Transductive active learning for fine-tuning large (language) models

Jonas Hübotter

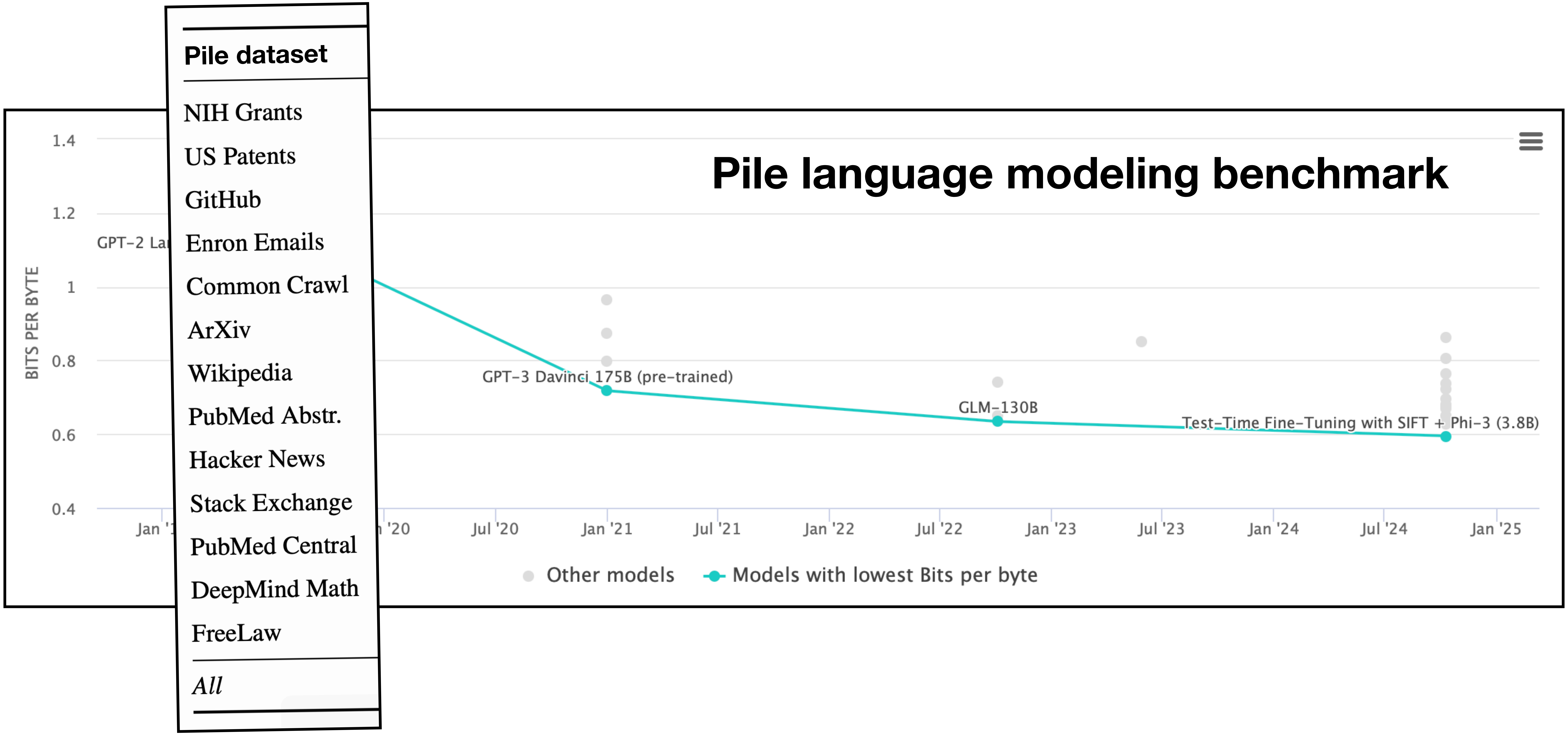
     

Efficiently learning at test-time with LLMs

~~Transductive active learning for fine-tuning large (language) models~~

Jonas Hübotter



[**H**, Bongni, Hakimi, Krause; preprint]

Train | **Test**

Local learning (at test-time)

known!

Test instance $\boldsymbol{x}^\star$

Training data

$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$

Learnt model

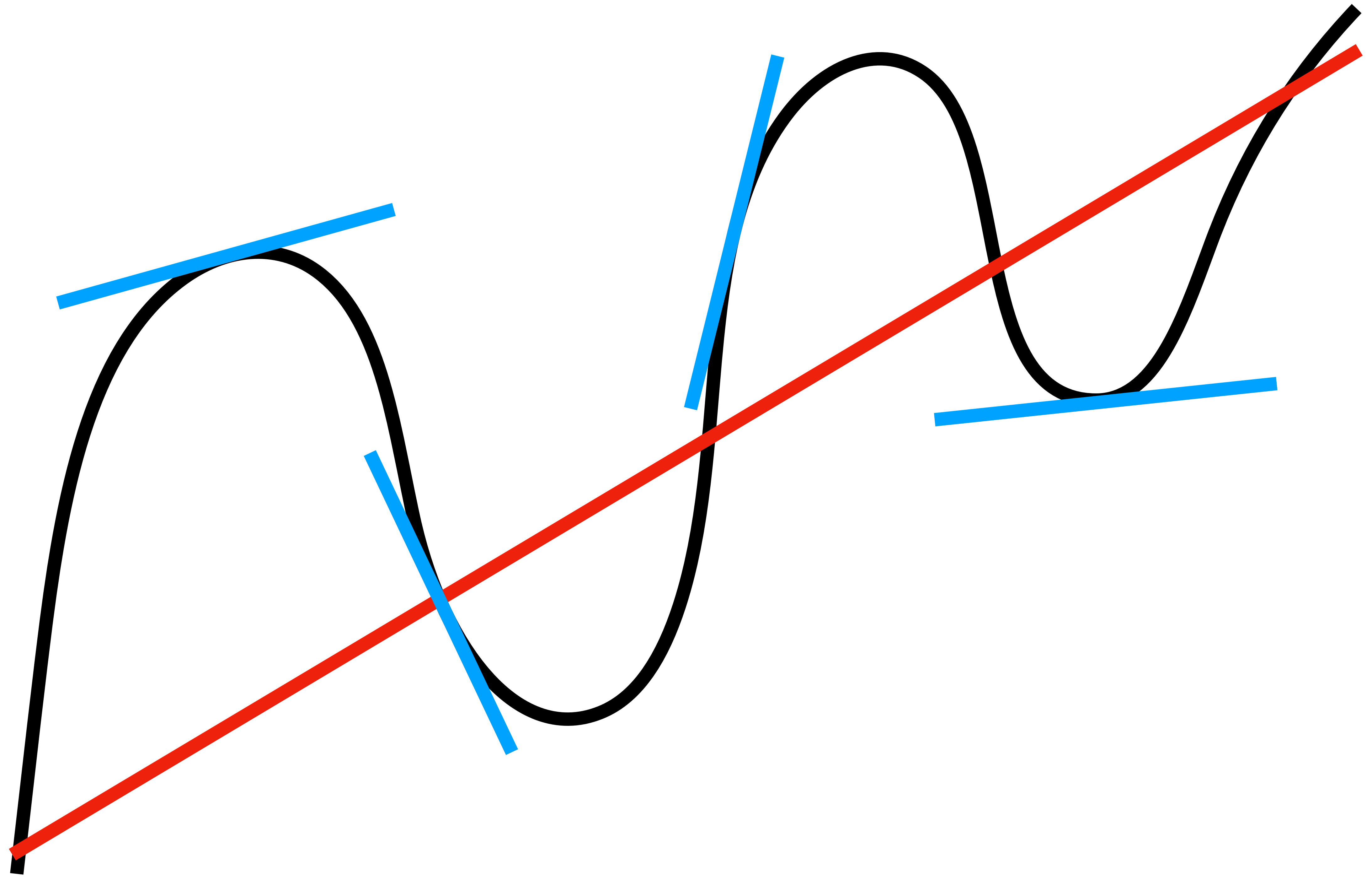
$f : \mathcal{X} \rightarrow \mathcal{Y}$

Prediction

$f(\boldsymbol{x}^\star)$

A story of curve fitting

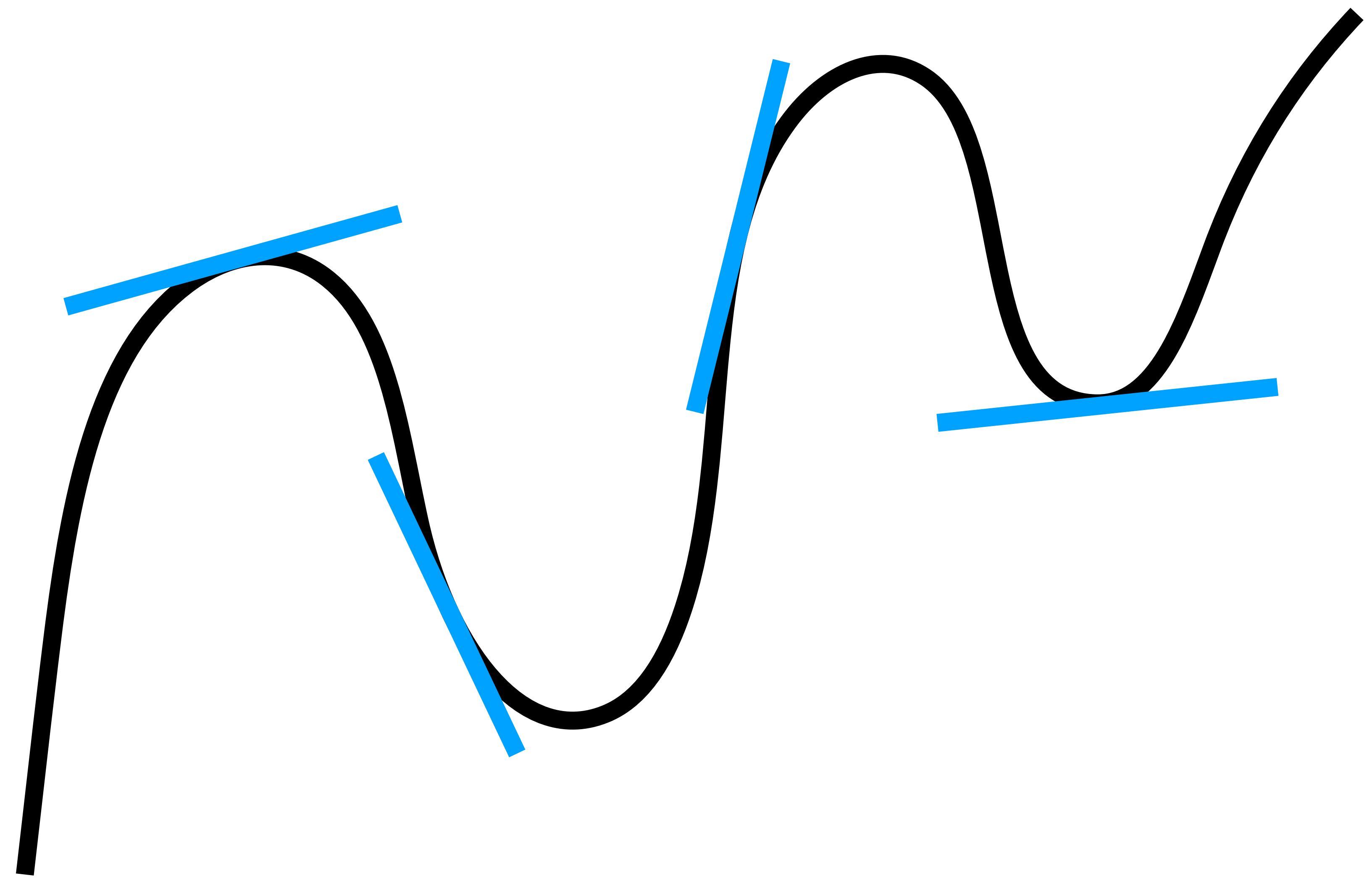
A story of curve fitting



Remedies:

- Parametric models
  polynomial regression
  neural networks
- Non-parametric models
  kernel (ridge) regression
  k-nearest neighbor
- **Local** models
  local linear regression
  ...

