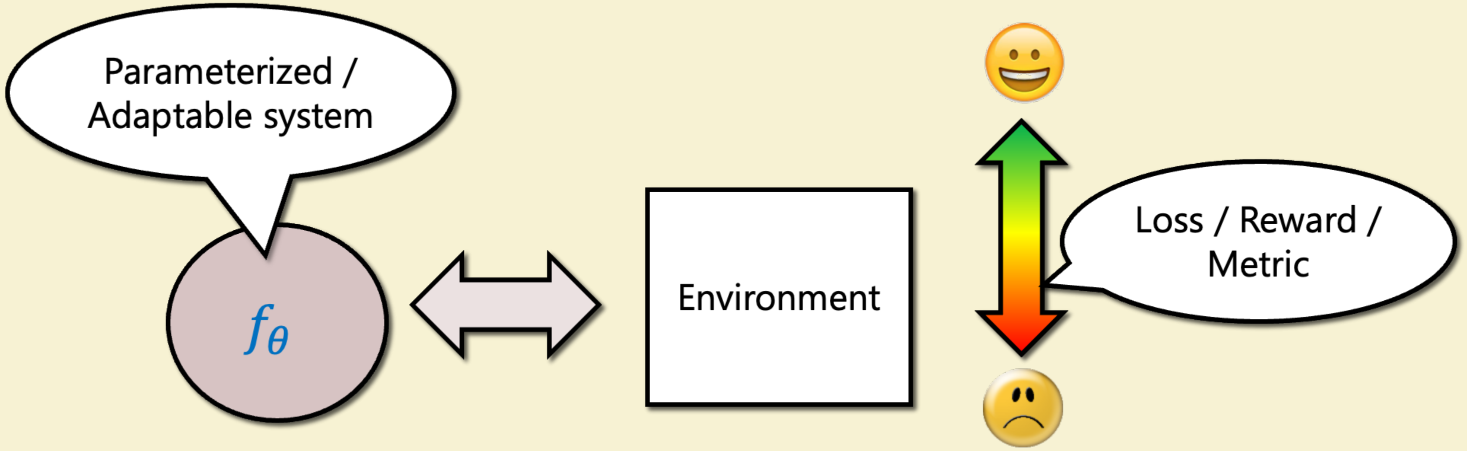
- Pre-reading before lectures
- Applied & theoretical problem sets

  Note: *Lectures will not teach practical skills – rely on students to pick up using suggested tutorials, other resources, and each other.*

  TFs happy to answer questions!
- (Possibly) scribe notes
- Projects – self chosen and directed.
- No midterm or final

Grading: We'll figure out some grade – hope that's not your loss function ☺

What is learning?



Parameterized / Adaptable system

f_θ

Environment

Loss / Reward / Metric

Traditionally:

Loss: Well specified $L(\theta)$ can compute estimator $\hat{L}(\theta)$

Supervised learning: $L(\theta) = \mathbb{E}[\ell(f_\theta(x), y)]$

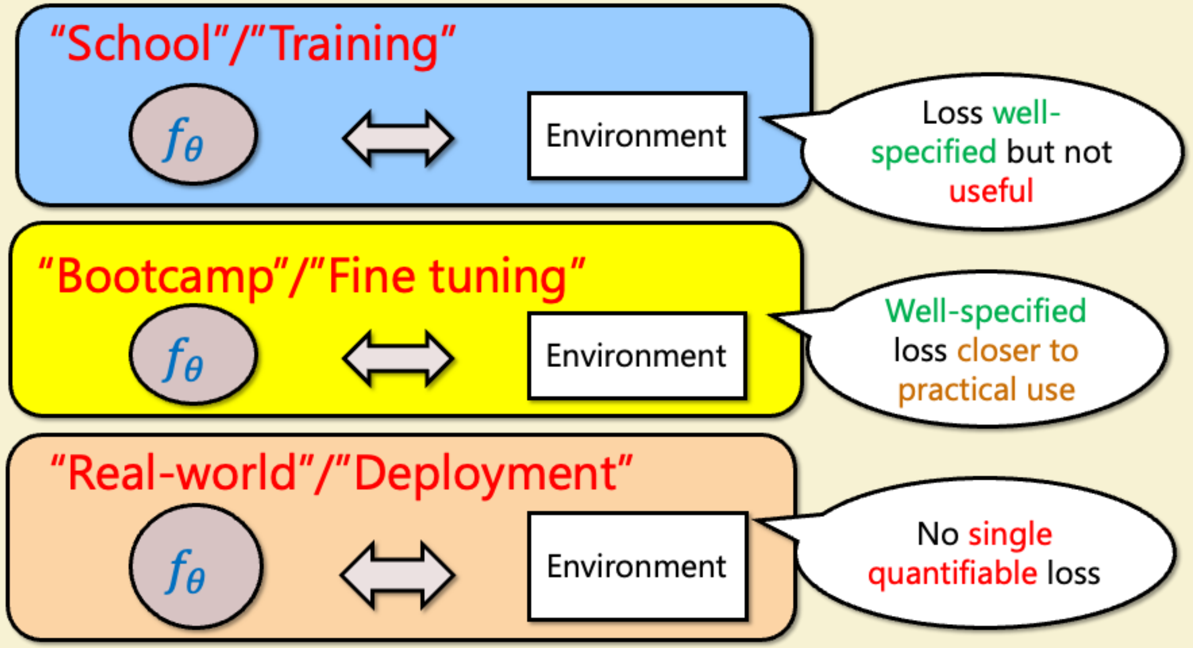
Reinforcement learning: $L(\theta) = -\mathbb{E}[\sum_{i=0}^{H} r(s_i)]$

Representation: Is there θ with small $\hat{L}(\theta)$?

Optimization: Can we find such θ?

Generalization: Can we guarantee connection of $L(\theta)$ vs $\hat{L}(\theta)$?

Modern:



“School”/”Training”

f_θ

Environment

Loss well-specified but not useful

“Bootcamp”/”Fine tuning”

f_θ

Environment

Well-specified loss closer to practical use

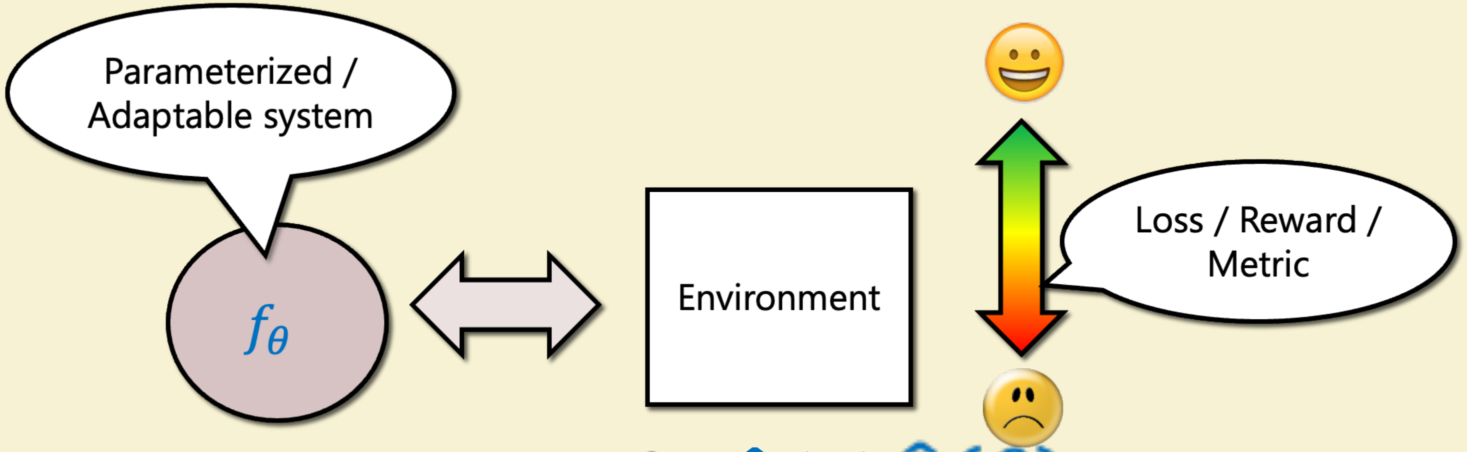
“Real-world”/”Deployment”

f_θ

Environment

No single quantifiable loss

What is learning?



Traditionally:

Loss: Well specified $L(\theta)$ can compute estimator $\hat{L}(\theta)$

Loss: Well specified $L(\theta)$ can compute estimator $\hat{L}(\theta)$

Supervised learning: $L(\theta) = \mathbb{E}[\ell(f_\theta(x), y)]$

Supervised learning: $L(\theta) = \mathbb{E}[\ell(f_\theta(x), y)]$

Reinforcement learning: $L(\theta) = \mathbb{E}[\sum_{i=0}^{H} r(s_i)]$

Reinforcement learning: $L(\theta) = \mathbb{E}[\sum_{i=0}^{H} r(s_i)]$

Representation: Is there θ with small $\hat{L}(\theta)$?

Optimization: Can we find such θ?

Generalization: Can we guarantee connection of $L(\theta)$ vs $\hat{L}(\theta)$?

Representation: Is there θ with small $\hat{L}(\theta)$?

