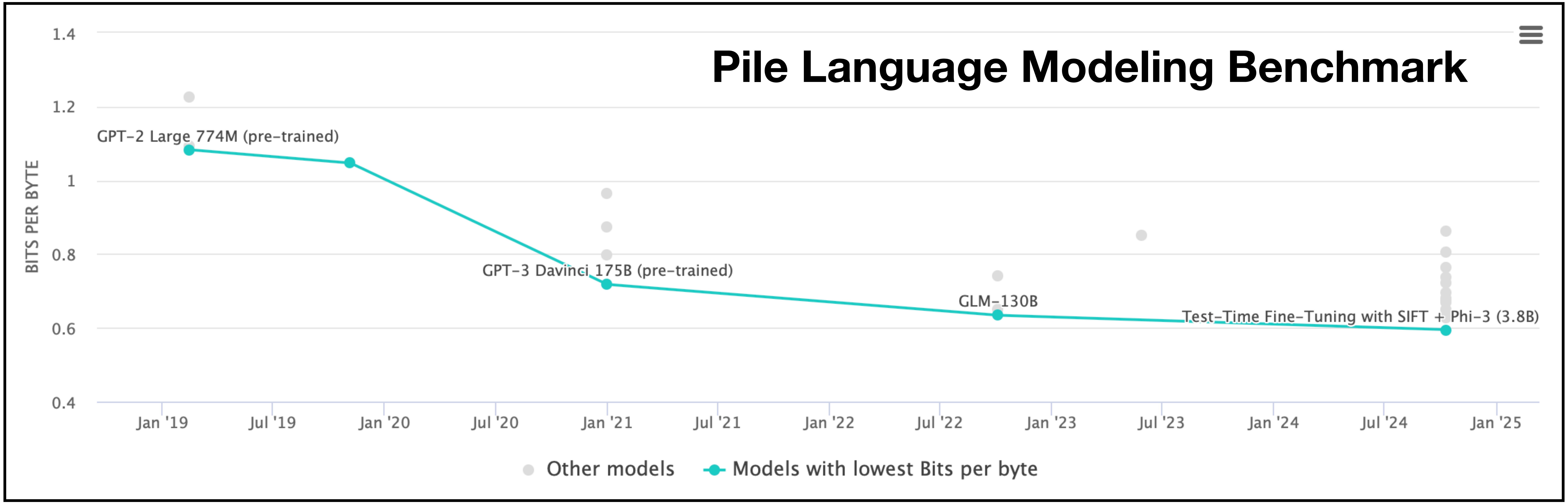
Efficiently Learning at Test-Time with LLMs

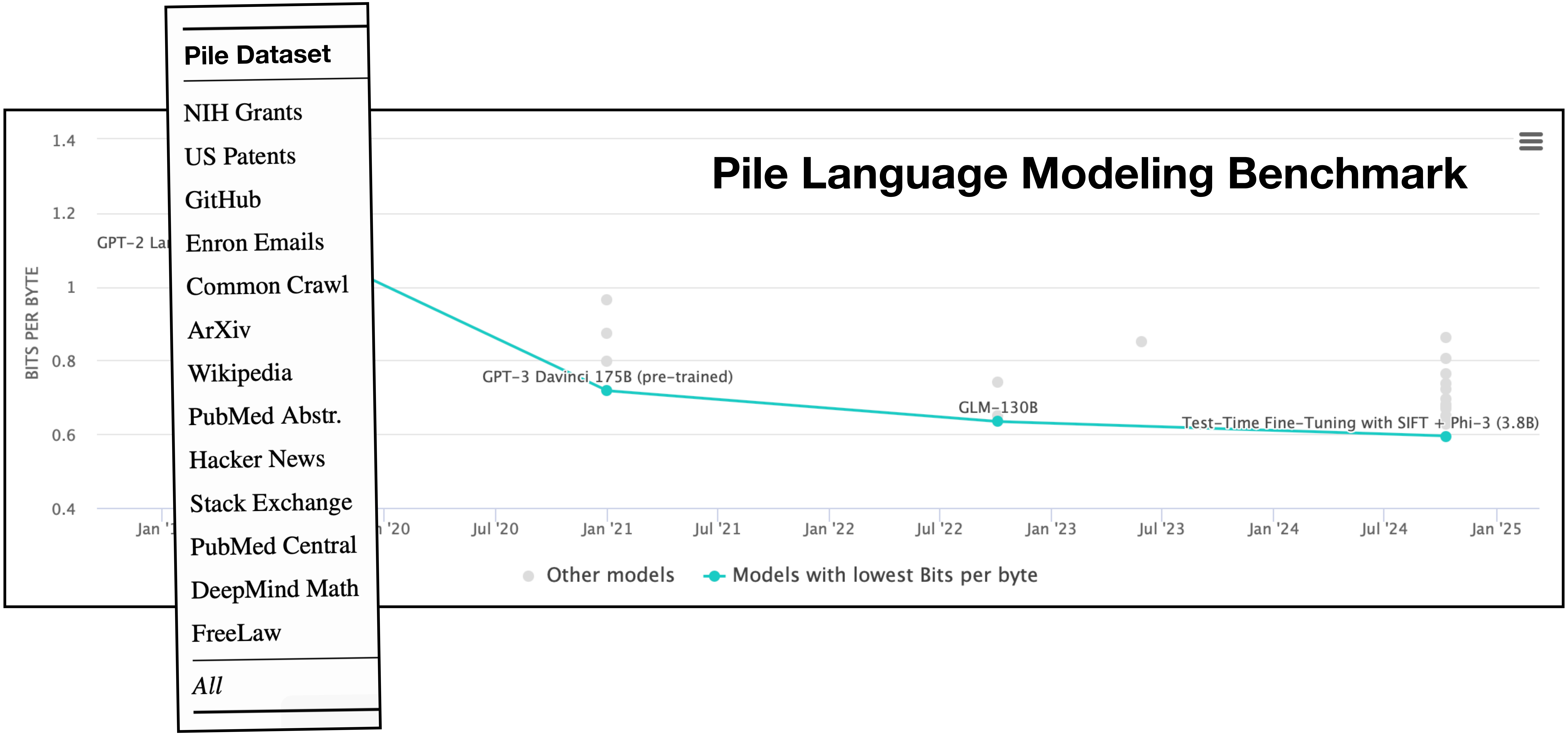
Jonas Hübotter

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



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



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

Training Data		Learnt Model		Prediction
$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$	$\longrightarrow$	$f : \mathcal{X} \to \mathcal{Y}$	$\longrightarrow$	$f(\boldsymbol{x}^\star)$