A story of curve fitting



Local models have two components:

- *Parametric* “controller”

  linear regression

  …

- *Non-parametric* “memory”

  k-nearest neighbor

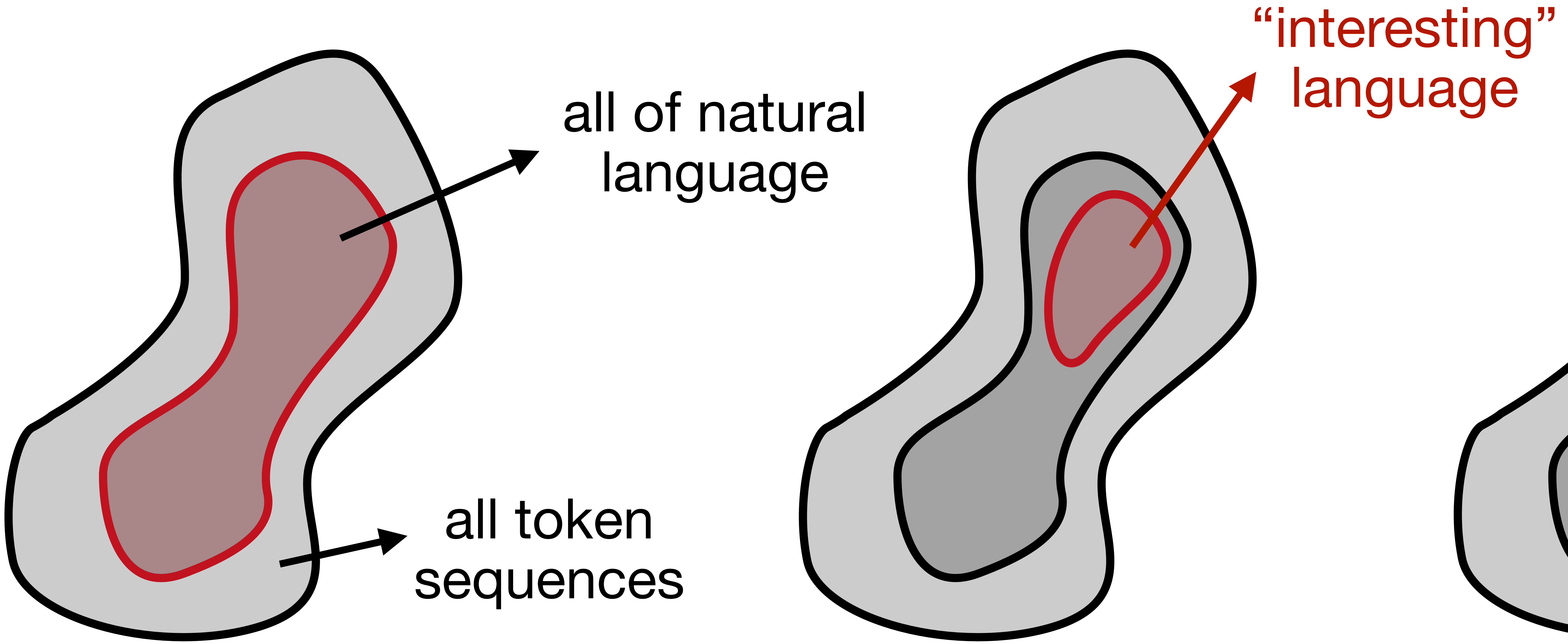
  …

→ a small model class can fit a rich function class!

→ one local model needs only little data!

→ too good to be true?

Local learning in a picture



inductive learning

"fine-tuning"

local learning

History

Fix

Hodges

Cover

Hart

- **since 1950s:** k-nearest neighbors
- **since 1960s:** kernel regression (Nadaraya & Watson)
- **since 1970s:** local (linear) learning (Cleveland & Devlin)
- **since 1980s:** transductive learning (Vapnik)

  > “When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one.”

- **in 1990s:** local fine-tuning (Vapnik & Bottou)

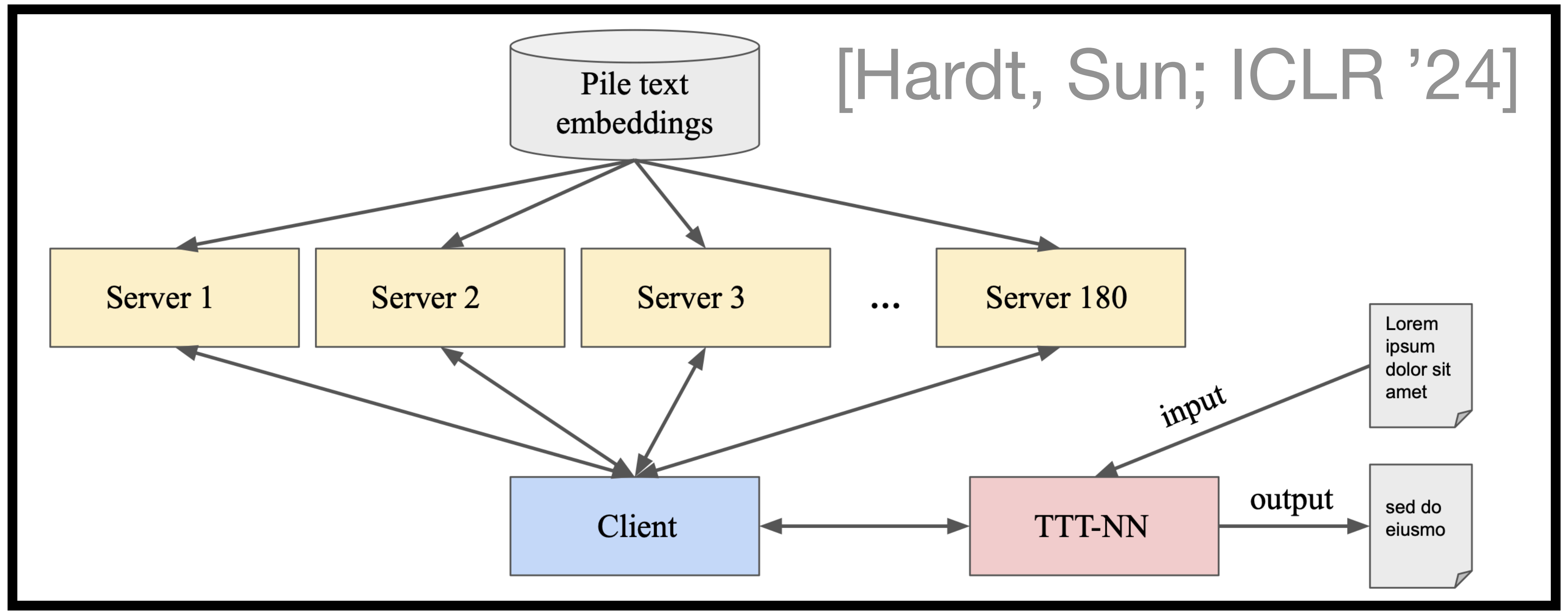
  CNNs on MNIST

History

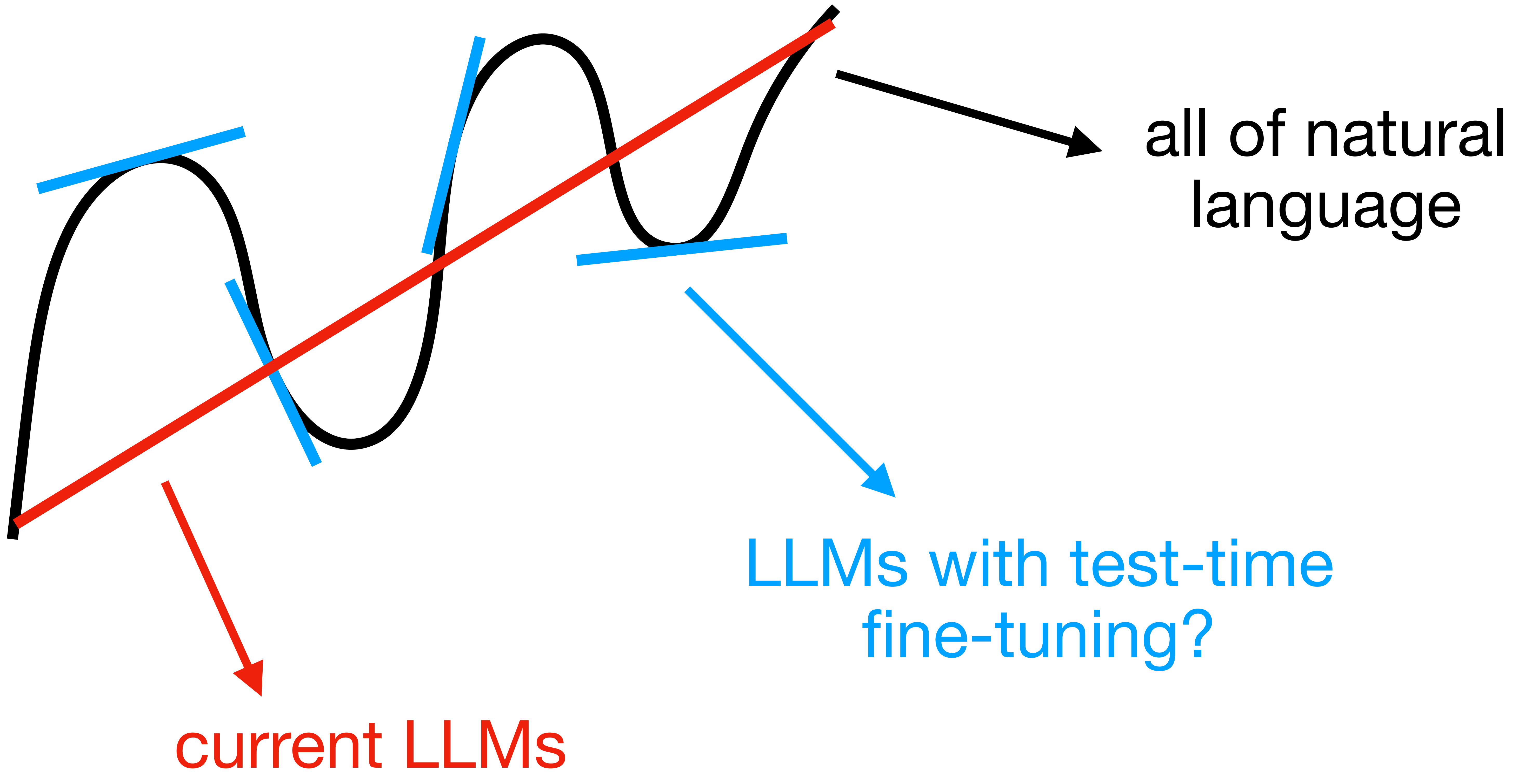
- **since 2020s:** (few-shot) in-context learning (GPT-3)

  parametric controller: LLM
  non-parametric memory: context (+ retrieval from database)