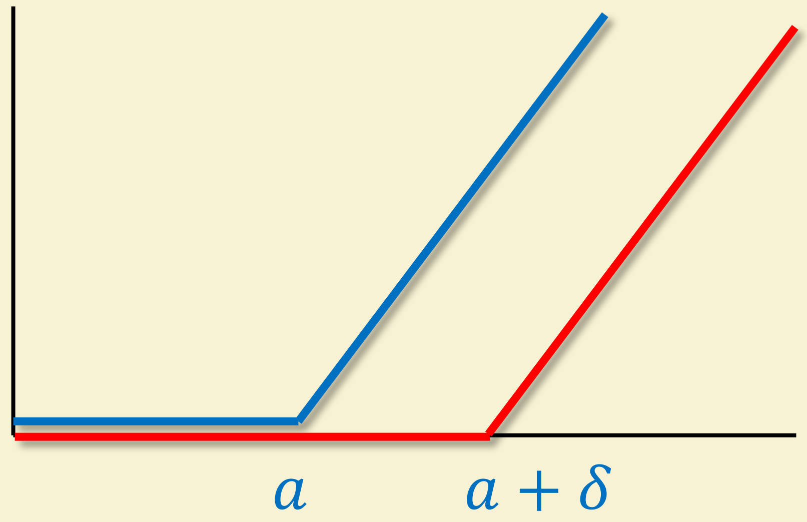
Every continuous $f\colon [0,1] \to \mathbb{R}$ can be arbitrarily approximated by g of form

$$g(x) \ = \sum \alpha_i \, ReLU(\beta_i x + \gamma_i)$$

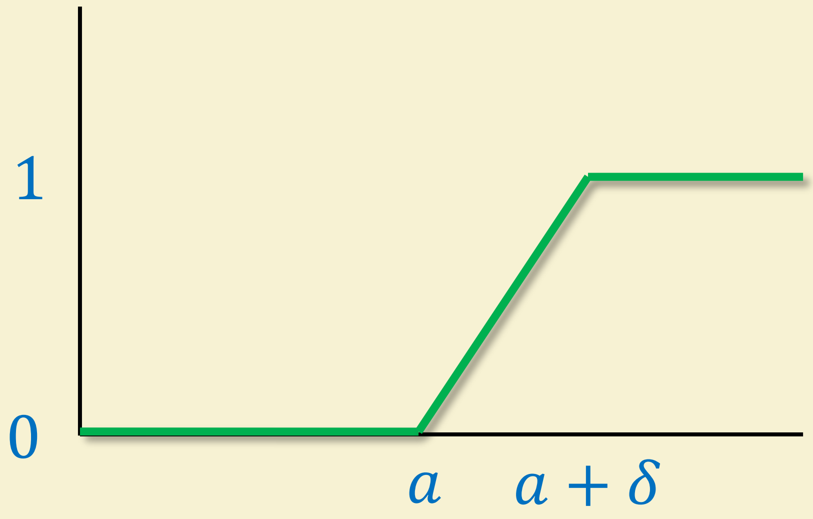
Proof by picture:

$g_a(x) = ReLU(x/\delta - a)$

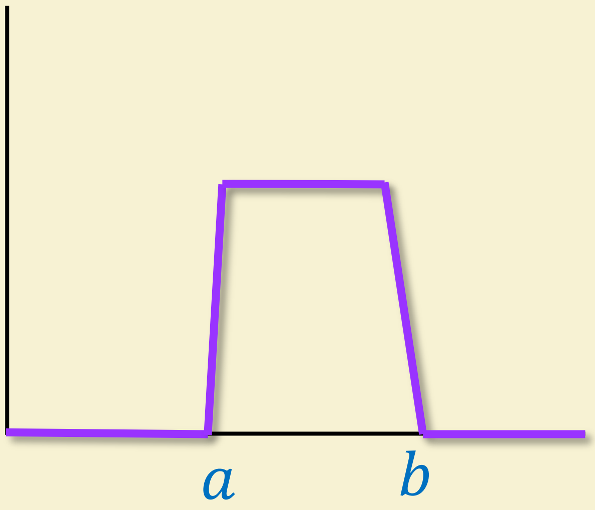
$g_{a+\delta}(x) = ReLU(x/\delta - a - \delta)$



$h_a = g_a - g_{a+\delta}$



$I_{a,b} = h_a - h_b$



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

Representation: Is there θ with small $\hat{L}(\theta)$?

Every continuous $f: [0,1] \to \mathbb{R}$ can be arbitrarily approximated by g of form

$$g(x) \;\; = \sum \alpha_i \, ReLU(\beta_i x + \gamma_i)$$

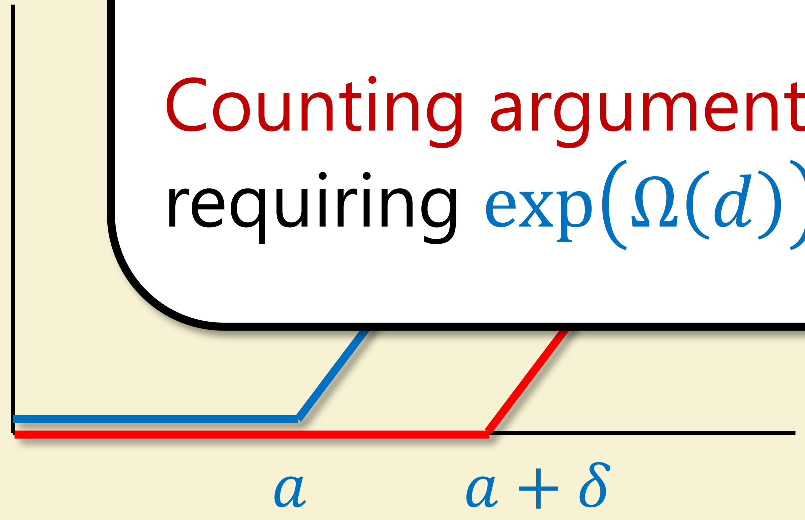
Proo

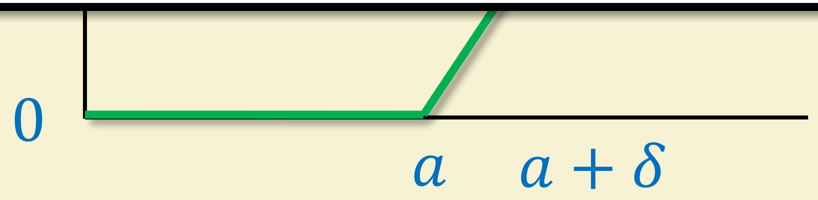
$g_a($

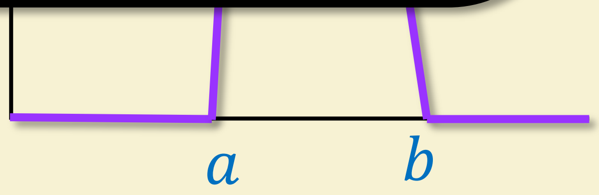
g_{a+}

Approximation Theorem: Every smooth function $F: \mathbb{R}^d \to [0,1]$ can be ϵ-approximated as a sum of $(1/\epsilon)^{O(d)}$ ReLUs. (depth 2 neural network)

Counting argument: For every architecture, $\exists$ smooth $F: \mathbb{R}^d \to [0,1]$ requiring $\exp\big(\Omega(d)\big)$ Neurons to 0.1 approximate F







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

Optimization: Can we find such θ?

Gradient Descent

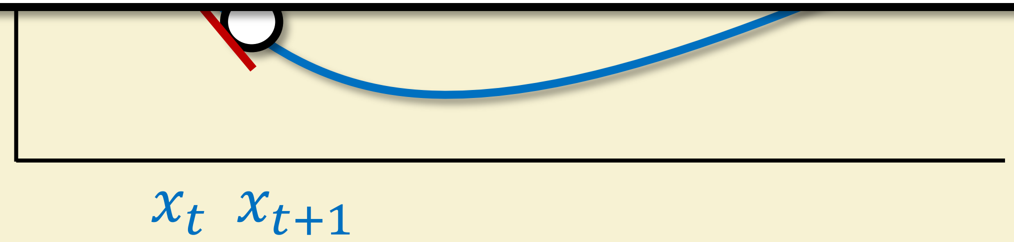
$$x_{t+1} = x_t - \eta f'(x_t)$$

Dimension d:

$f'(x) \rightarrow \nabla f(x) \in \mathbb{R}^d$

$f''(x) \rightarrow H_f(\mathrm{x}) = \nabla_2 f(x) \in \mathbb{R}^{d \times d}$ (psd)

If $\eta \lesssim 2/\lambda_d$ drop by $\sim \frac{\lambda_1}{\lambda_d} \|\nabla\|^2$



$\delta = -\eta f'(x_t)$

$$f(x_t + \delta) \approx f(x_t) + \delta f'(x_t) + \frac{\delta^2}{2} f''(x_t)$$

$$f(x_{t+1}) \approx f(x_t) - \eta f'(x_t)^2 + \frac{\eta^2 f'(x_t)^2}{2} f''(x_t) \quad = f(x_t) - \eta f'(x_t)^2 (1 - \frac{\eta f''(x_t)}{2})$$

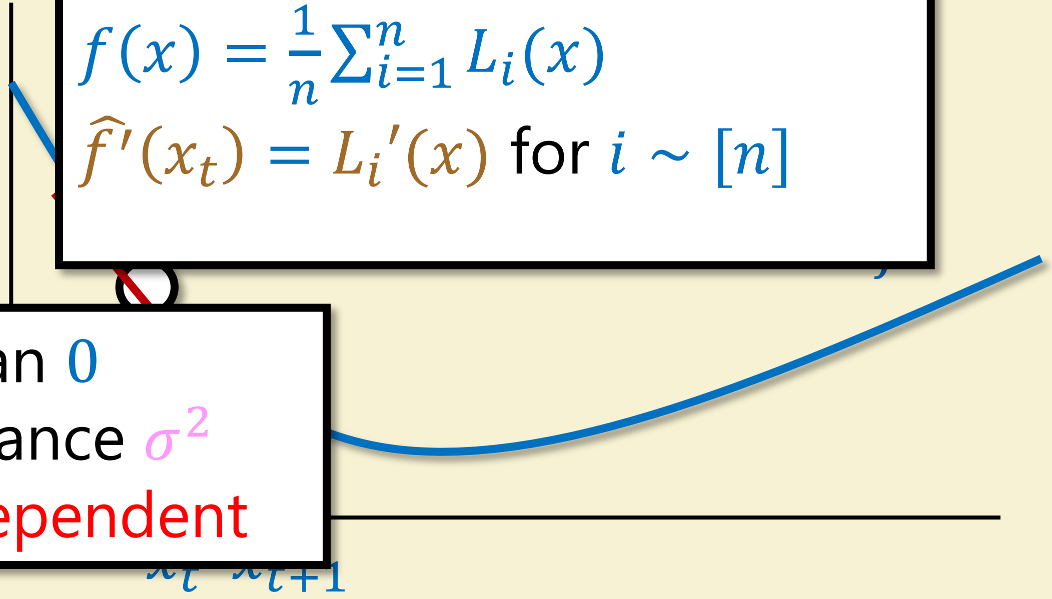
- If $\eta < 2/f''(x_t)$ then make progress
- If $\eta \sim 1/f''(x_t)$ then drop by $\sim f'(x_t)^2/f''(x_t)$