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

Training Data

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

Learnt Model

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

Prediction

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

Train | **Test**

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

Training Data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ → Learnt Model $f : \mathcal{X} \to \mathcal{Y}$ → Prediction $f(\boldsymbol{x}^\star)$

Train | **Test**

known!

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

Training Data

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

Learnt Model

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

Prediction

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

Local Learning (at Test-Time)

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

Training Data

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

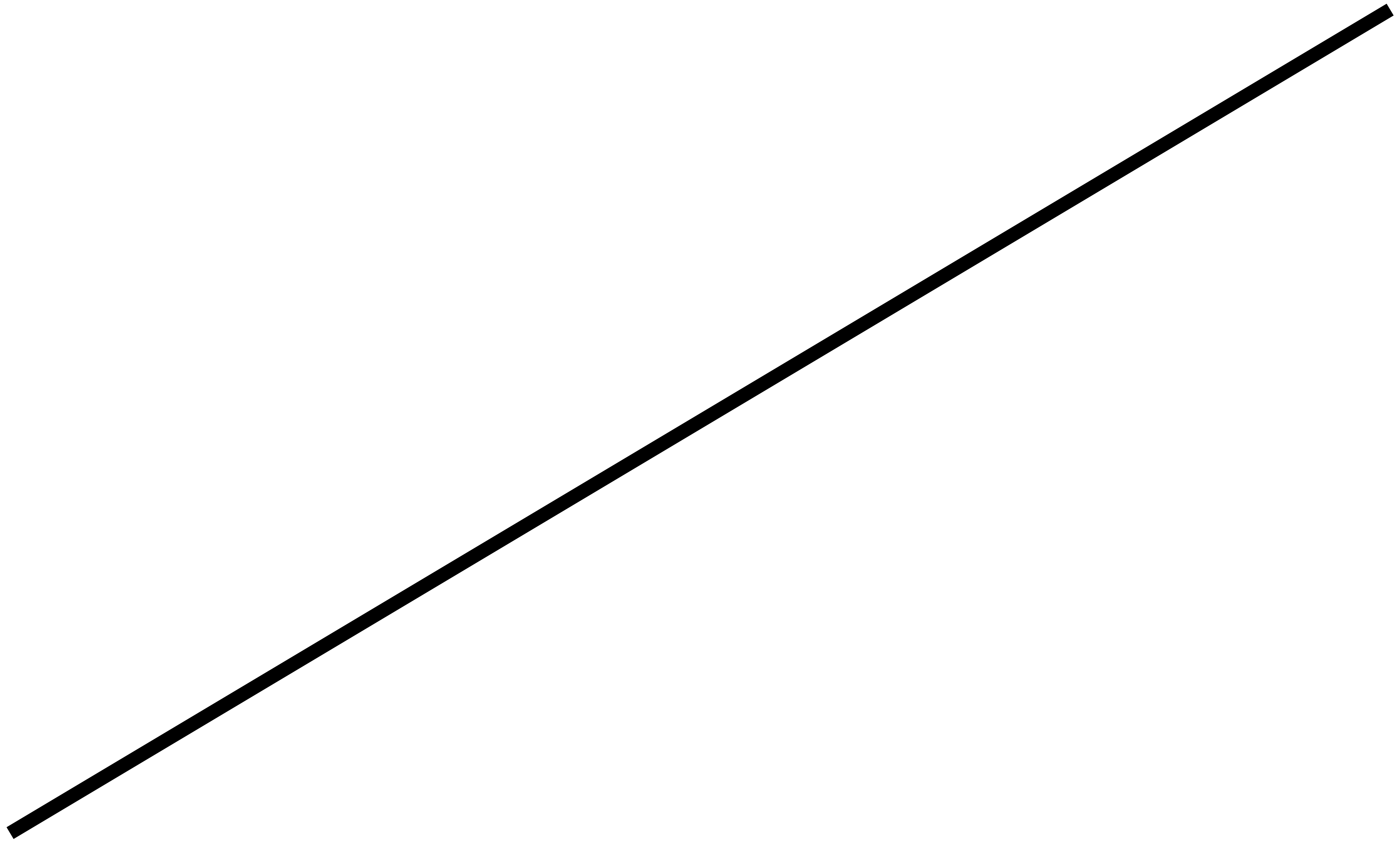
Learnt Model

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

Prediction

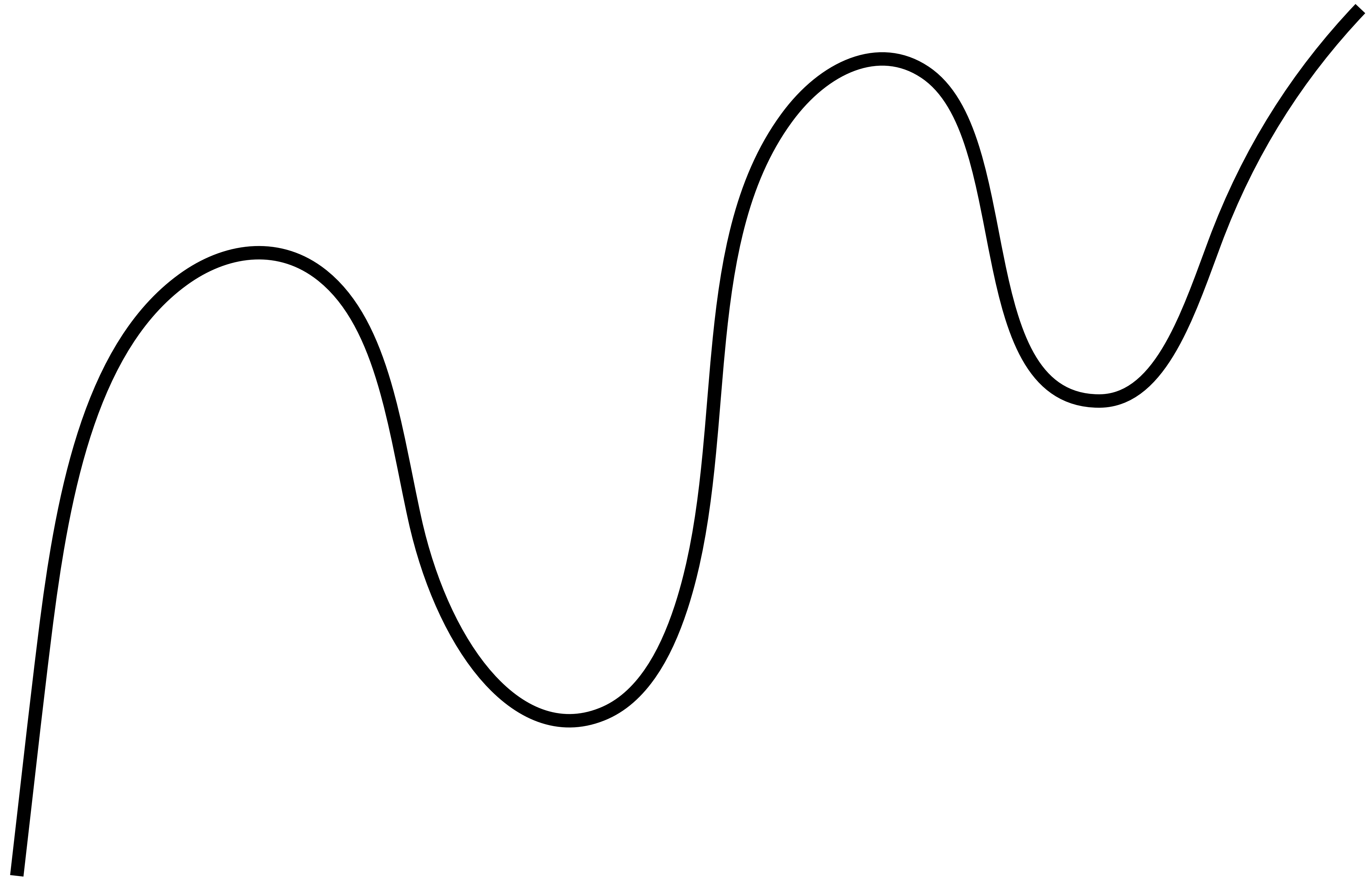