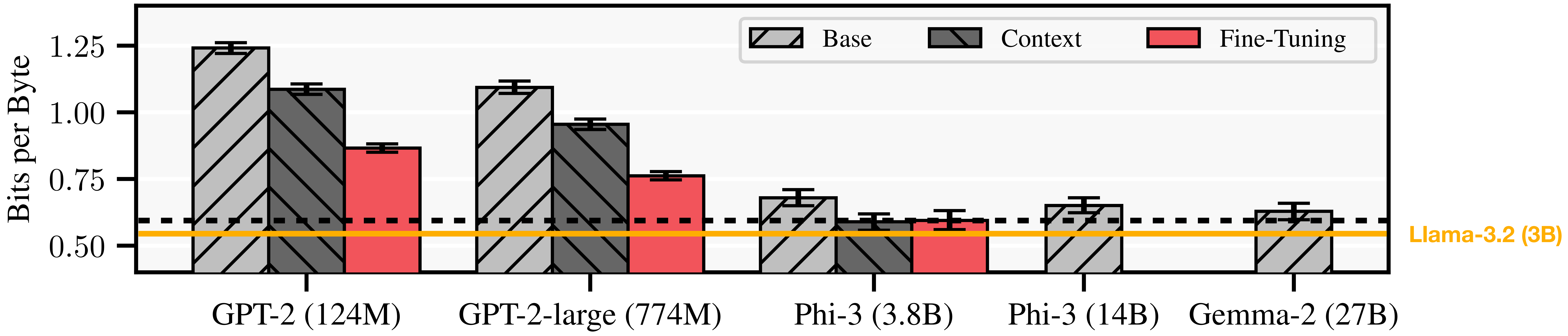
- **recently:** local fine-tuning (again!) with GPT-2 (Hardt & Sun)



Hypothesis for LLMs



Does local learning work with LLMs?



	Context	Fine-Tuning	Δ
GitHub	74.6 (2.5)	**28.6** (2.2)	↓56.0
DeepMind Math	100.2 (0.1)	**70.1** (2.1)	↓30.1
US Patents	87.4 (2.5)	**62.2** (3.6)	↓25.2
FreeLaw	87.2 (3.6)	**65.5** (4.2)	↓21.7

GPT-2

	Context	Fine-Tuning	Δ
GitHub	74.6 (2.5)	**31.0** (2.2)	↓43.6
DeepMind Math	100.2 (0.7)	**74.2** (2.3)	↓26.0
US Patents	87.4 (2.5)	**64.7** (3.8)	↓22.7
FreeLaw	87.2 (3.6)	**68.3** (4.2)	↓18.9

GPT-2-large

	Context	Fine-Tuning	Δ
DeepMind Math	100.8	75.3	↓25.5
GitHub	71.3	46.5	↓24.8
FreeLaw	78.2	67.2	↓11.0
ArXiv	101.0	94.3	↓6.4

Phi-3

Key challenge: which data to select?

Key challenge: which data to select?

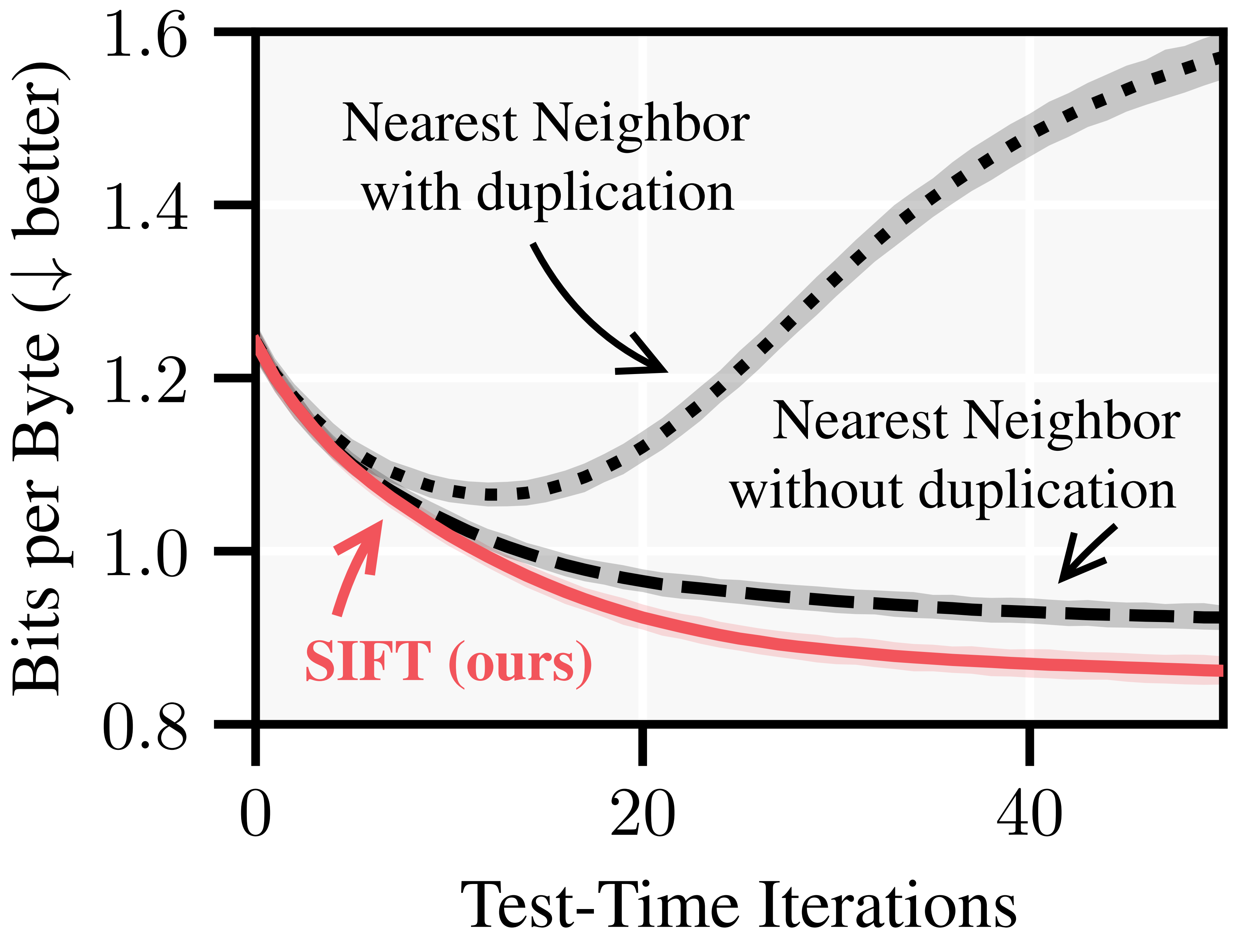
Prompt: What is the age of Michael Jordan and how many kids does he have?

Nearest Neighbor:

1. The age of Michael Jordan is 61 years.
2. Michael Jordan was born on February 17, 1963.

SIFT (ours):

1. The age of Michael Jordan is 61 years.
2. Michael Jordan has five children.



Key challenge: which data to select?

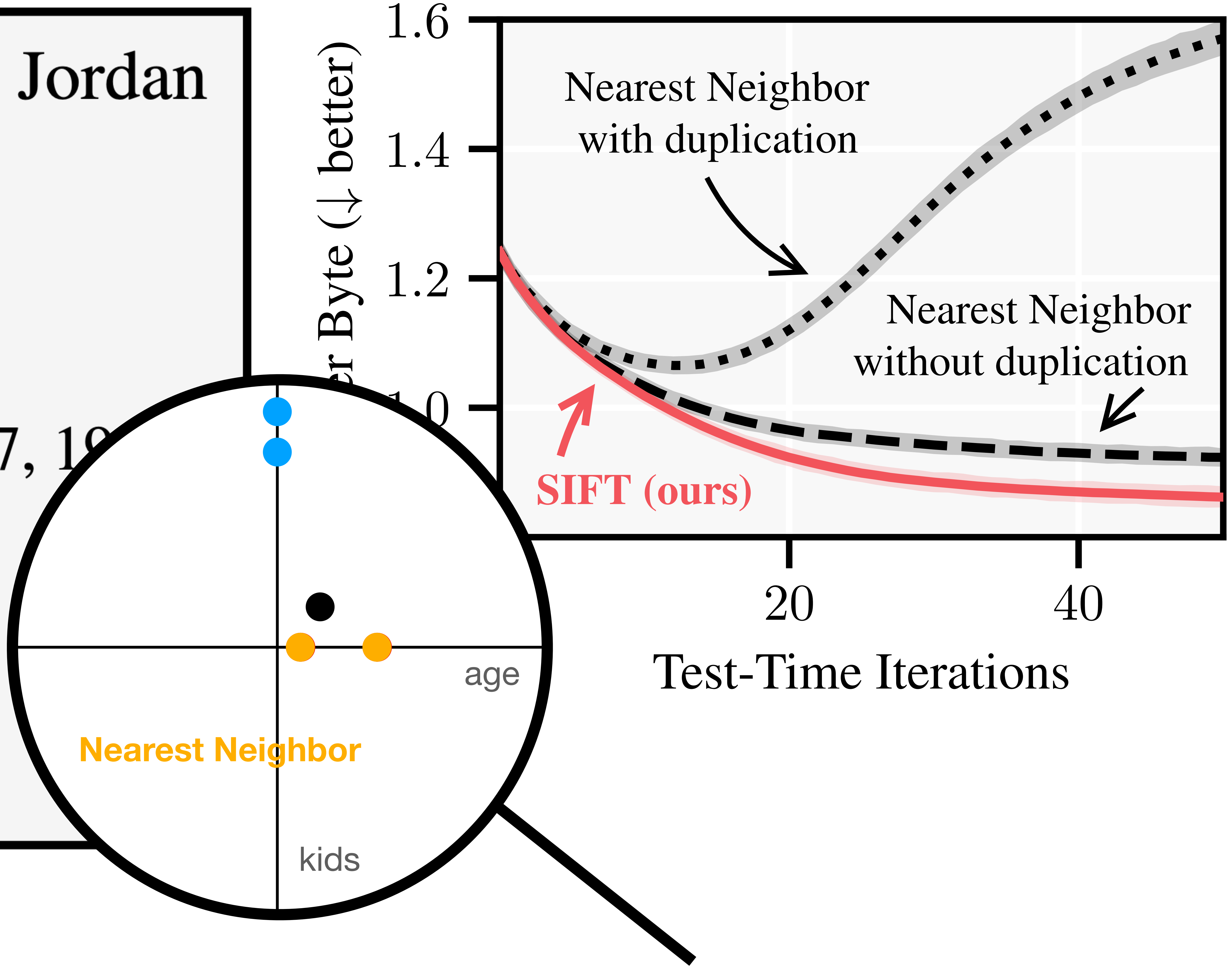
Prompt: What is the age of Michael Jordan and how many kids does he have?