Stochastic Gradient Descent

$$x_{t+1} = x_t - \eta \widehat{f}'(x_t)$$

$$\mathbb{E}[\widehat{f}'(x)] = f'(x_t), V[\widehat{f}'(x)] = \sigma^2$$

Assume $\widehat{f}'(x) = f'(x) + N$



In Machine Learning:

$f(x) = \frac{1}{n}\sum_{i=1}^{n} L_i(x)$

$\widehat{f}'(x_t) = L_i{}'(x)$ for $i \sim [n]$

Mean 0

Variance σ^2

Independent

$$f(x_t + \delta) \approx f(x_t) + \delta f'(x_t) + \frac{\delta^2}{2} f''(x_t)$$

$$f(x_{t+1}) \approx f(x_t) - \eta f'(x_t)^2 \left(1 - \frac{\eta f''(x_t)}{2}\right) + \eta^2 \sigma^2 f''(x_t)$$

- If $\eta < 2/f''(x_t)$ and (*) $\eta\sigma^2 \ll f'(x_t)^2/f''(x)$ then make progress
- If $\eta \sim 1/f''(x_t)$ and (*) then drop by $\sim f'(x_t)^2/f''(x_t)$

Generalization: Can we guarantee

Empirical Risk Minimization (ERM):

$$A(S) = \arg\min_{f \in \mathcal{F}} \hat{\mathcal{L}}_S(f)$$

$S = (x_i, y_i)_{i=1..n}$

Learning Algorithm

A

$f \in \mathcal{F}$

Population 0-1 loss $\mathcal{L}(f) = \Pr[f(X) \neq Y]$

Empirical 0-1 loss $\hat{\mathcal{L}}_S(f) = \frac{1}{n}\sum_{i=1}^{n} 1_{f(x_i)\neq y_i}$

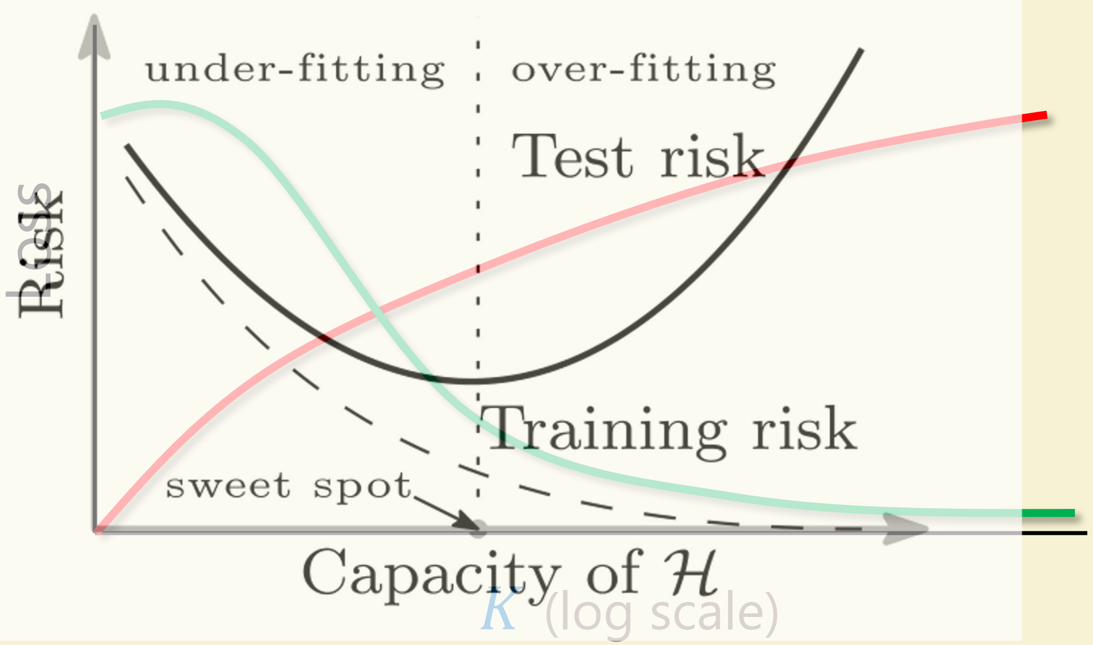
$K \approx \exp(\#\, params)$

bias

variance

Assume $\mathcal{F}_K = \{ f_1, \ldots, f_K \}$

$$\hat{\mathcal{L}}(f_i) = \mathcal{L}(f_i) + N(0, 1/n)$$

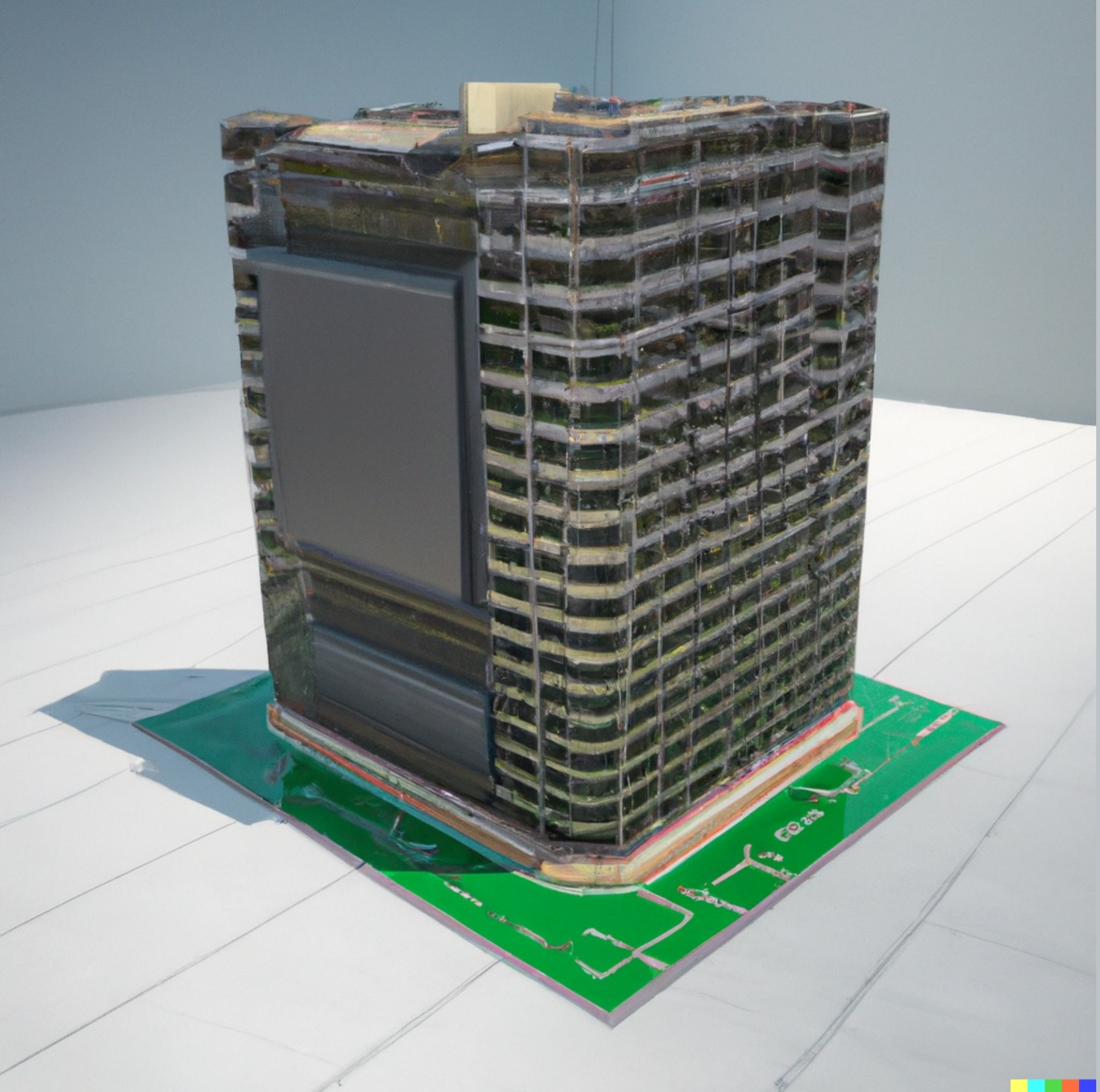


$$\max_{1\le i\le k} N_i(0,1/n) \approx \sqrt{\frac{\log K}{n}}$$

Pessimistic bound

$$\min_{1\le i\le k} \mathcal{L}(f_i) \approx (\log K)^{-\alpha}?$$

“Scaling laws”



Part II: Architecture

Why architecture?

Inductive bias: “Hard wire” prior knowledge to use less data.

Execution efficiency: Find architectures that use smaller number of total operations or better match of operations to the hardware.

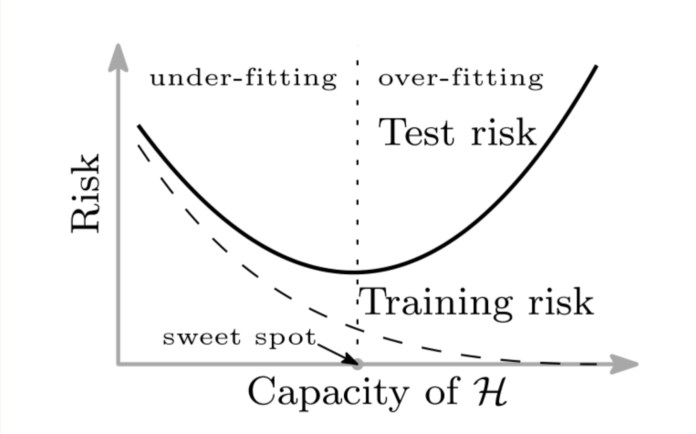
Training efficiency: Find architectures that are a good match to the optimization algorithm (e.g., gradient descent).