$f(\boldsymbol{x}^\star)$

A Story of Curve Fitting

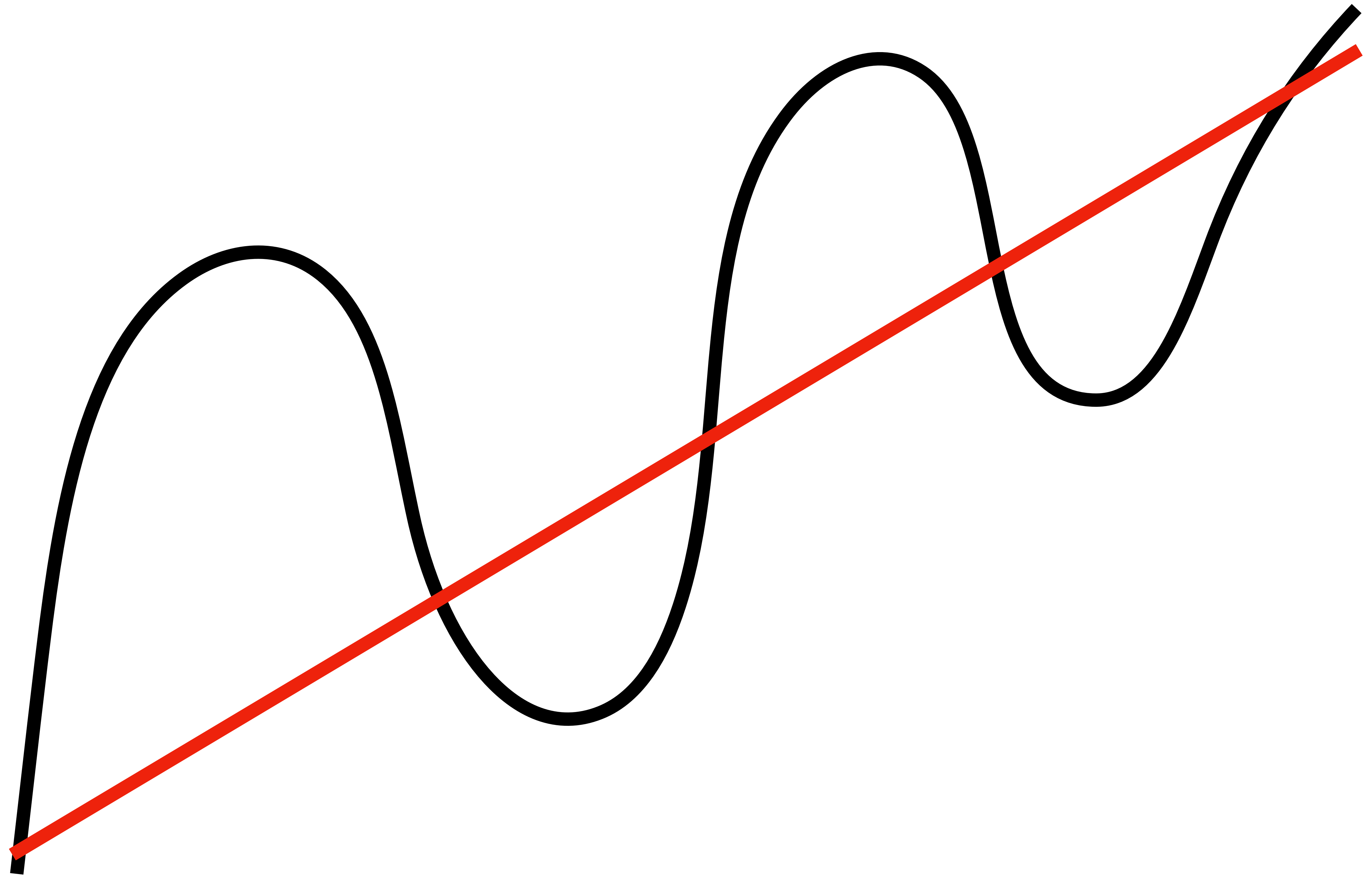


A Story of Curve Fitting

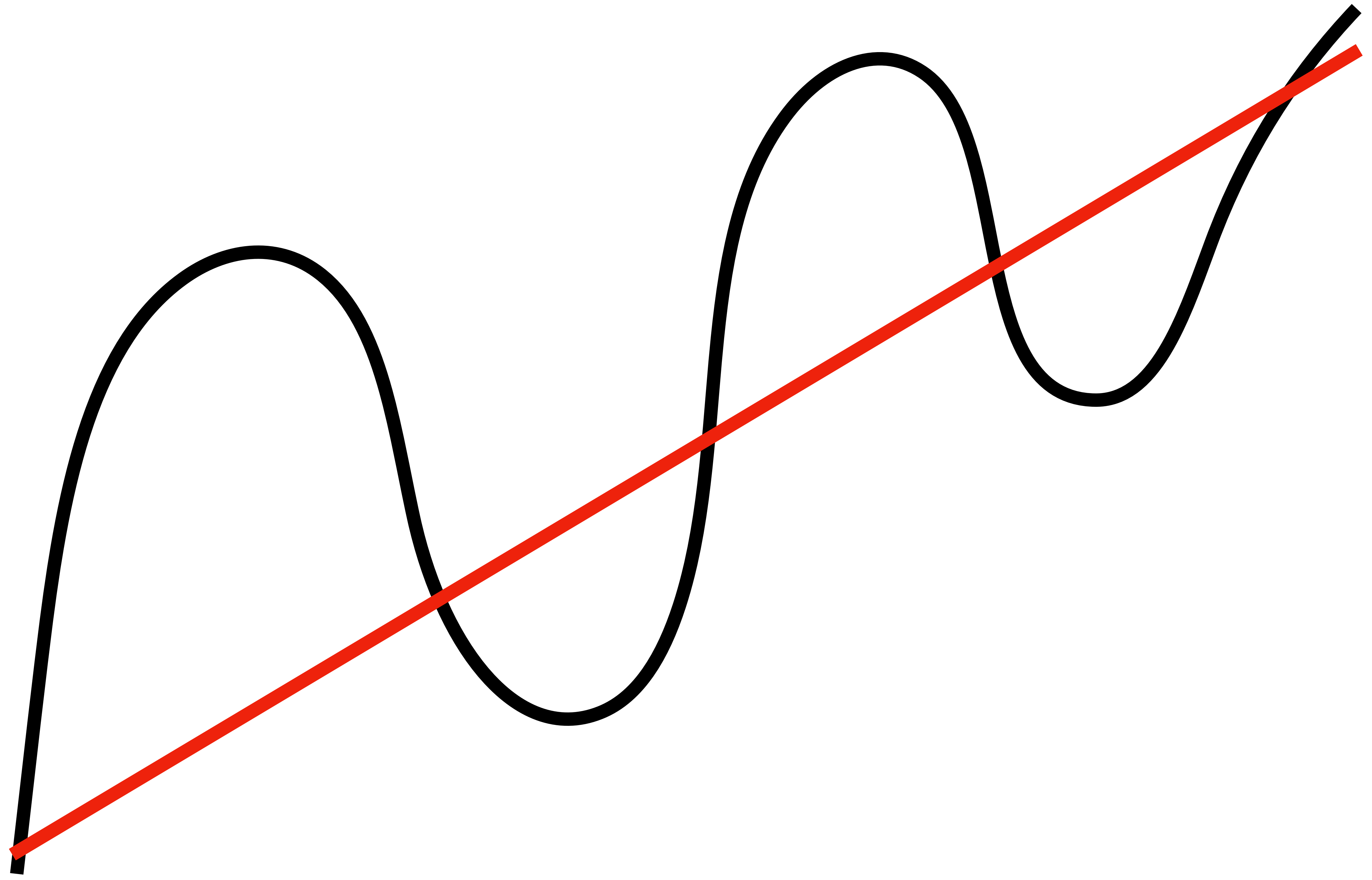
A Story of Curve Fitting



A Story of Curve Fitting

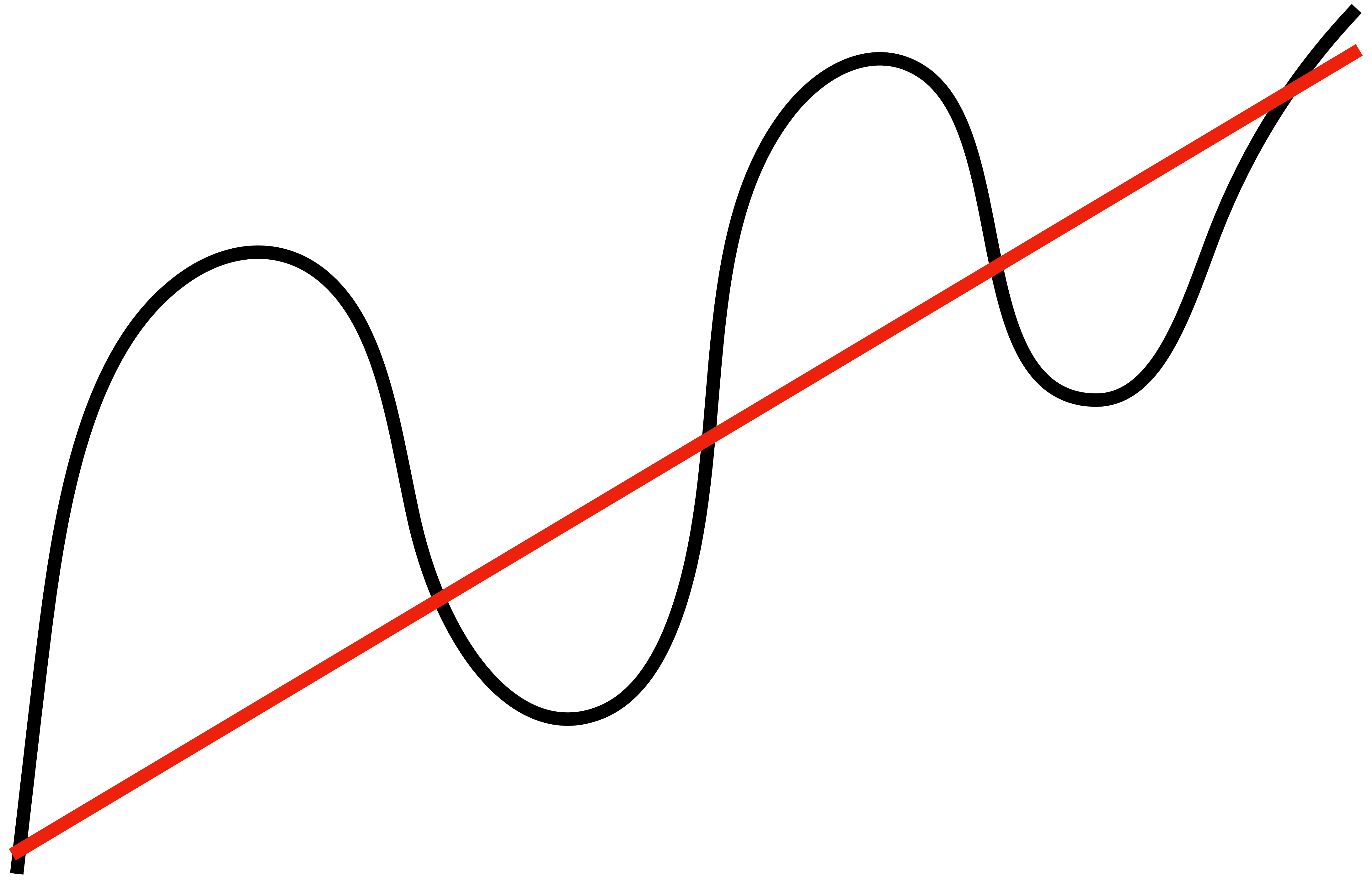


A Story of Curve Fitting



Remedies:

A Story of Curve Fitting



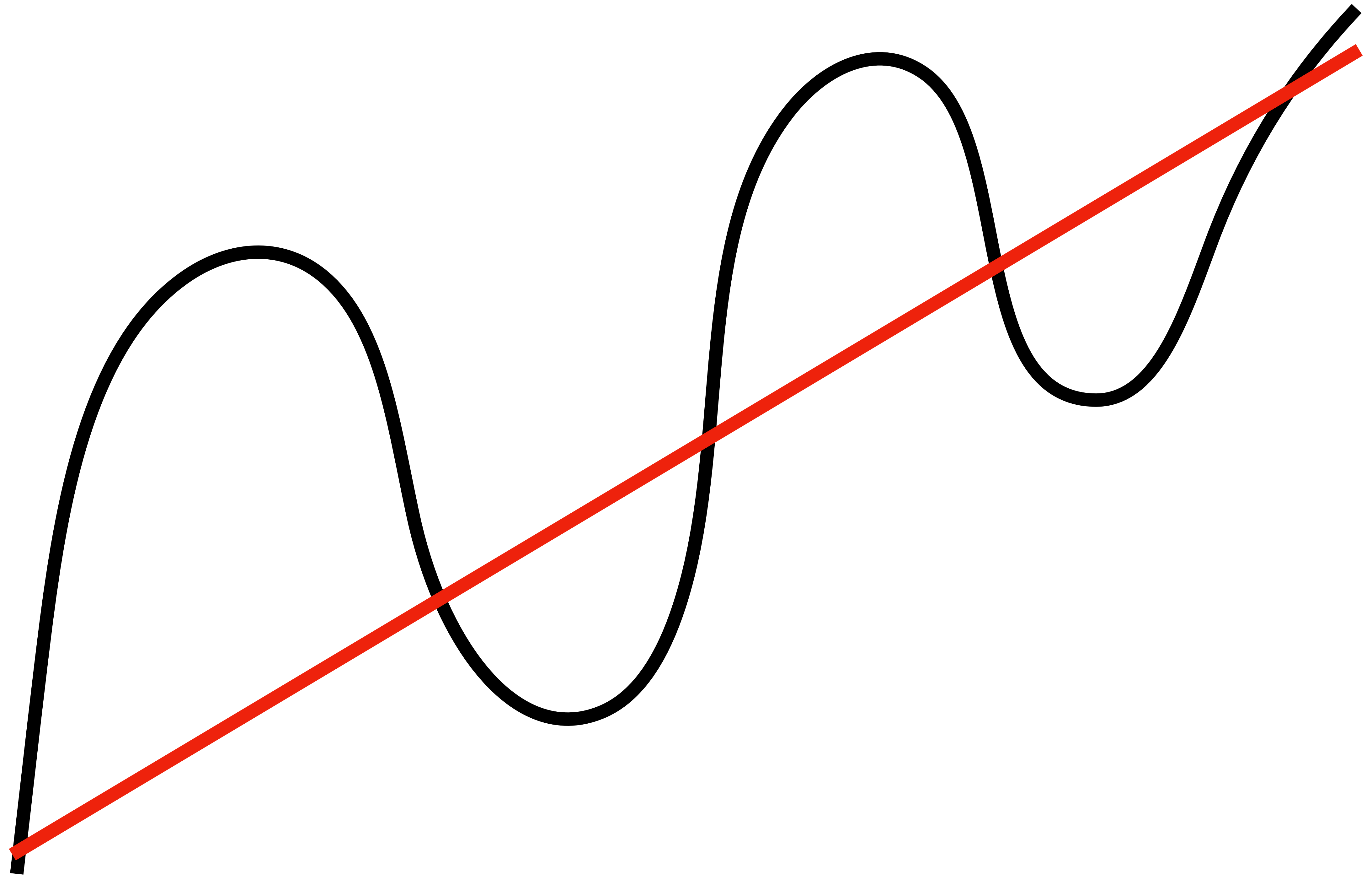
Remedies:

- Parametric models

  polynomial regression

  neural networks

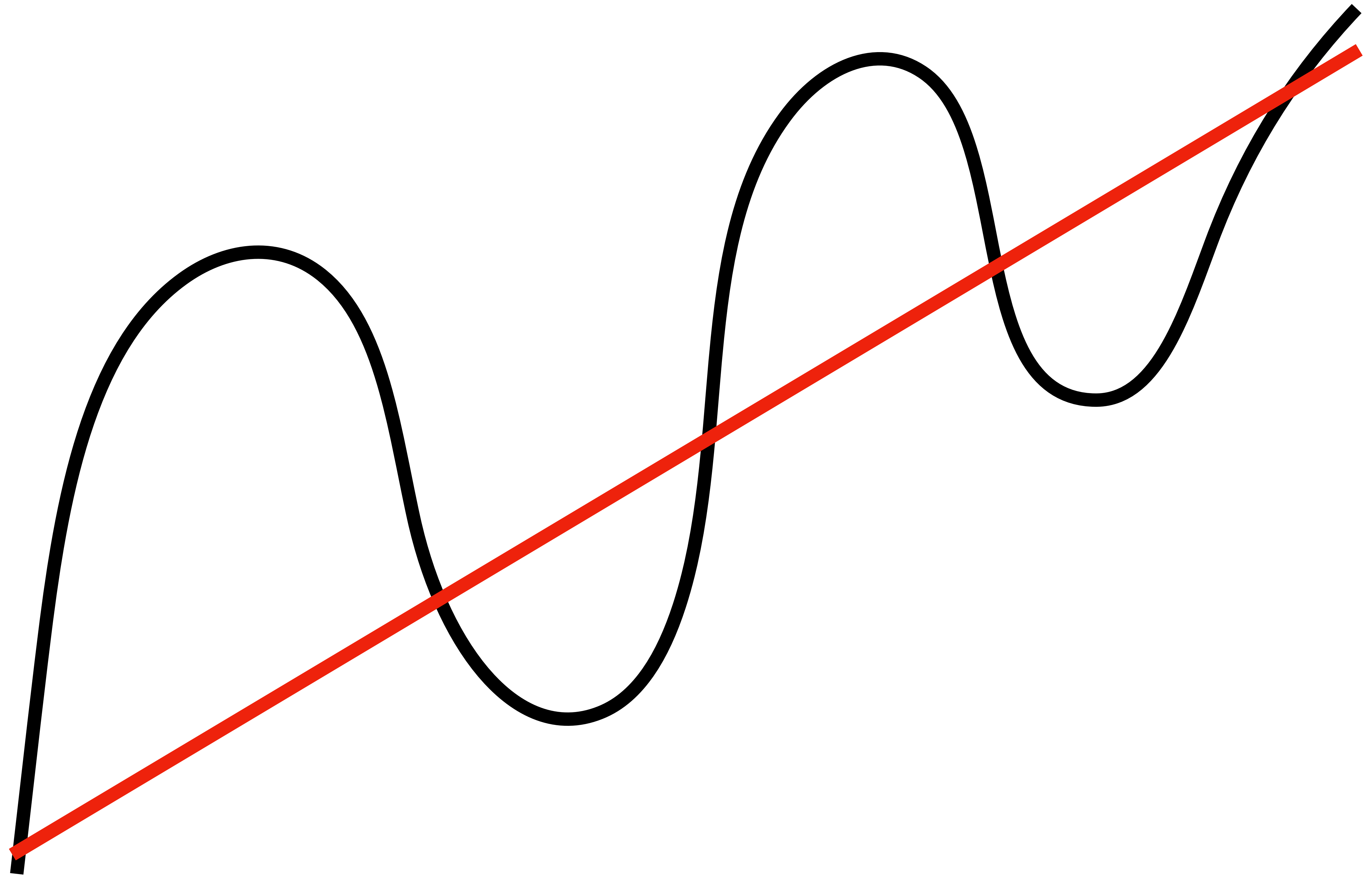
A Story of Curve Fitting



Remedies:

- Parametric models
  - polynomial regression
  - neural networks
- Non-parametric models
  - kernel (ridge) regression
  - k-nearest neighbor