Nearest Neighbor:

1. The age of Michael Jordan is 61 years.
2. Michael Jordan was born on February 17, 19

SIFT (ours):

1. The age of Michael Jordan is 61 years.
2. Michael Jordan has five children.



Key challenge: which data to select?

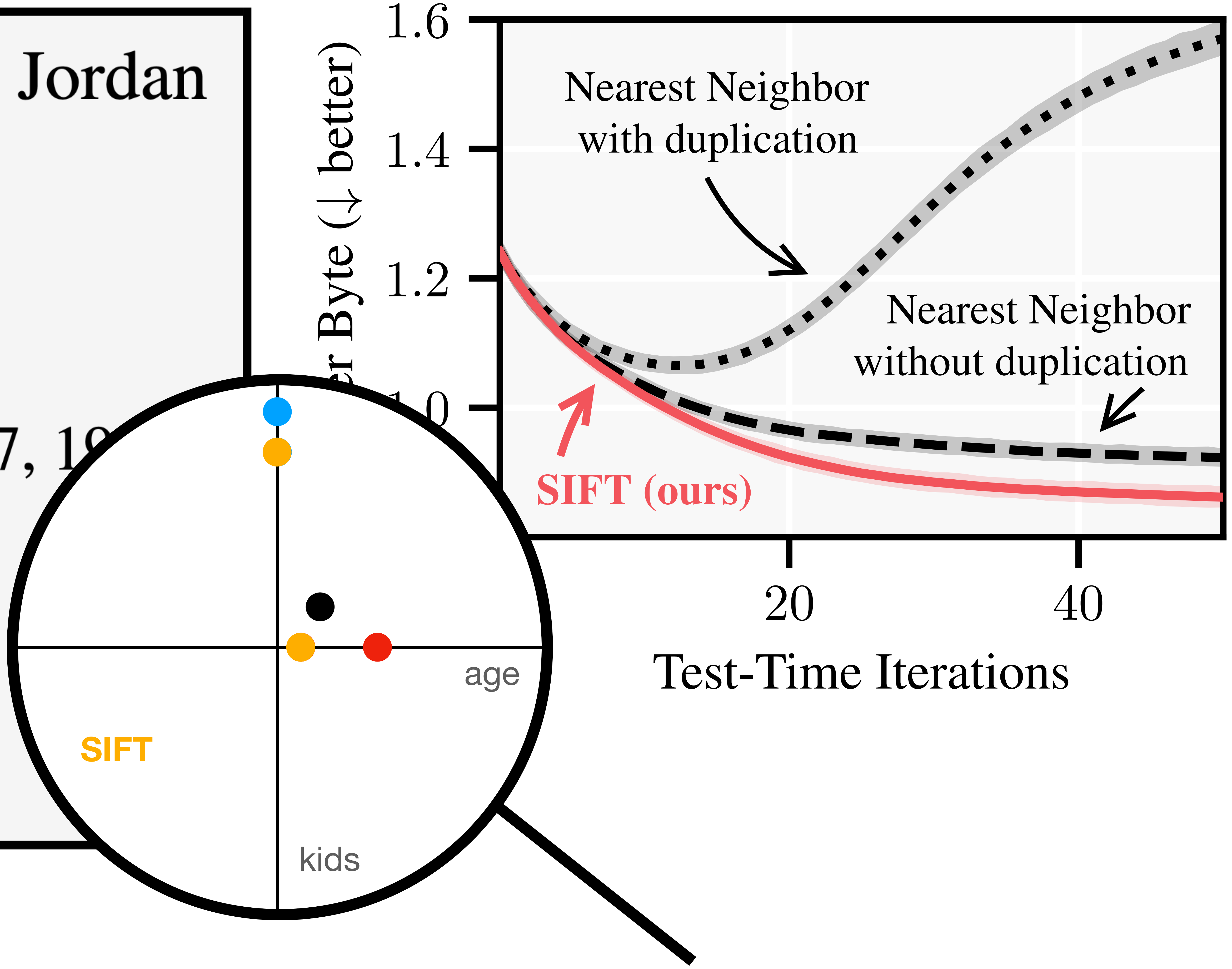
Prompt: What is the age of Michael Jordan and how many kids does he have?

Nearest Neighbor:

1. The age of Michael Jordan is 61 years.
2. Michael Jordan was born on February 17, 19

SIFT (ours):

1. The age of Michael Jordan is 61 years.
2. Michael Jordan has five children.



SIFT: selecting informative data for fine-tuning

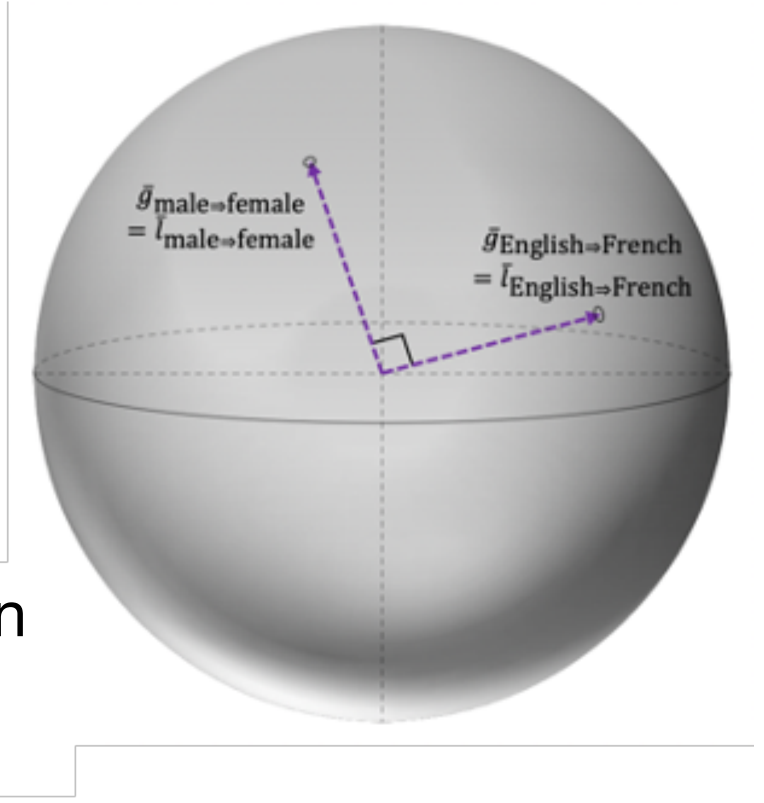
Principle:

Select data that *maximally* reduces “uncertainty” about how to respond to the prompt.

1. Estimate uncertainty
2. Minimize “posterior” uncertainty

[**H**, Bongni, Hakimi, Krause; preprint]

❶ Estimating uncertainty



- Making this tractable:

unknown ↘ known ↙

Surrogate model: logit-linear model $s(f^\star(x))$ with $f^\star(x) = \boldsymbol{W}^\star \boldsymbol{\phi}(x)$

→ linear representation hypothesis [Park, Choe, Veitch; ICML '24]

$$\mathcal{L}^\lambda(\boldsymbol{W}; D) = \underbrace{-\sum_{(x,y)\in D} \log s_y(f(x; \boldsymbol{W}))}_{\text{cross-entropy loss (NLL)}} + \underbrace{\frac{\lambda}{2}\|\boldsymbol{W} - \boldsymbol{W}^{\text{pre}}\|_F^2}_{\text{regularization}} \qquad \boldsymbol{W}_n = \arg\min_{\boldsymbol{W}} \mathcal{L}^\lambda(\boldsymbol{W}; D_n)$$

$$\underbrace{s^\star(x) = s(f^\star(x))}_{\text{"truth"}} \qquad \underbrace{s_n(x) = s(\boldsymbol{W}_n \boldsymbol{\phi}(x))}_{\text{model trained on } n \text{ pieces of data}}$$

- *Confidence sets:* $\underbrace{d_{\text{TV}}(s_n(x), s^\star(x))}_{\text{error}} \leq \underbrace{\beta_n(\delta)}_{\text{scaling}} \underbrace{\sigma_n(x)}_{\text{key object}}$ (w.p. $1 - \delta$)

→ $\sigma_n(x)$ measures **uncertainty** about response to x!

❶ Estimating uncertainty

- Are **regularized loss minimization** and **fine-tuning** related?

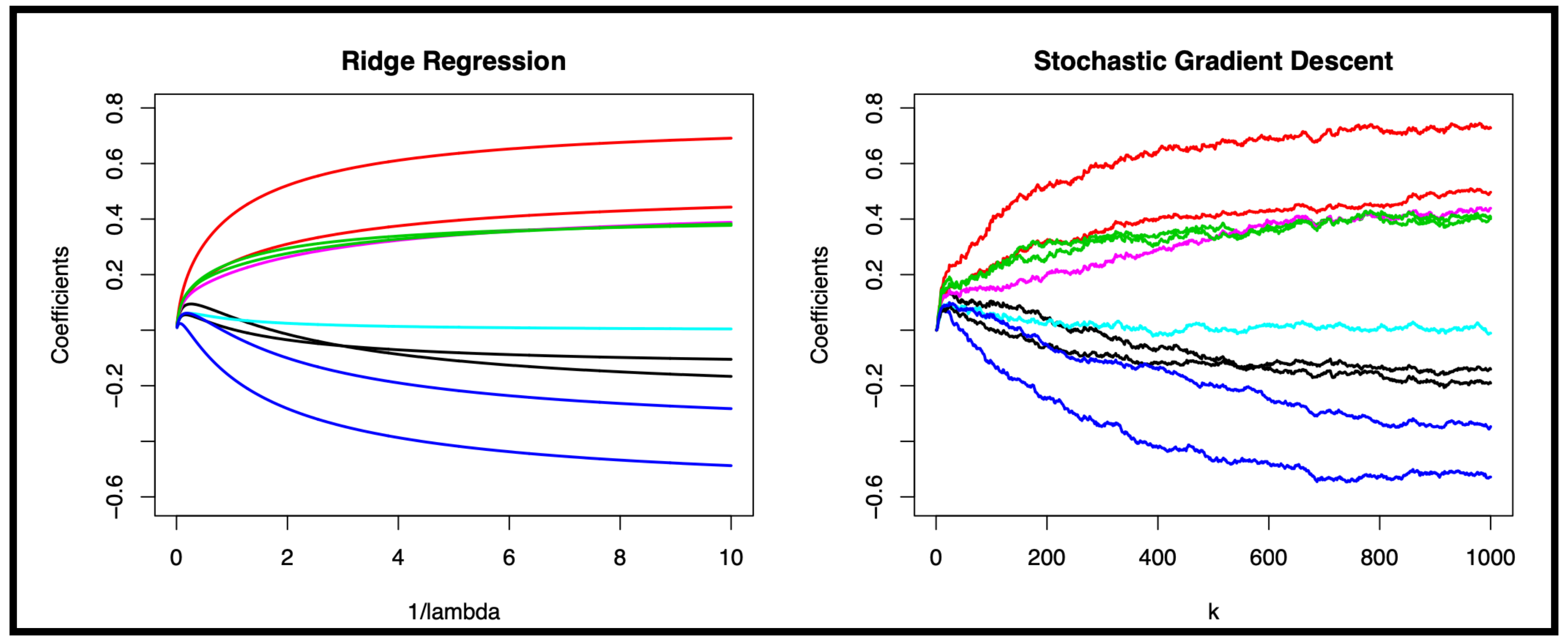
  Consider two alternative models:

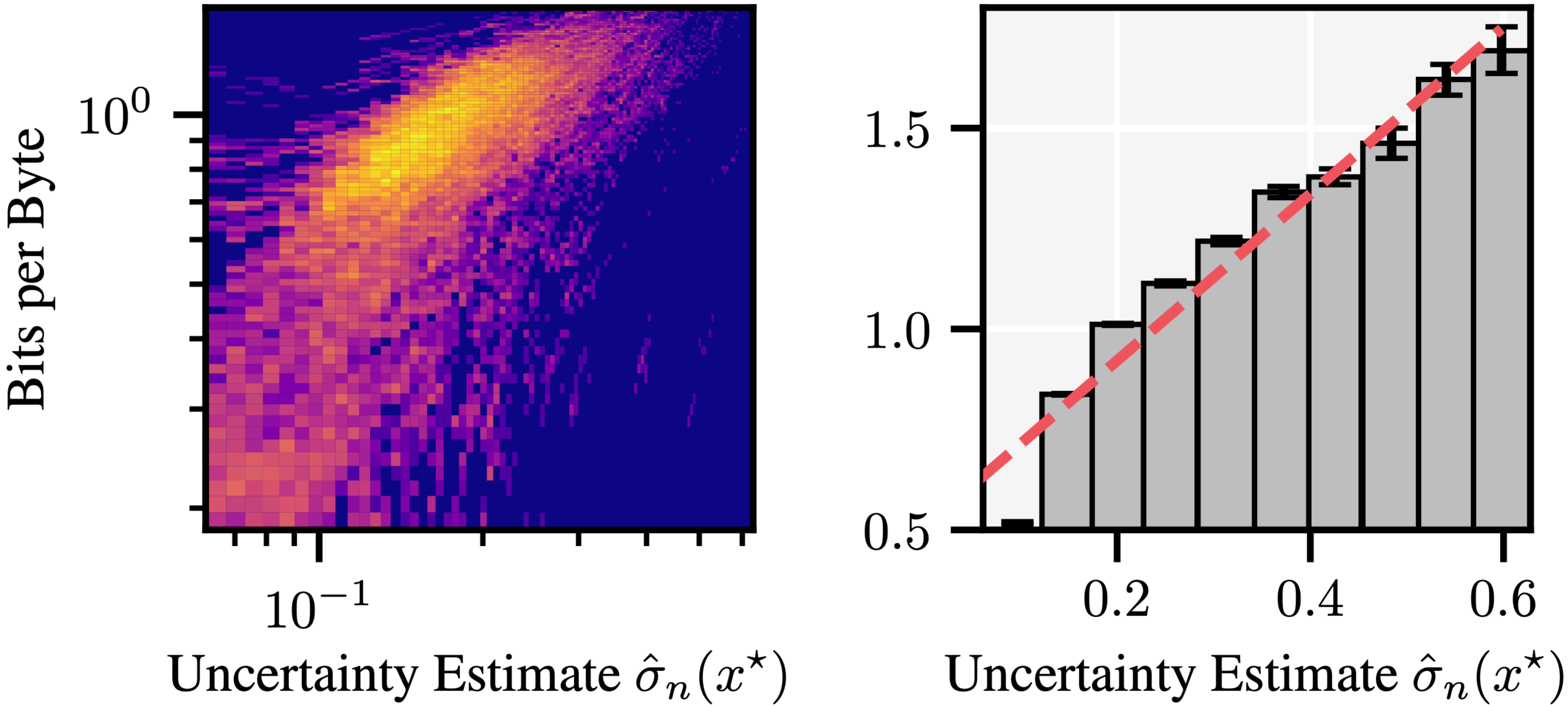
  - $\boldsymbol{W}_\lambda = \arg\min_{\boldsymbol{W}} \mathscr{L}^\lambda(\boldsymbol{W})$ → minimizer of regularized loss
  - $\widehat{\boldsymbol{W}}_\eta = \boldsymbol{W}^{\text{pre}} - \eta \nabla \mathscr{L}(\boldsymbol{W}^{\text{pre}})$ → single gradient-step fine-tuning ($\mathscr{L}$ is NLL)

- *Proposition:* $\|\boldsymbol{W}_{1/\eta} - \widehat{\boldsymbol{W}}_\eta\|_{\text{F}} \leq \eta \|\nabla \mathscr{L}(\boldsymbol{W}_{1/\eta}) - \nabla \mathscr{L}(\boldsymbol{W}^{\text{pre}})\|$

[see also Ali et al.; ICML ’20]

→ models similar for $\lambda \approx 1/\eta$!





→ $\sigma_n(x)$ measures **uncertainty** about response to x!

❷ Minimizing "posterior" uncertainty

- Choose data that minimizes uncertainty of the model <u>after</u> seeing this data:

$$x_{n+1} = \underset{x}{\operatorname{argmin}}\ \sigma_{X_n \cup \{x\}}(x^\star)$$

← prompt

$$= \underset{x}{\operatorname{argmax}} \begin{bmatrix} k(x^\star, x_1) \\ \vdots \\ k(x^\star, x_n) \\ k(x^\star, x) \end{bmatrix}^\top \left(\begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) & k(x_1, x) \\ \vdots & \ddots & \vdots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) & k(x_n, x) \\ k(x, x_1) & \cdots & k(x, x_n) & k(x, x) \end{bmatrix} + \lambda I_{n+1} \right)^{-1} \begin{bmatrix} k(x^\star, x_1) \\ \vdots \\ k(x^\star, x_n) \\ k(x^\star, x) \end{bmatrix}$$

with $k(x, x') = \boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(x')$

maximize relevance

minimize redundancy

- *Convergence guarantee* (in case of no synergies):

$$\sigma_n^2(x^\star) - \sigma_\infty^2(x^\star) \leq O(\lambda \log n)/\sqrt{n}$$

irreducible uncertainty

Not possible with nearest neighbor retrieval!

→ predictions can be only as good as the data and the learned abstractions!

❷ Minimizing "posterior" uncertainty (example)

- Example: suppose embeddings are normalized

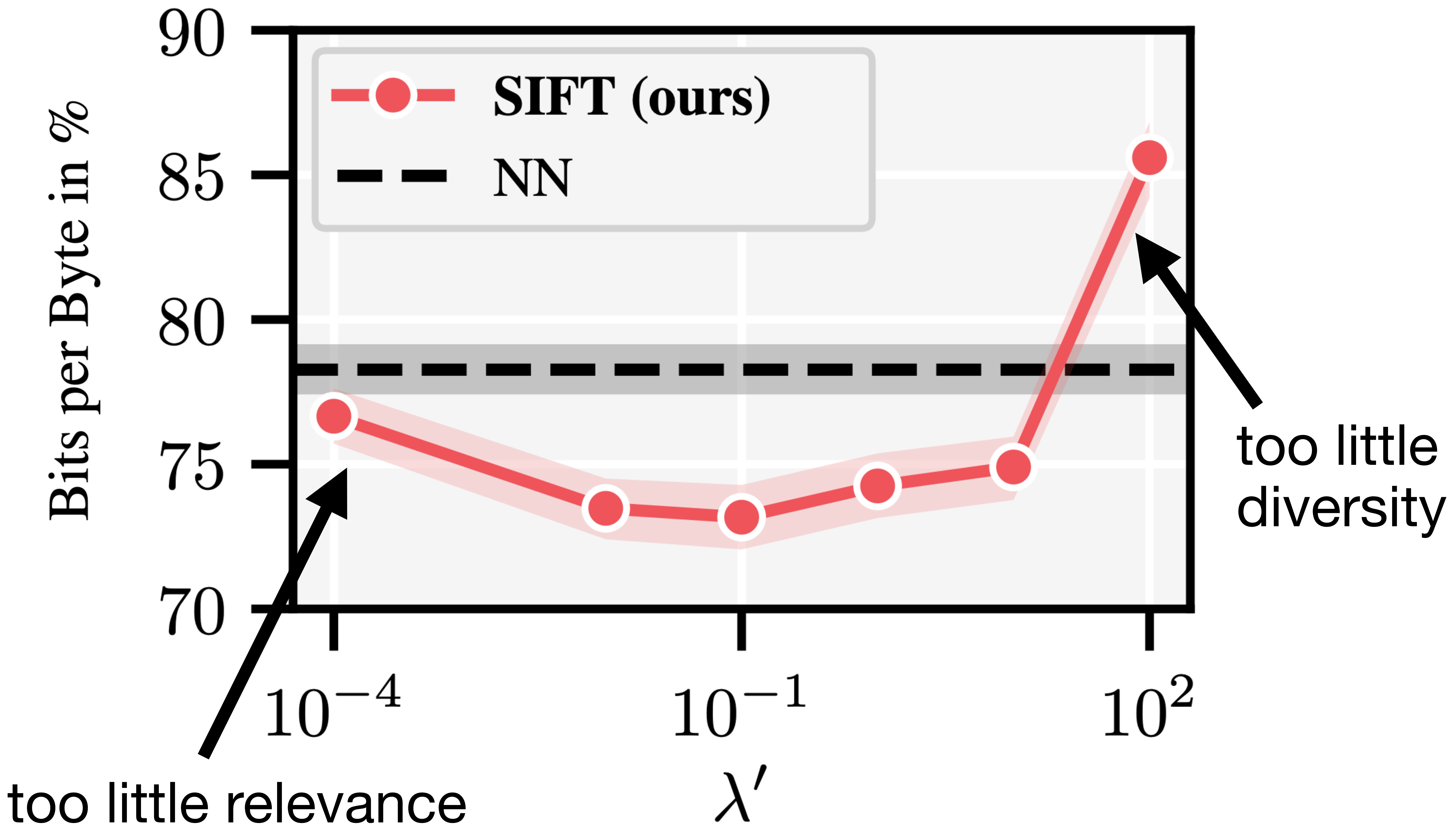
$$\boldsymbol{x}_1 = \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathcal{D}} \sigma^2_{\{\boldsymbol{x}\}}(\boldsymbol{x}^\star) = \operatorname*{arg\,max}_{\boldsymbol{x}\in\mathcal{D}} \frac{(\boldsymbol{\phi}(\boldsymbol{x}^\star)^\top \boldsymbol{\phi}(\boldsymbol{x}))^2}{1+\lambda} = \operatorname*{arg\,max}_{\boldsymbol{x}\in\mathcal{D}} \Big(\underbrace{\measuredangle_{\boldsymbol{\phi}}(\boldsymbol{x}^\star, \boldsymbol{x})}_{\text{cosine similarity of } \boldsymbol{\phi}(\boldsymbol{x}^\star),\, \boldsymbol{\phi}(\boldsymbol{x})} \Big)^2. \quad \textbf{(1st point)}$$

$$\boldsymbol{x}_2 = \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathcal{D}} \sigma^2_{\{\boldsymbol{x}_1,\boldsymbol{x}\}}(\boldsymbol{x}^\star) = \operatorname*{arg\,max}_{\boldsymbol{x}\in\mathcal{D}} \begin{bmatrix} \measuredangle_{\boldsymbol{\phi}}(\boldsymbol{x}^\star, \boldsymbol{x}_1) \\ \measuredangle_{\boldsymbol{\phi}}(\boldsymbol{x}^\star, \boldsymbol{x}) \end{bmatrix}^\top \begin{bmatrix} 1+\lambda & \measuredangle_{\boldsymbol{\phi}}(\boldsymbol{x}_1, \boldsymbol{x}) \\ \measuredangle_{\boldsymbol{\phi}}(\boldsymbol{x}_1, \boldsymbol{x}) & 1+\lambda \end{bmatrix}^{-1} \begin{bmatrix} \measuredangle_{\boldsymbol{\phi}}(\boldsymbol{x}^\star, \boldsymbol{x}_1) \\ \measuredangle_{\boldsymbol{\phi}}(\boldsymbol{x}^\star, \boldsymbol{x}) \end{bmatrix}. \quad \textbf{(2nd point)}$$