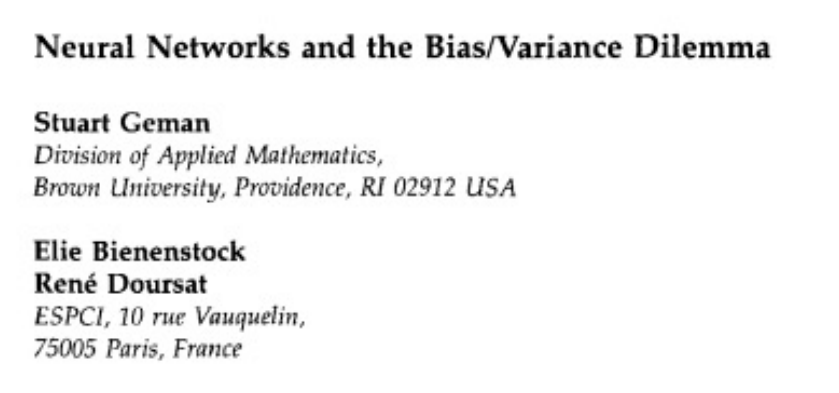
What else?

Inductive bias: “Hard wire” prior knowledge to use less data.

“No Free Lunch”

Is inductive bias everything?





Neural Networks and the Bias/Variance Dilemma

Stuart Geman
Division of Applied Mathematics,
Brown University, Providence, RI 02912 USA

Elie Bienenstock
René Doursat
ESPCI, 10 rue Vauquelin,
75005 Paris, France

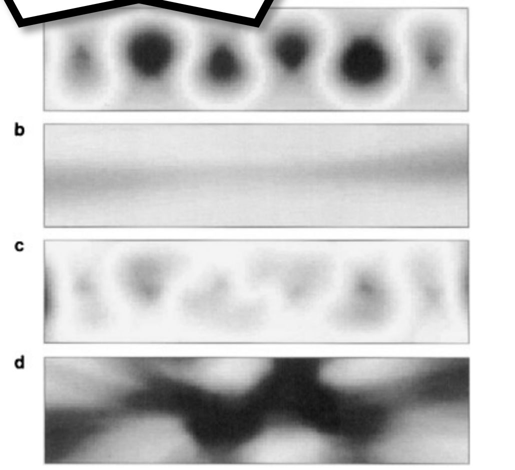
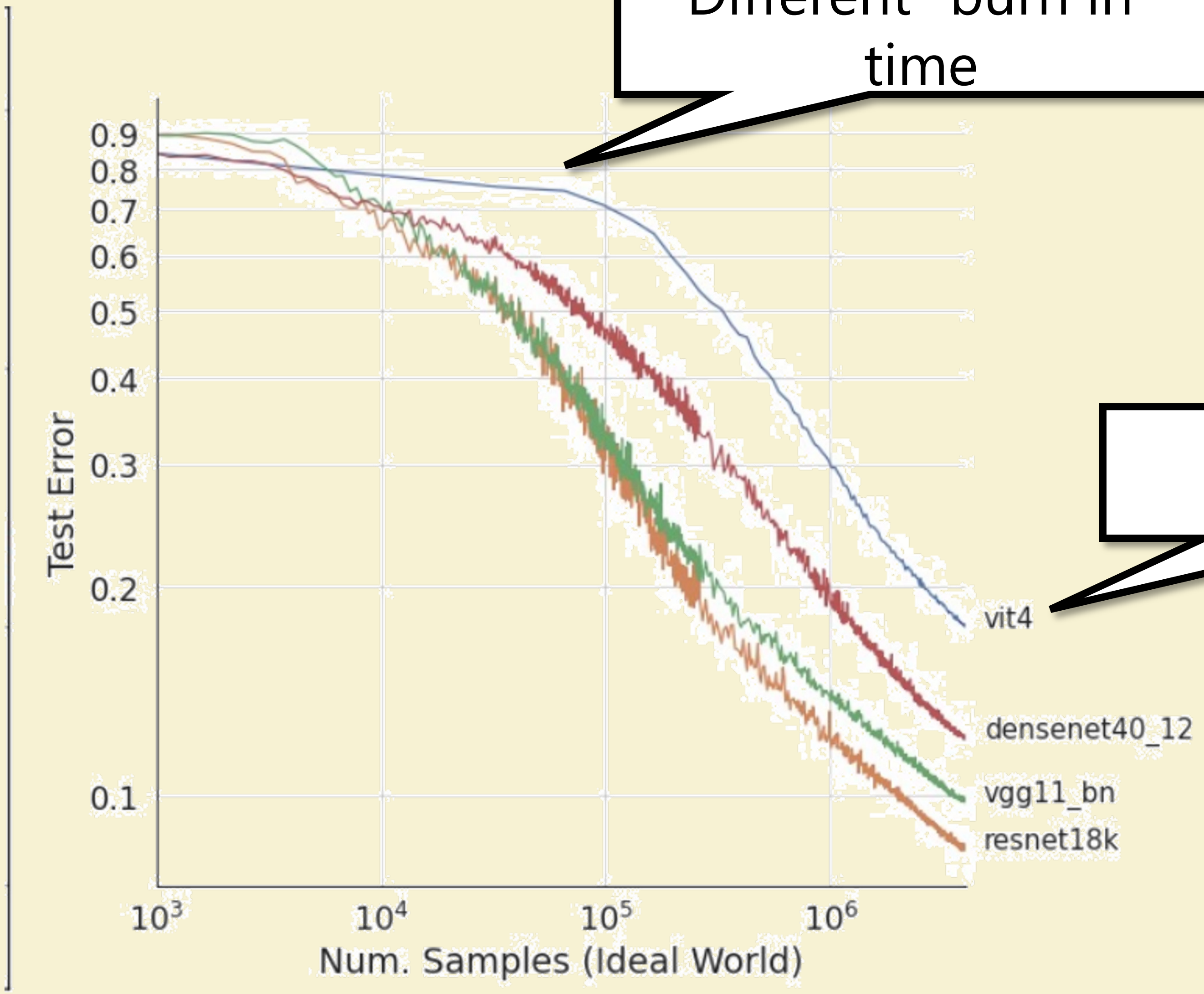


Figure 11: Bias and variance of single-hidden-unit and 15-hidden-unit feedforward neural networks, as functions of input vector. Regression surface is depicted in Figure 3b. Scale is by gray levels, running from largest values, coded in black, to zero, coded in white. (a) Bias of single-hidden-unit machine. (b) Variance of single-hidden-unit machine. (c) Bias of 15-hidden-unit machine. (d) Variance of 15-hidden-unit machine. Bias decreases and variance increases with the addition of hidden units.

“To mimic substantial human behavior … will require complex machinery. Inferring this complexity from examples .. [is] not feasible: too many examples would be needed. Important properties must be built-in or “hard-wired,” ”

Of course most neural modelers do not take tabula rasa architectures as serious models of the nervous system … identifying the right “preconditions” is the substantial problem in neural modeling. … categorization must be largely built in

Is inductive bias nothing? everything?