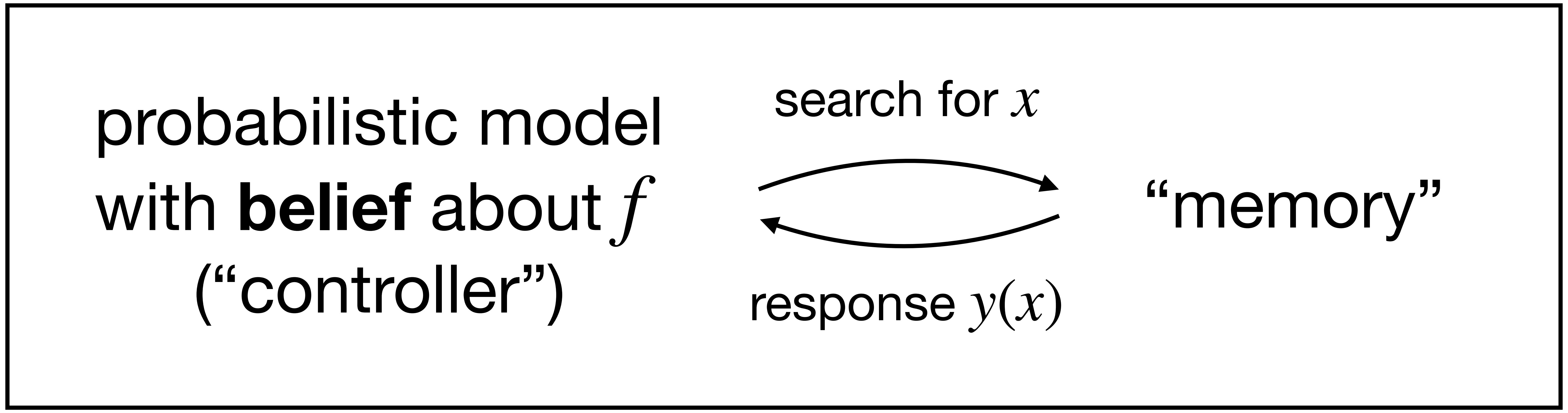
- Example: suppose $\boldsymbol{x}$ is such that $\measuredangle_{\boldsymbol{\phi}}(\boldsymbol{x}_1, \boldsymbol{x}) = 0$. Then $\boldsymbol{x}$ is preferred over $\boldsymbol{x}_1$ iff

$$\measuredangle_{\boldsymbol{\phi}}(\boldsymbol{x}^\star, \boldsymbol{x})^2 > \frac{\lambda}{2+\lambda} \measuredangle_{\boldsymbol{\phi}}(\boldsymbol{x}^\star, \boldsymbol{x}_1)^2$$

→ as $\lambda \to \infty$: maximum relevance, as $\lambda \to 0$: minimum redundancy


SIFT (ours)
NN
Bits per Byte in %
90
85
80
75
70
10^{-4}
10^{-1}
10^{2}
λ'
too little relevance
too little diversity

A probabilistic interpretation of SIFT



Tractable Probabilistic Model

$$y(x) = f(x) + \varepsilon(x)$$

$$f \sim \mathcal{GP}(\mu, k)$$

$$\varepsilon(x) \overset{iid}{\sim} \mathcal{N}(0, \sqrt{\lambda})$$

posterior variance $\sigma_n^2(x^\star)$

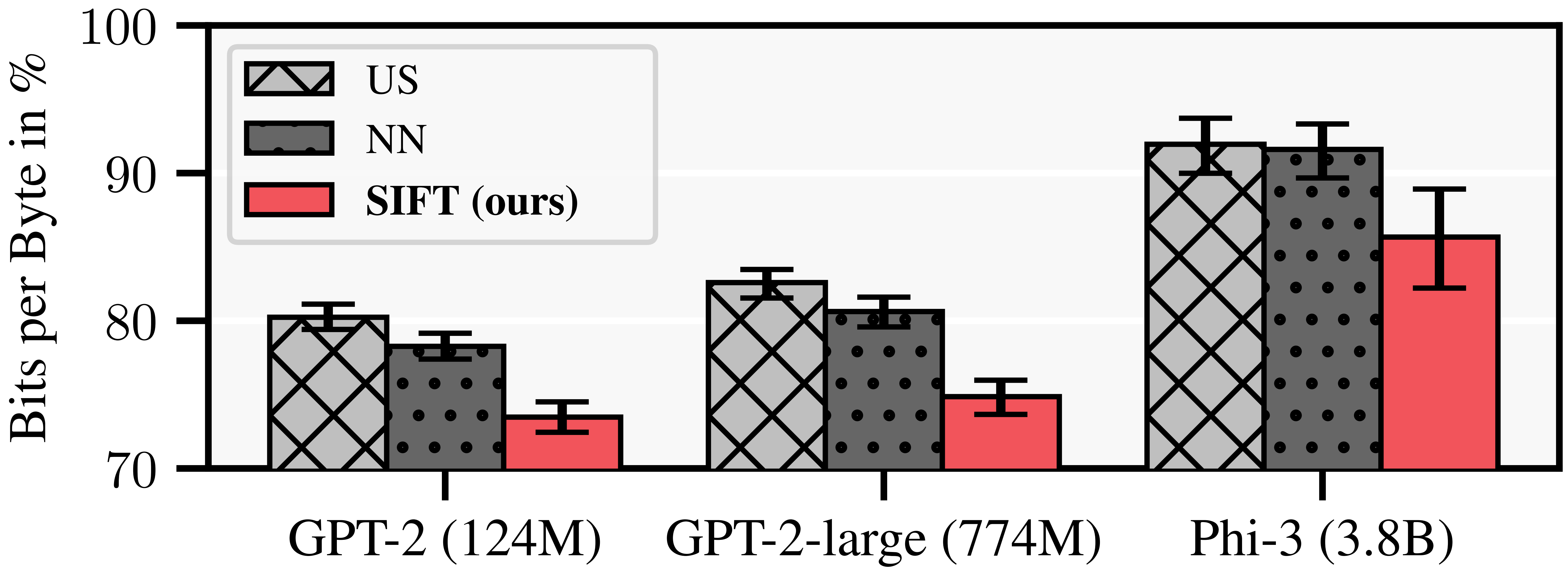
$$x_{n+1} = \arg\min_x \mathrm{Var}(f(x^\star) \mid y_{1:n}, y(x))$$

$$= \arg\max_x \mathrm{I}(f(x^\star); y(x) \mid y_{1:n})$$

$$= \arg\max_x \mathrm{I}(f(x^\star); y(x)) - \mathrm{I}(f(x^\star); y(x); y_{1:n})$$

relevance — redundancy

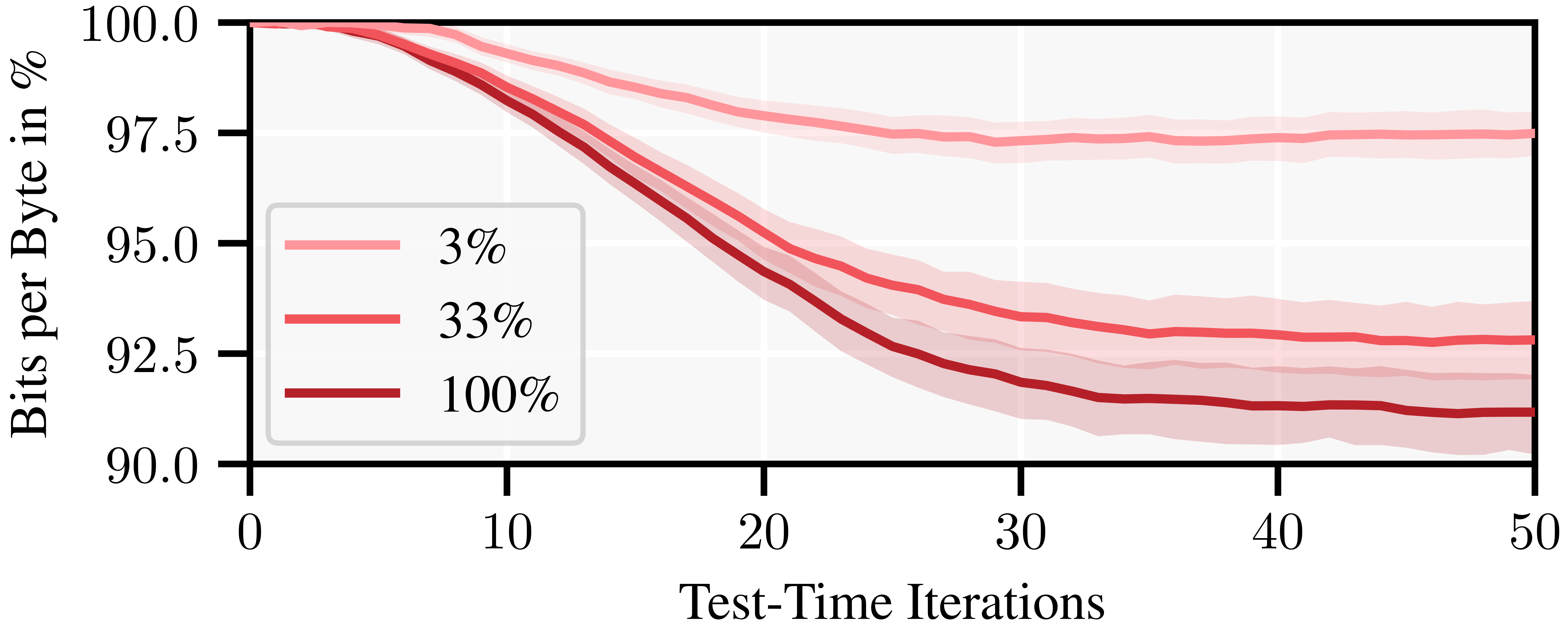
Does SIFT work?



	US	NN	NN-F	SIFT	Δ
NIH Grants	93.1 (1.1)	84.9 (2.1)	91.6 (16.7)	**53.8** (8.9)	↓31.1
US Patents	85.6 (1.5)	80.3 (1.9)	108.8 (6.6)	**62.9** (3.5)	↓17.4
GitHub	45.6 (2.2)	42.1 (2.0)	53.2 (4.0)	**30.0** (2.2)	↓12.1
Enron Emails	**68.6** (9.8)	**64.4** (10.1)	91.6 (20.6)	**53.1** (11.4)	↓11.3
Wikipedia	67.5 (1.9)	**66.3** (2.0)	121.2 (3.5)	**62.7** (2.1)	↓3.6
Common Crawl	92.6 (0.4)	90.4 (0.5)	148.8 (1.5)	**87.5** (0.7)	↓2.9
PubMed Abstr.	88.9 (0.3)	87.2 (0.4)	162.6 (1.3)	**84.4** (0.6)	↓2.8
ArXiv	85.4 (1.2)	**85.0** (1.6)	166.8 (6.4)	**82.5** (1.4)	↓2.5
PubMed Central	**81.7** (2.6)	**81.7** (2.6)	155.6 (5.1)	**79.5** (2.6)	↓2.2
Stack Exchange	78.6 (0.7)	78.2 (0.7)	141.9 (1.5)	**76.7** (0.7)	↓1.5
Hacker News	**80.4** (2.5)	**79.2** (2.8)	133.1 (6.3)	**78.4** (2.8)	↓0.8
FreeLaw	**63.9** (4.1)	**64.1** (4.0)	122.4 (7.1)	**64.0** (4.1)	↑0.1
DeepMind Math	**69.4** (2.1)	**69.6** (2.1)	121.8 (3.1)	**69.7** (2.1)	↑0.3
All	80.2 (0.5)	78.3 (0.5)	133.3 (1.2)	**73.5** (0.6)	↓4.8

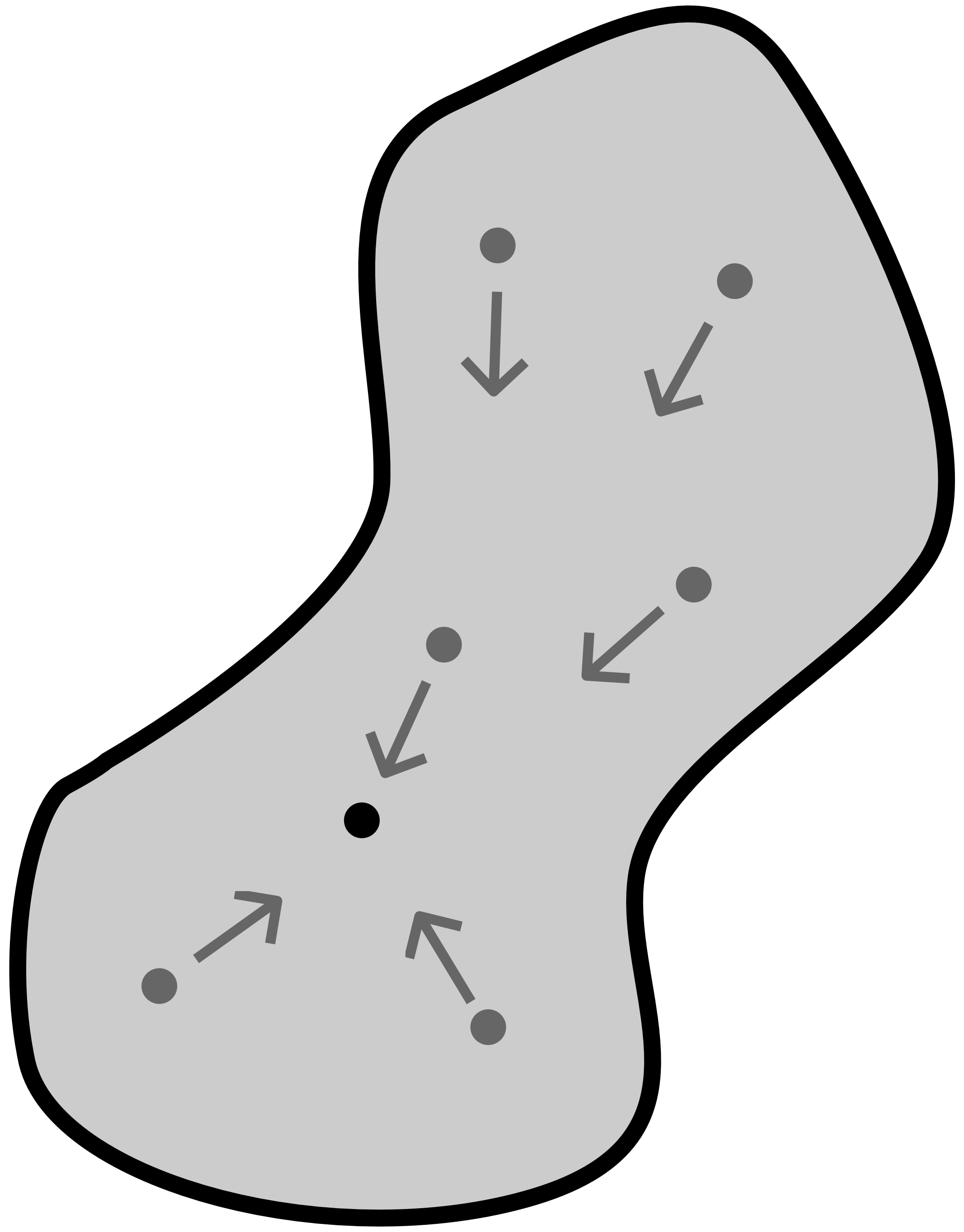
→ larger gains with stronger base models!

Does SIFT work?



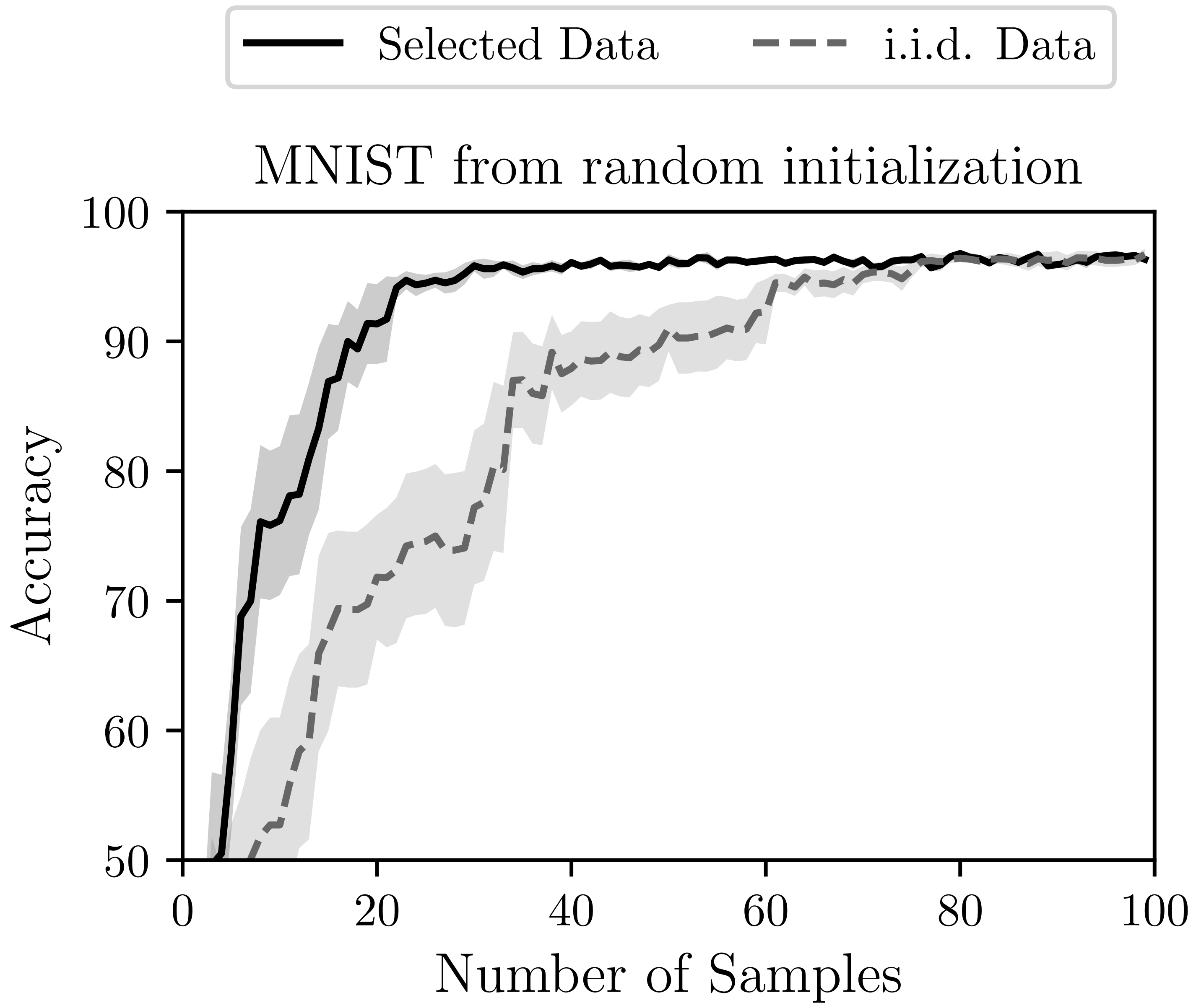
→ larger gains with larger “memory”!

Can we learn representations over time?



Strong representations can be bootstrapped!

[**H**, Sukhija, Treven, As, Krause; NeurIPS ’24]



Summary

Local models
solve one problem at a time

Inductive models (most current SOTA models)
attempt to solve all possible problems at once

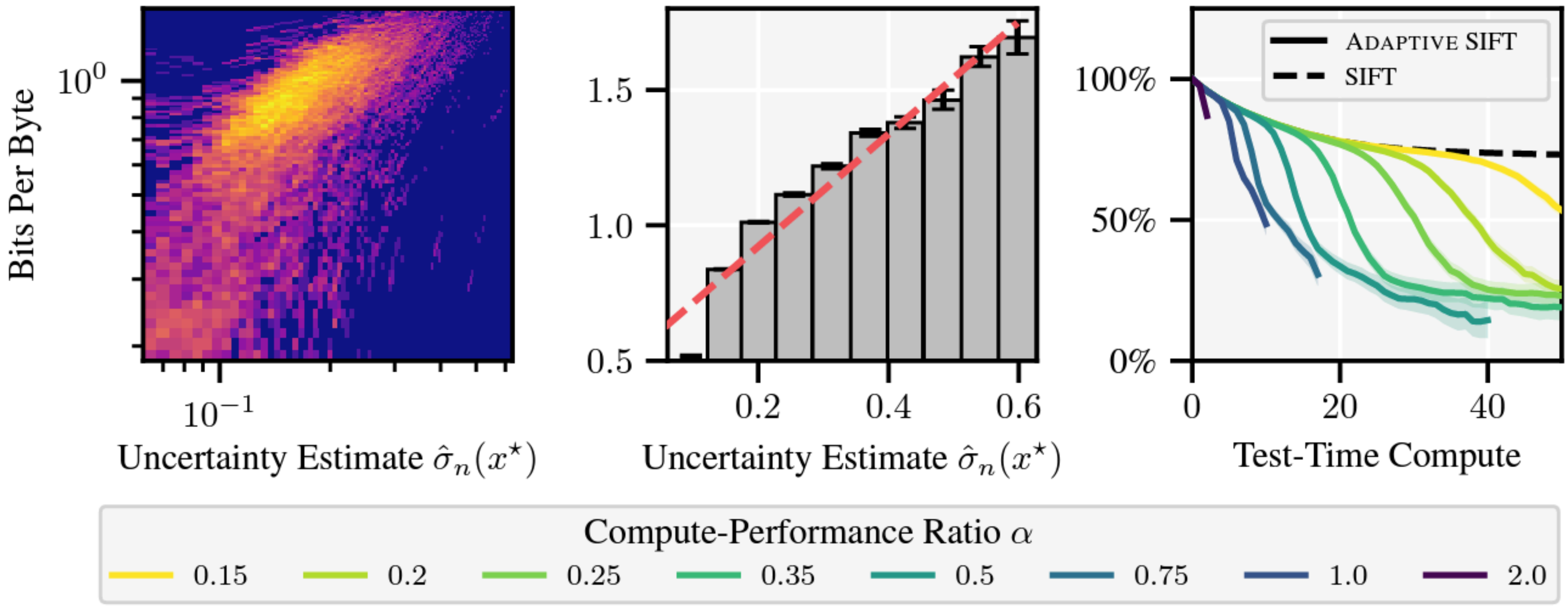
→ local learning allows allocating compute where it is “interesting”!