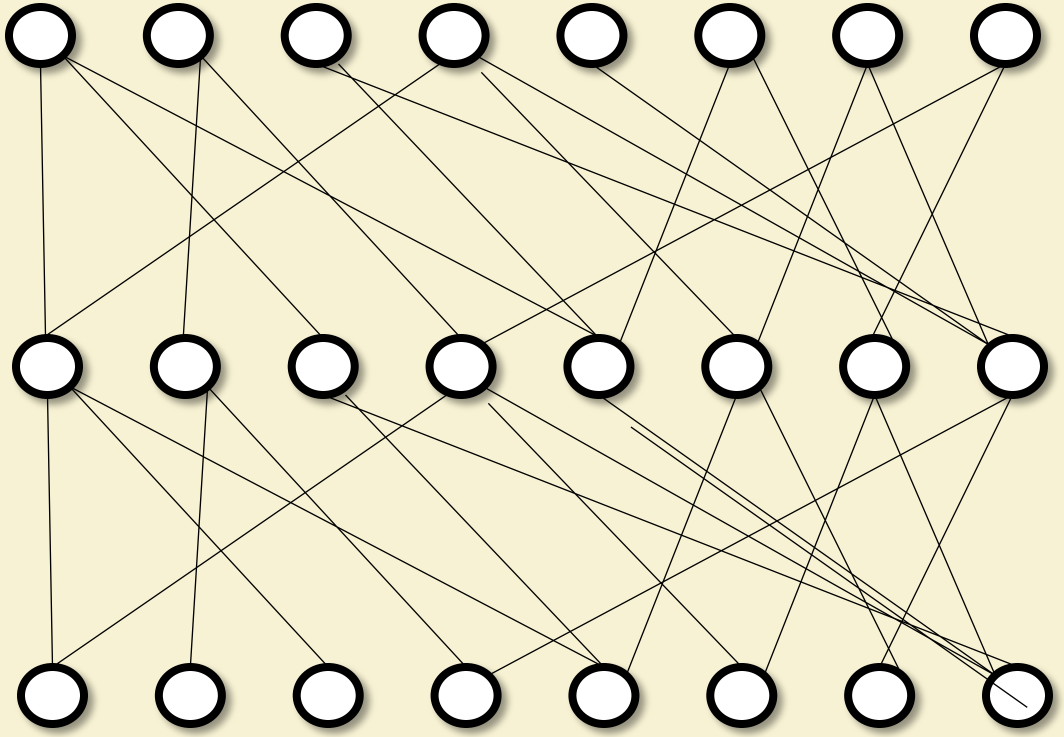


Different “burn in” time

Same exponent

Nakkiran

"No inductive bias": Boolean Circuits

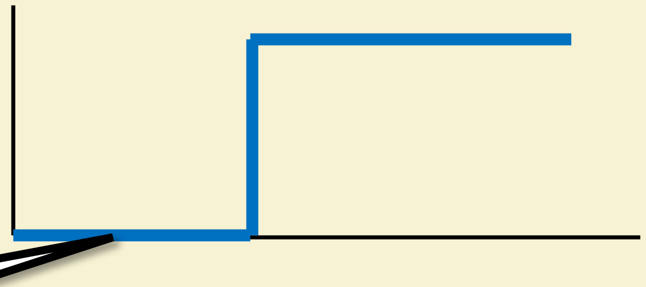


Gates: AND/OR/NOT

$x_1 \vee \overline{x}_2 \vee x_3$

$x_1 + (1 - x_2) + x_3 \geq 1$

Special case of Threshold function

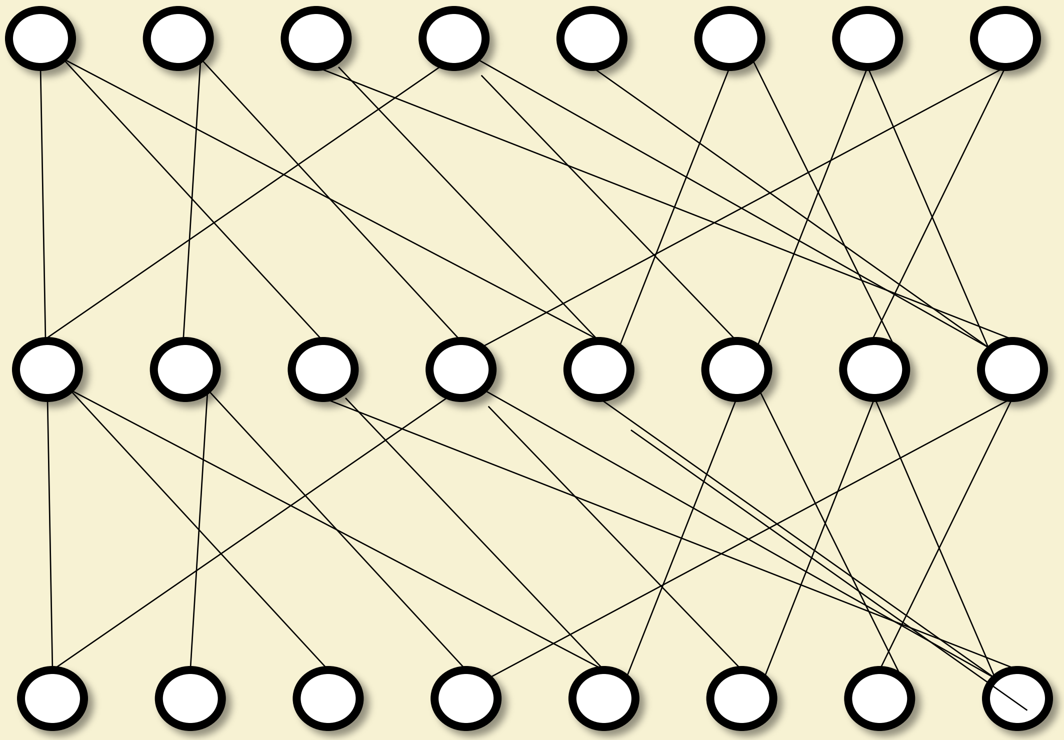


Non differentiable

Destroys training efficiency

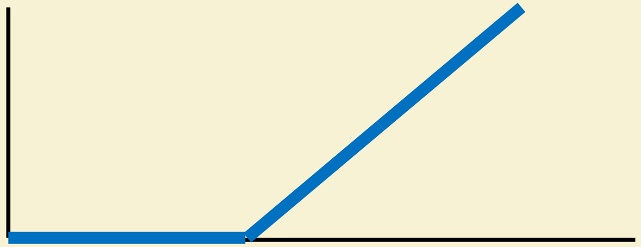
$$f_{w,b}(x) = \begin{cases} 1, & \langle x, w \rangle > b \\ 0, & \langle x, w \rangle \leq b \end{cases}$$

"No inductive bias": Boolean Circuits MLP



Units: ReLUs

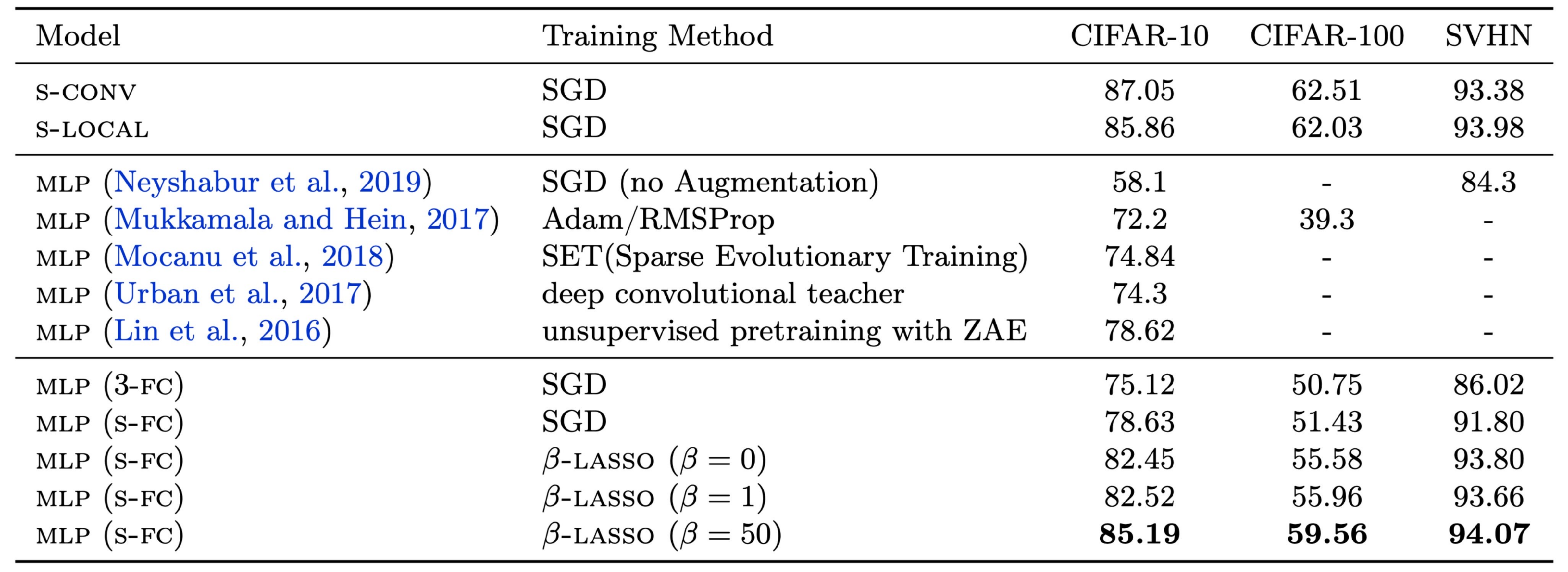
(or other non-linearities)



$\langle x, w\rangle - b$

$$f_{w,b}(x) = \begin{cases} \cancel{1}, & \langle x, w\rangle > b \\ 0, & \langle x, w\rangle \le b \end{cases}$$

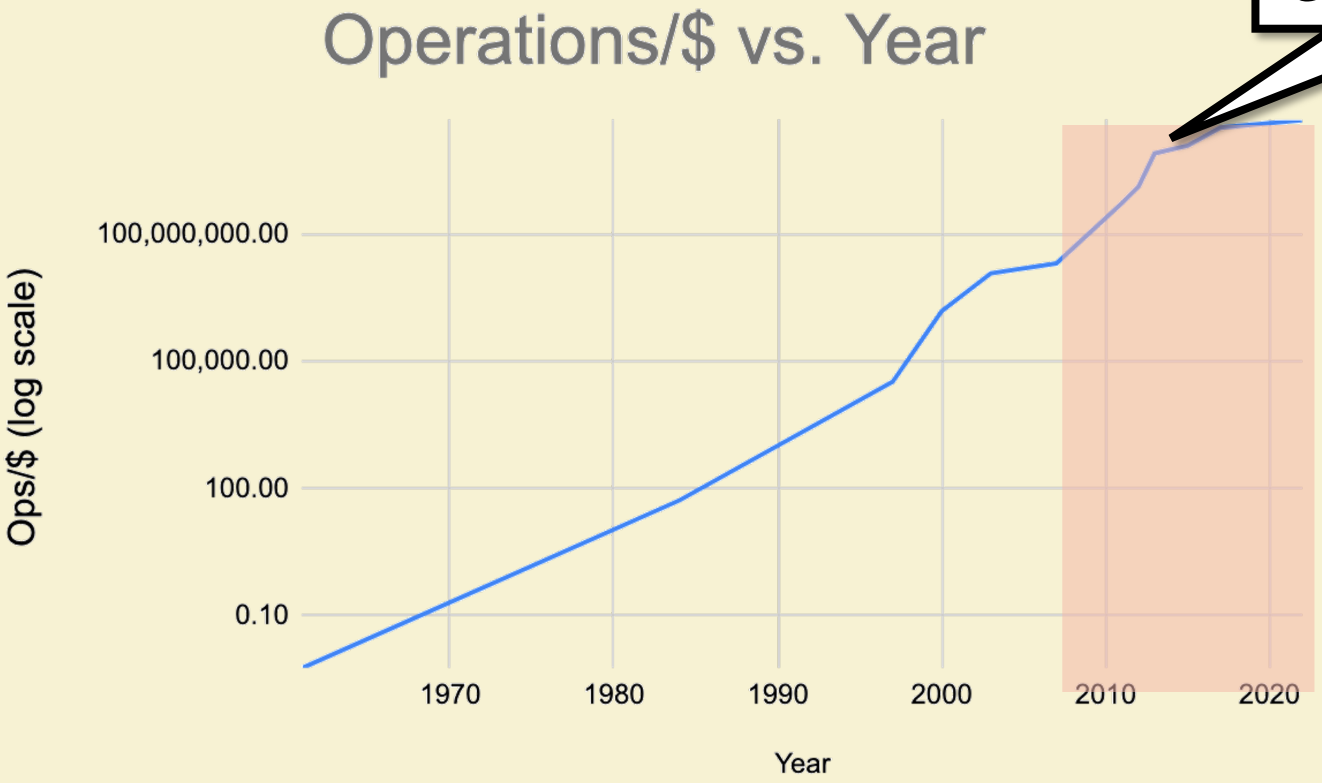
Intuition: Sparsity is all you need



Model	Training Method	CIFAR-10	CIFAR-100	SVHN
S-CONV	SGD	87.05	62.51	93.38
S-LOCAL	SGD	85.86	62.03	93.98
MLP (Neyshabur et al., 2019)	SGD (no Augmentation)	58.1	-	84.3
MLP (Mukkamala and Hein, 2017)	Adam/RMSProp	72.2	39.3	-
MLP (Mocanu et al., 2018)	SET(Sparse Evolutionary Training)	74.84	-	-
MLP (Urban et al., 2017)	deep convolutional teacher	74.3	-	-
MLP (Lin et al., 2016)	unsupervised pretraining with ZAE	78.62	-	-
MLP (3-FC)	SGD	75.12	50.75	86.02
MLP (S-FC)	SGD	78.63	51.43	91.80
MLP (S-FC)	β-LASSO ($\beta = 0$)	82.45	55.58	93.80
MLP (S-FC)	β-LASSO ($\beta = 1$)	82.52	55.96	93.66
MLP (S-FC)	β-LASSO ($\beta = 50$)	**85.19**	**59.56**	**94.07**

Neyshabur 2020

Execution efficiency: Find architectures that use smaller number of total operations or better match of operations to the hardware.