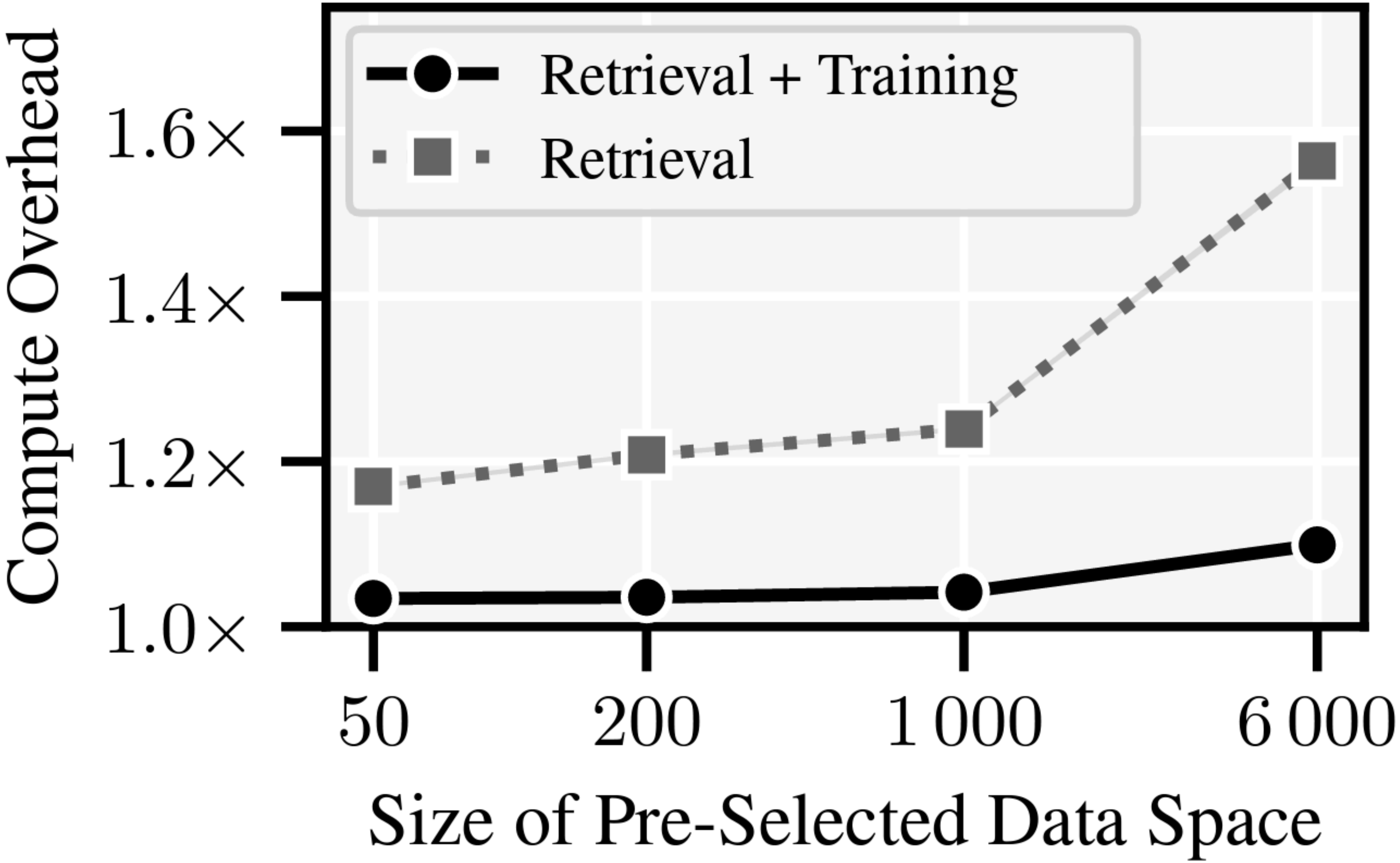
I’m happy to chat!

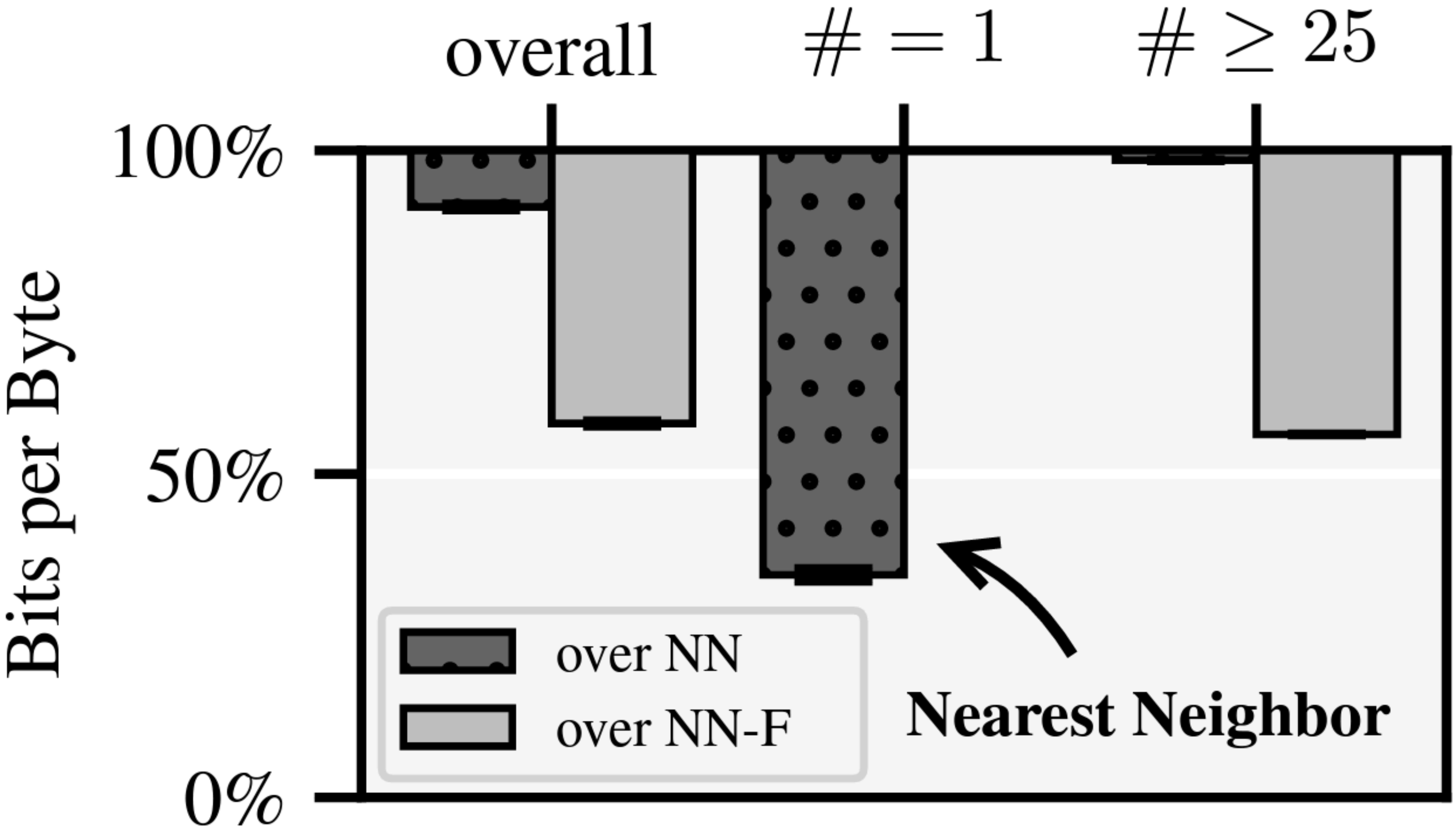
Jonas Hübotter
jonas.huebotter@inf.ethz.ch

- **Transductive Active Learning: Theory and Applications**
  NeurIPS ’24

- **Efficiently Learning at Test-Time: Active Fine-Tuning of LLMs**
  NeurIPS ’24 Workshop


Bits Per Byte
10^0
10^{-1}
Uncertainty Estimate $\hat{\sigma}_n(x^\star)$
0.5
1.0
1.5
0.2
0.4
0.6
Uncertainty Estimate $\hat{\sigma}_n(x^\star)$
Adaptive SIFT
SIFT
100%
50%
0%
0
20
40
Test-Time Compute
Compute-Performance Ratio α
0.15
0.2
0.25
0.35
0.5
0.75
1.0
2.0


Retrieval + Training
Retrieval
Compute Overhead
1.6×
1.4×
1.2×
1.0×
50
200
1 000
6 000
Size of Pre-Selected Data Space


overall
= 1
≥ 25
100%
50%
0%
Bits per Byte
over NN
over NN-F
Nearest Neighbor

Model	Bits per Byte	Bits per Byte (without Wikipedia)
Jurassic-1 (178B, Lieber et al., 2021)	n/a	0.601
GLM (130B, Zeng et al., 2022)	n/a	0.622
GPT-2 (124M, Radford et al., 2019)	1.241	
GPT-2 (774M, Radford et al., 2019)	1.093	
Llama-3.2-Instruct (1B)	0.807	
Llama-3.2-Instruct (3B)	0.737	
Gemma-2 (2B, Team et al., 2024)	0.721	
Llama-3.2 (1B)	0.697	
Phi-3 (3.8B, Abdin et al., 2024)	0.679	0.678
Phi-3 (7B, Abdin et al., 2024)	0.678	
Gemma-2 (9B, Team et al., 2024)	0.670	
GPT-3 (175B, Brown et al., 2020)	0.666	
Phi-3 (14B, Abdin et al., 2024)	0.651	
Llama-3.2 (3B)	0.640	
Gemma-2 (27B, Team et al., 2024)	0.629	
Test-Time FT with SIFT + GPT-2 (124M)	0.862	
Test-Time FT with SIFT + GPT-2 (774M)	0.762	
Test-Time FT with SIFT + Phi-3 (3.8B)	**0.595**	**0.599**

Table 2: Evaluation of state-of-the-art models on the Pile language modeling benchmark, without copyrighted datasets. Results with GPT-3 are from Gao et al. (2020). Results with Jurassic-1 and GLM are from Zeng et al. (2022) and do not report on the Wikipedia dataset. For a complete comparison, we also evaluate our Phi-3 with test-time fine-tuning when excluding the Wikipedia dataset.