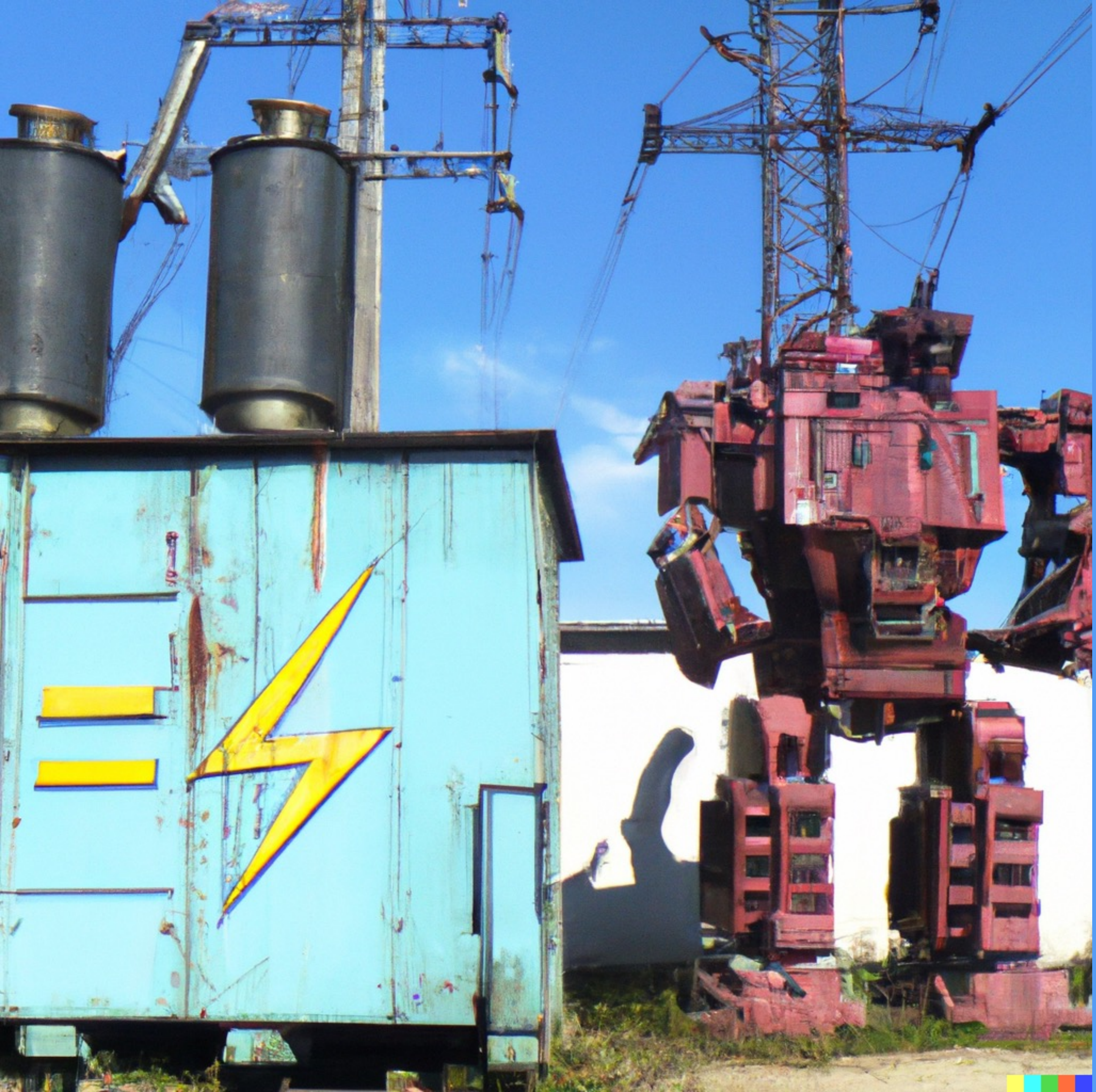


GPUs dominance

First-Order approximation:

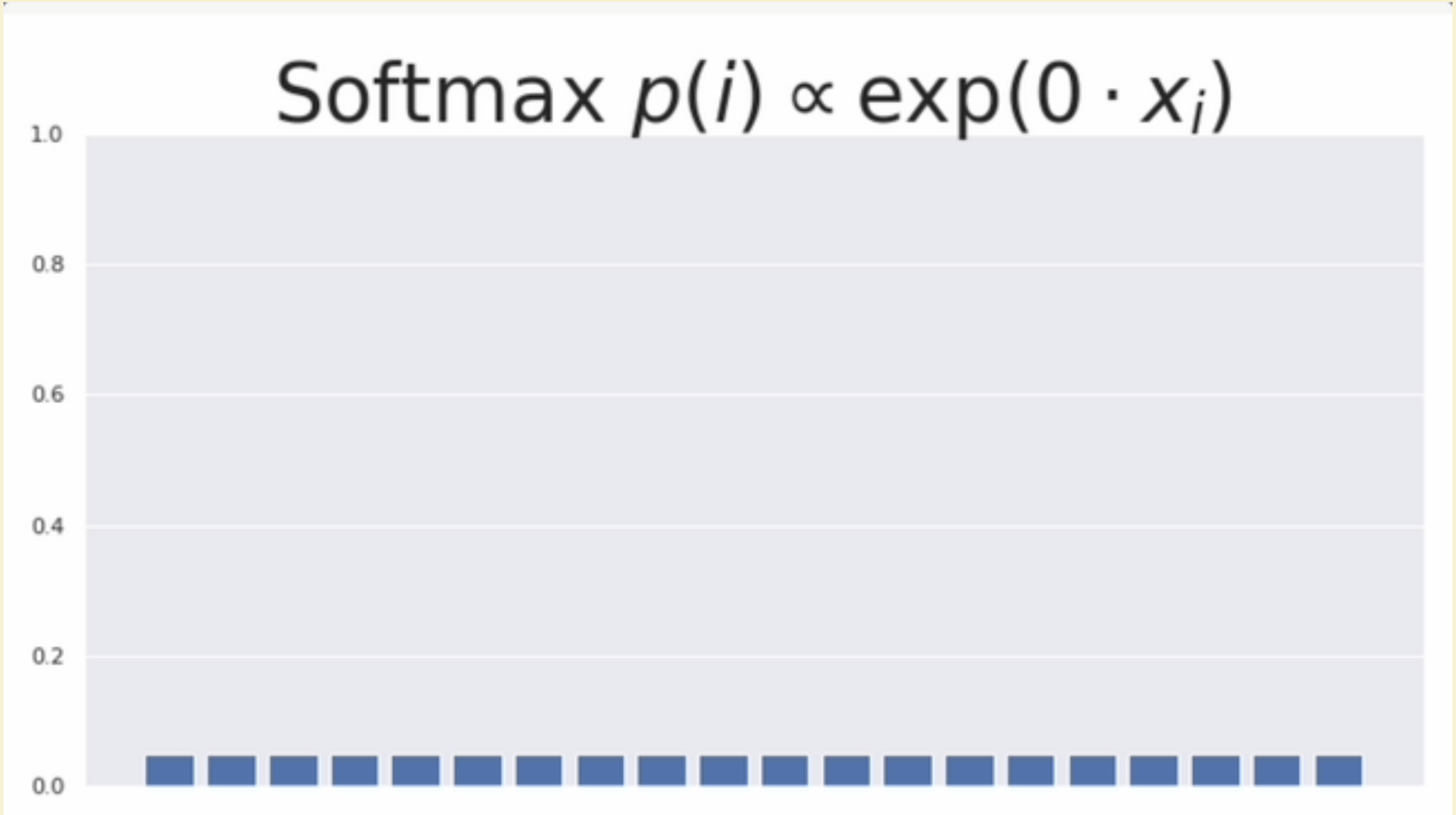
GPU = Dense Matrix Multiply Oracle

Wikipedia

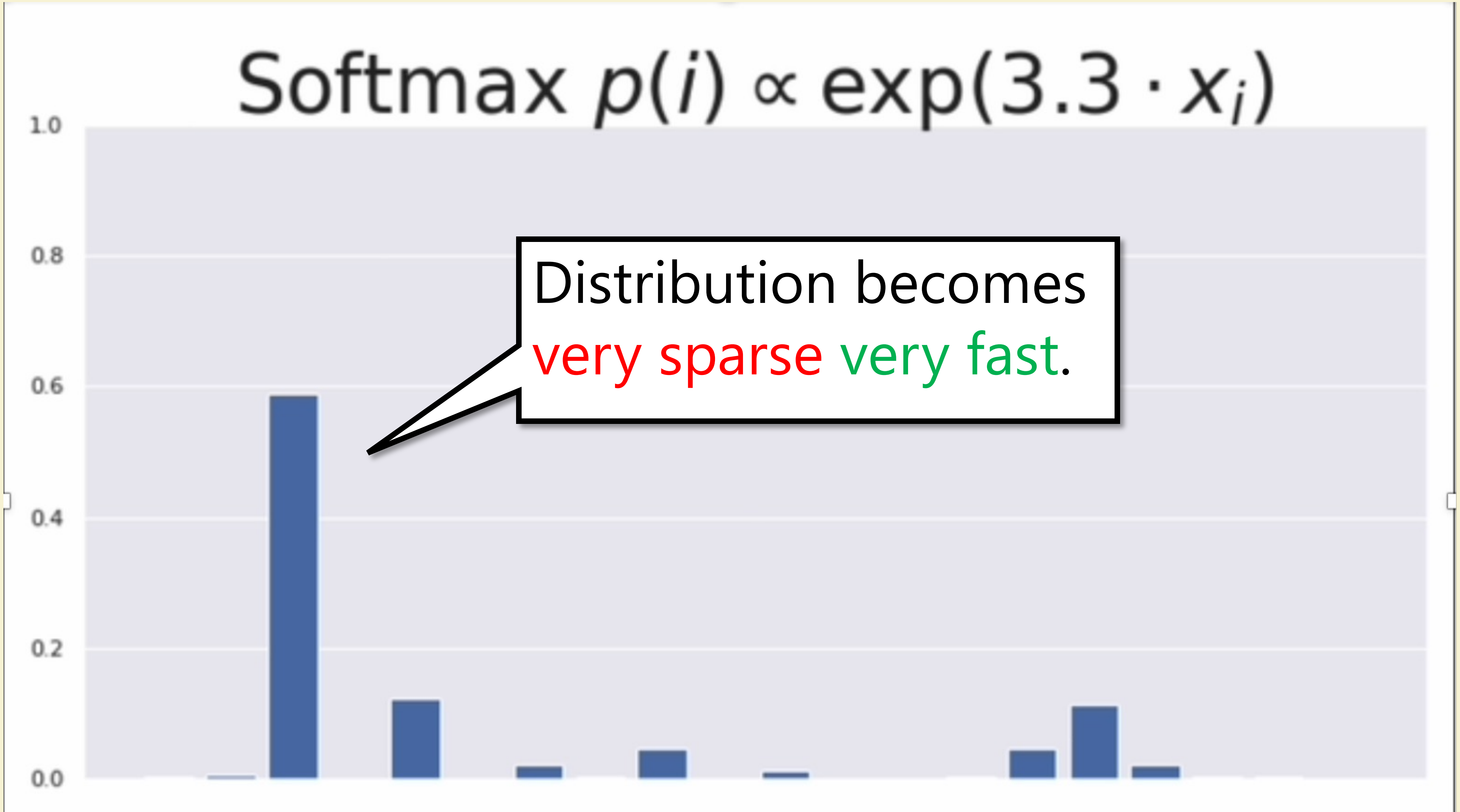


Part III: Transformers

Digression: Softmax



Digression: Softmax



Distribution becomes very sparse very fast.

Next-Token Prediction

Input: $t_1, \dots, t_n \in [k]$

Output: p distribution over $[k]$

Loss: $-\log p(t_{n+1})$

Next-Token Prediction

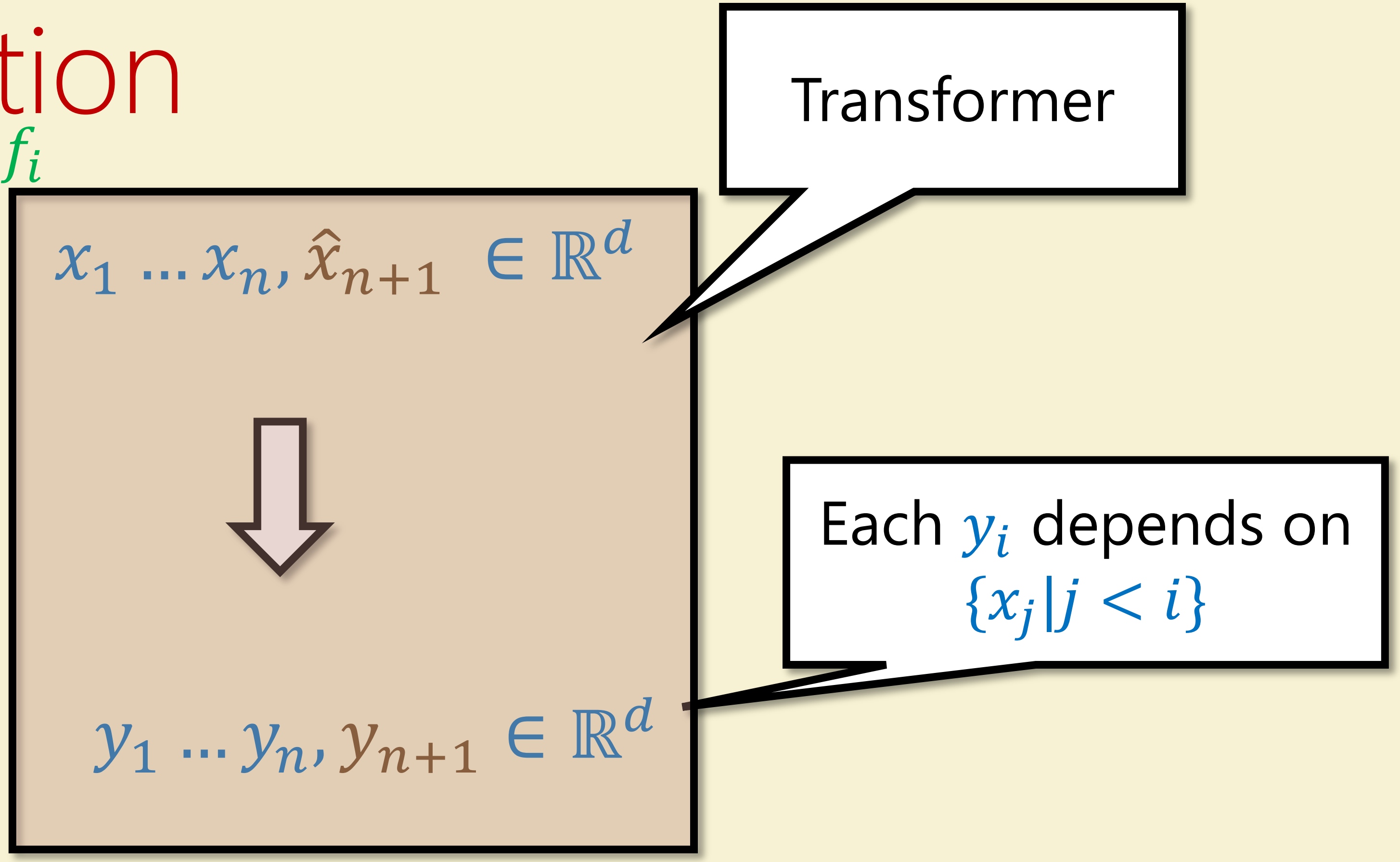
$x_i = e_{t_i} + f_i$

Input: $t_1, \dots, t_n \in [k]$

Embedding: $e_1 \dots e_k \in \mathbb{R}^d$

Positional embedding: $f_1 \dots f_n \in \mathbb{R}^d$

$p(i) \propto \exp(\langle e_i, y_{n+1} \rangle)$



$x_1 \dots x_n, \hat{x}_{n+1} \in \mathbb{R}^d$

$y_1 \dots y_n, y_{n+1} \in \mathbb{R}^d$

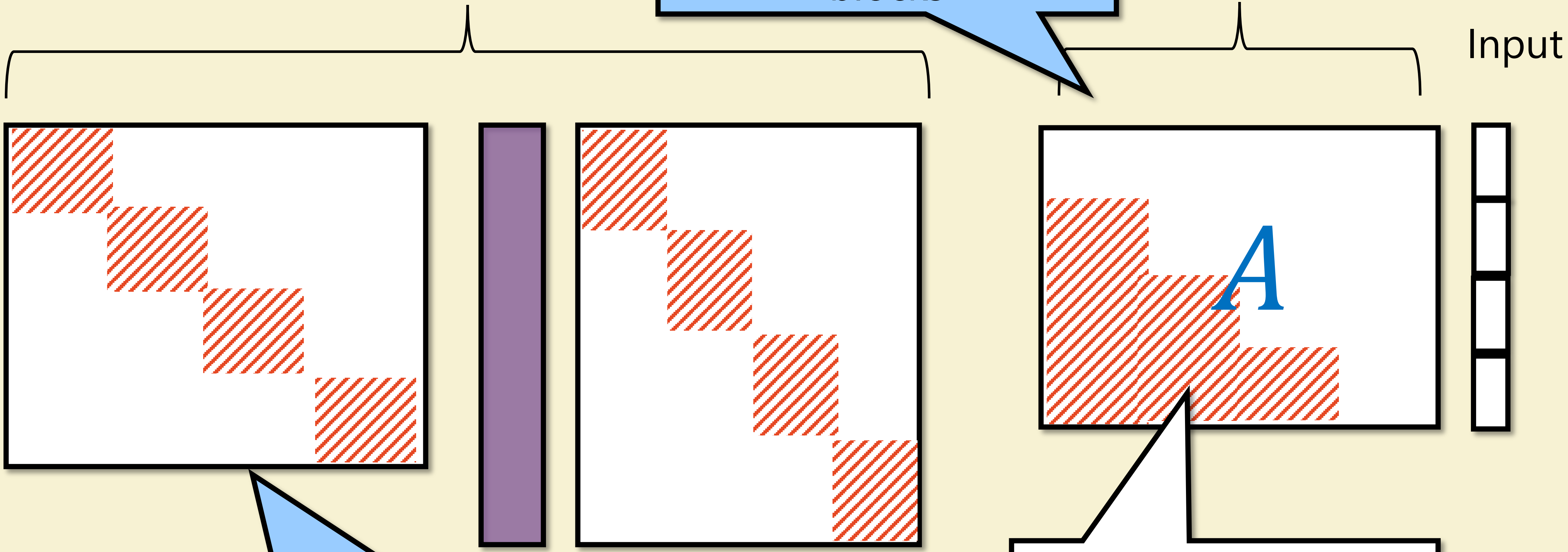
Transformer

Each y_i depends on $\{x_j | j < i\}$

Output: p distribution over $[k]$

Loss: $-\log p(t_{n+1})$

Transformers



Each output depends only on preceding blocks

Outputs computed block by block

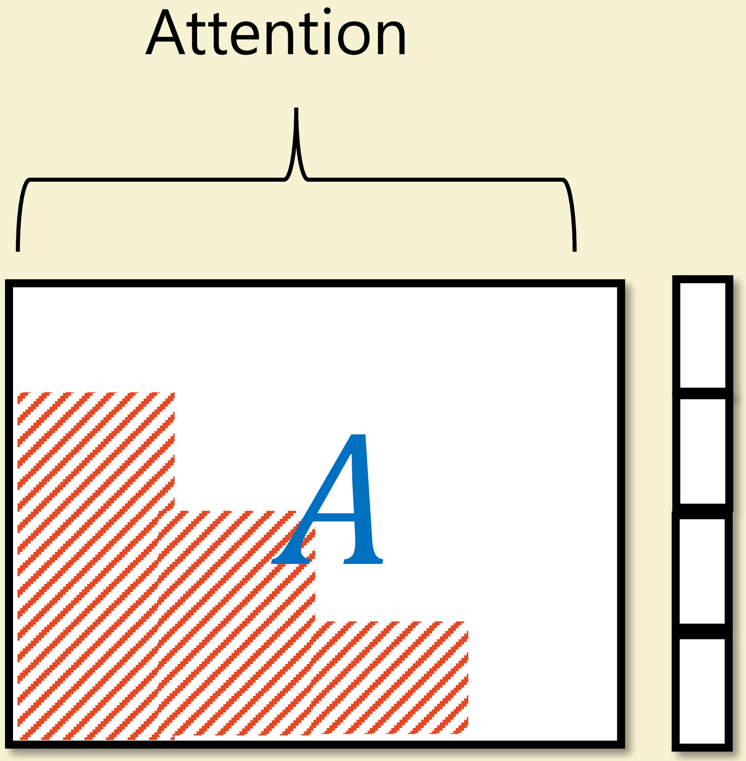
Cheating: Entries of A depend on input

Transformers

$A_{i,j}$ is H blocks

We want this to be $\leq O(1)$

$$A_{i,j,h} \propto \exp\left(\frac{Q_h x_i \cdot K_h x_j}{\sqrt{d_k}}\right) V_j$$



Why $\sqrt{d_k}$?

- Two vectors u, v in d dimensions, typically have $|u \cdot v| \approx \frac{\|u\| \cdot \|v\|}{\sqrt{d}}$
- If M is a random $d_k \times d_e$ matrix with $N(0,1)$ entries then $\| Mx \| \approx \sqrt{d_k} \| x \|$

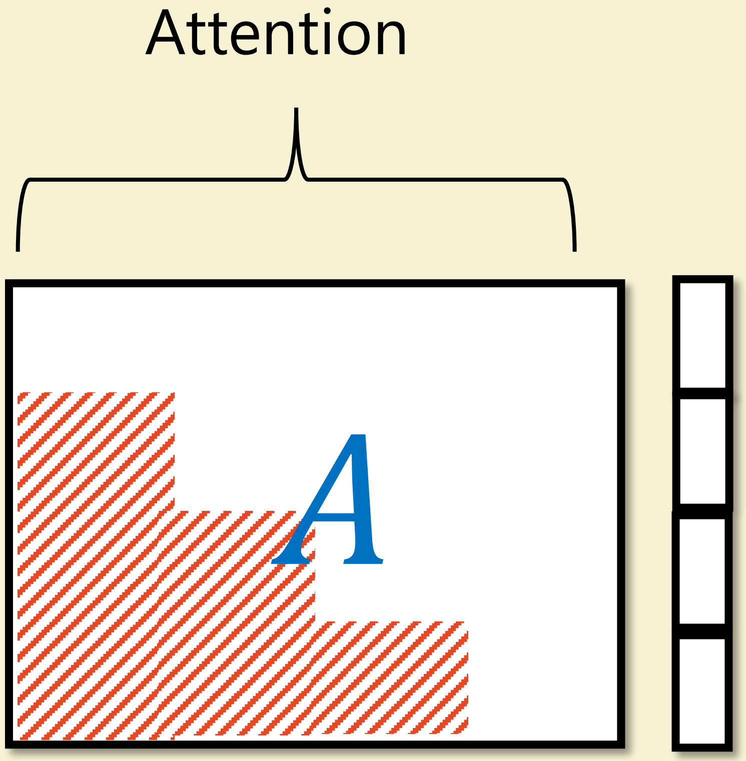
Proof: $\mathbb{E}(Mx)_i^2 = \sum_{j=1}^{d} \mathbb{E}[M_{i,j}^2 x_j^2] = \| x \|^2$

Transformers

$A_{i,j}$ is H blocks

We want this to be $\leq O(1)$

$$A_{i,j,h} \propto \exp\left(\frac{Q_h x_i \cdot K_h x_j}{\sqrt{d_k}}\right) V_j$$



Why $\sqrt{d_k}$?

- Two vectors u, v in d dimensions, typically have $|u \cdot v| \approx \frac{\|u\| \cdot \|v\|}{\sqrt{d}}$
- If M is a random $d_k \times d_e$ matrix with $N(0,1)$ entries then $\| Mx \| \approx \sqrt{d_k} \| x \|$
- We use layer norm to ensure $\| x_i \| = \| x_j \| \approx 1$

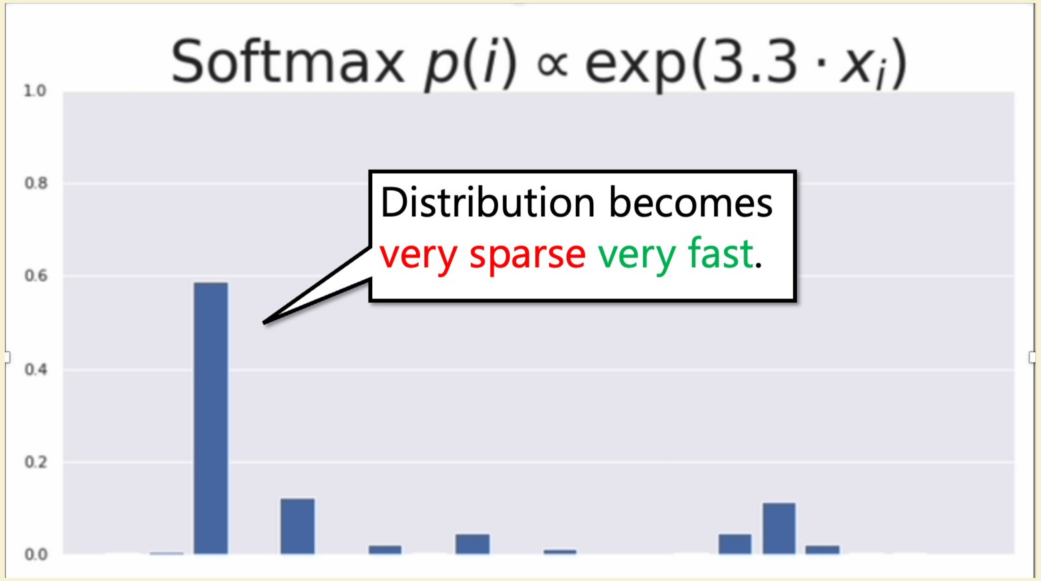
⇨ $$|Q_h x_i \cdot K_h x_j| \approx \frac{d_k}{\sqrt{d_k}}$$

Transformers

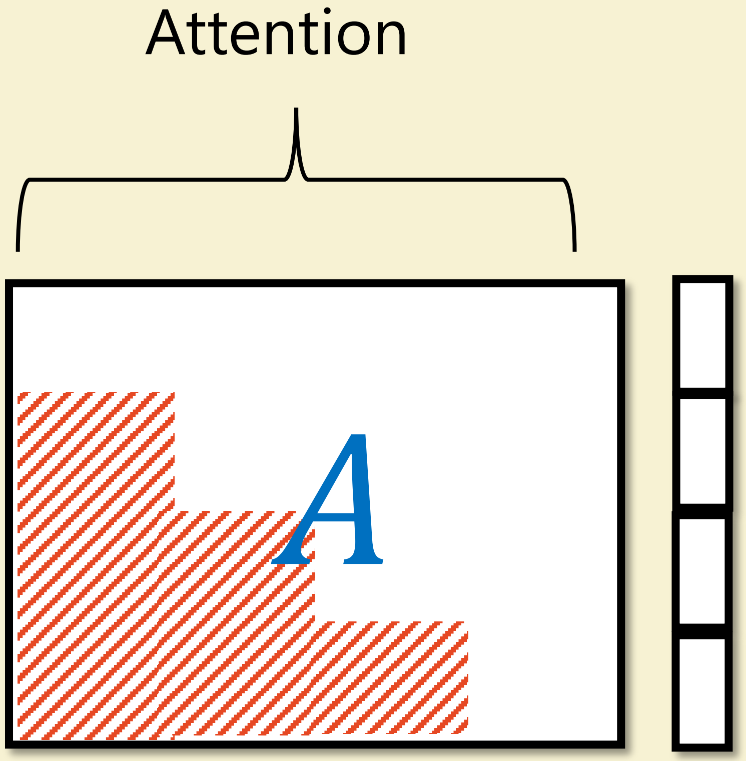
$A_{i,j}$ is H blocks

$$A_{i,j,h} \propto \exp\left(\frac{Q_h x_i \cdot K_h x_j}{\sqrt{d_k}}\right) V_j$$

Why multihead?



Distribution becomes very sparse very fast.



Limitations of transformers

1) Finite context n

$n = 4{,}000$ enough for language?

2) Quadratic overhead in n

Approaches to fix involve approximation