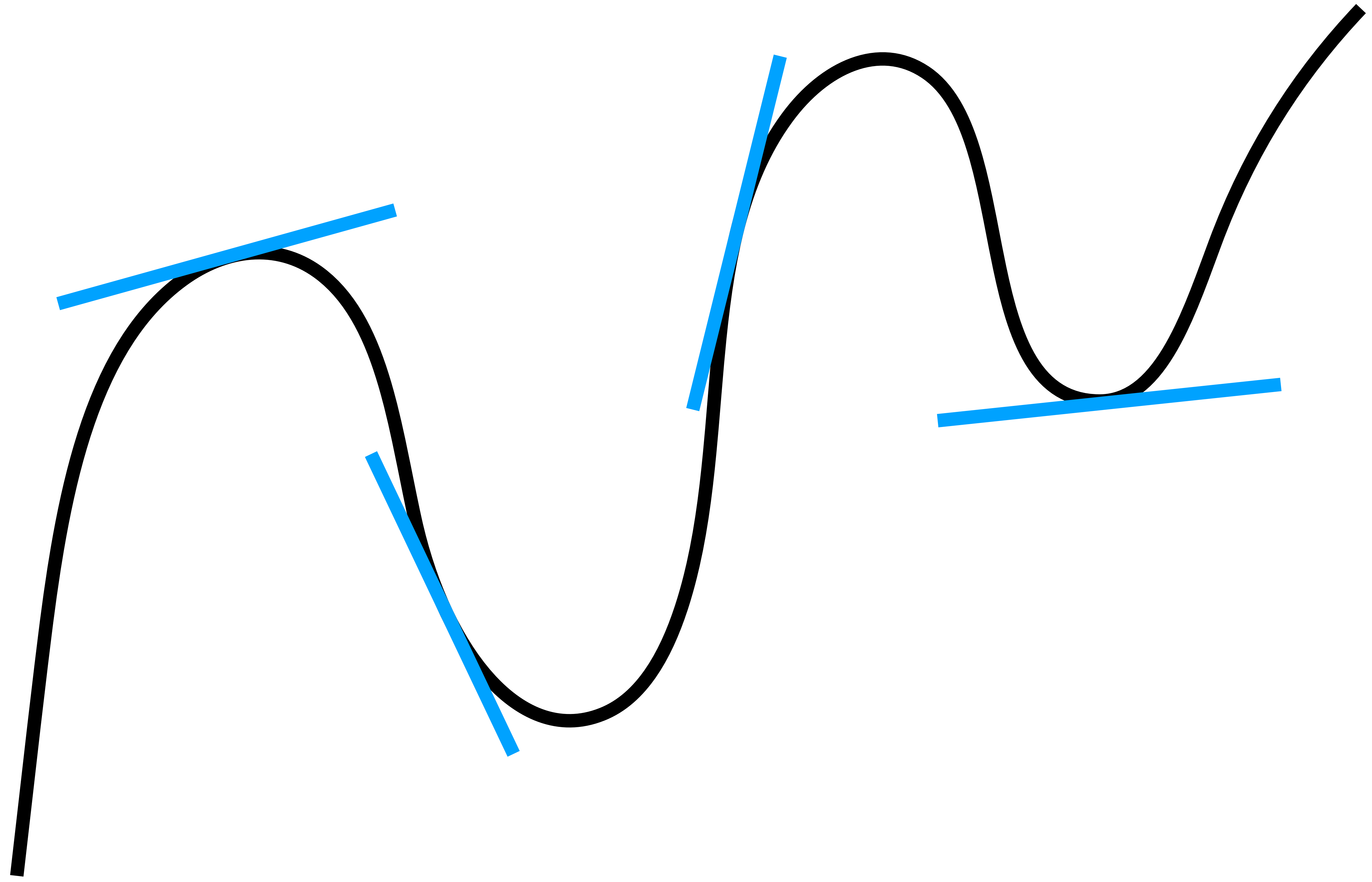
A Story of Curve Fitting



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

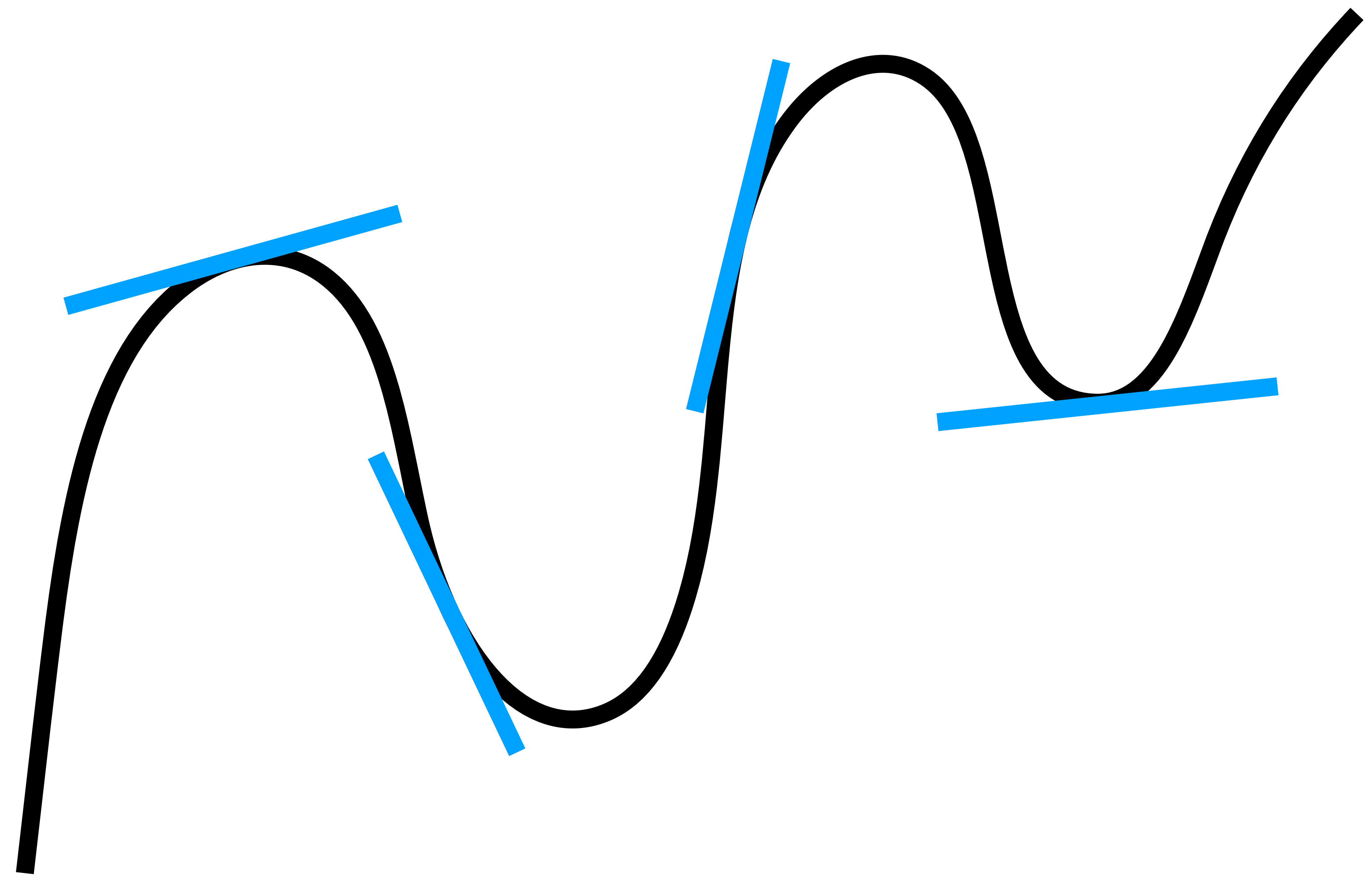


Remedies:

- 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:



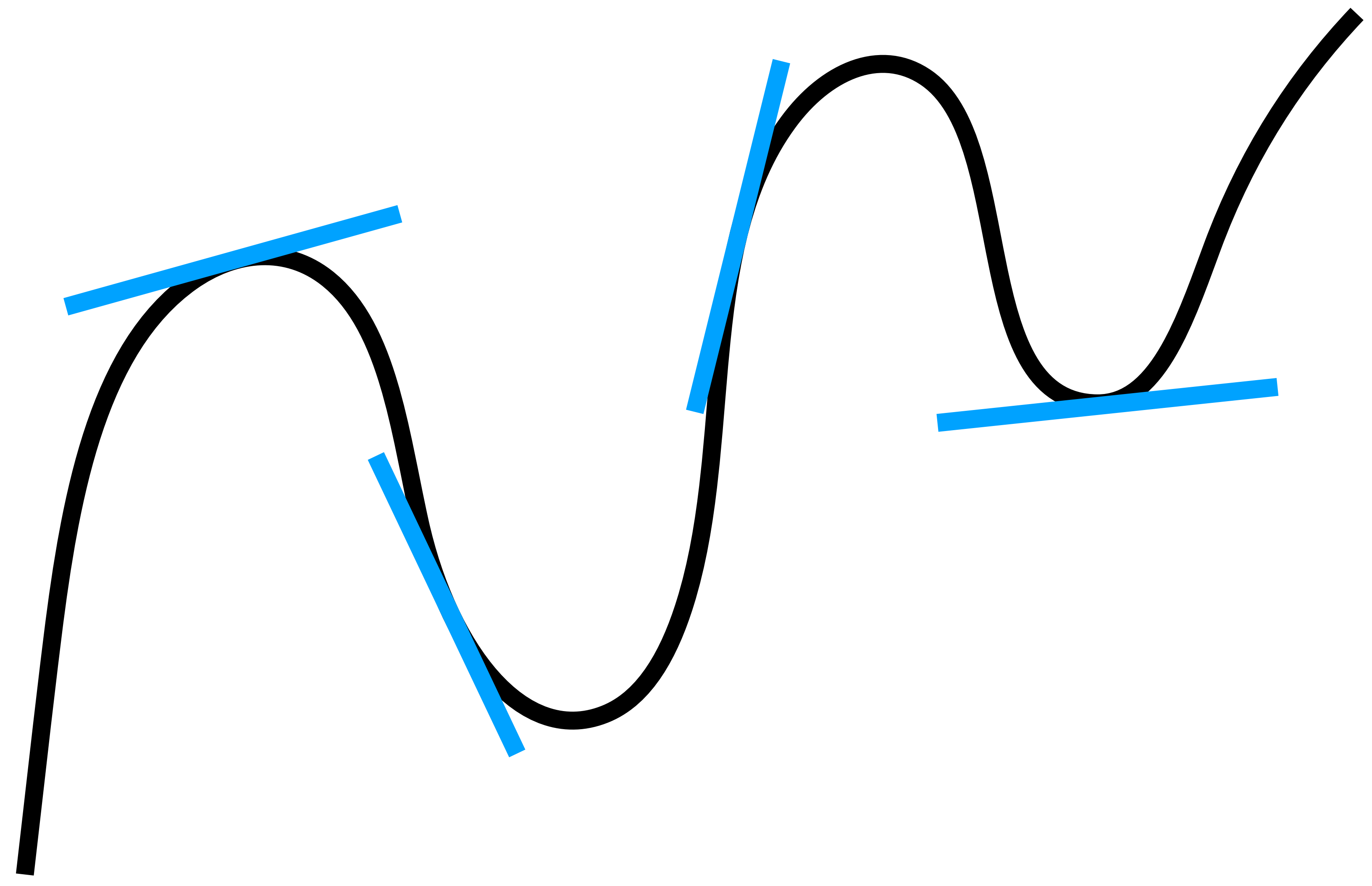
A Story of Curve Fitting

Local models have two components:

- *Parametric* “controller”

  linear regression

  …



A Story of Curve Fitting

Local models have two components:

- *Parametric* “controller”

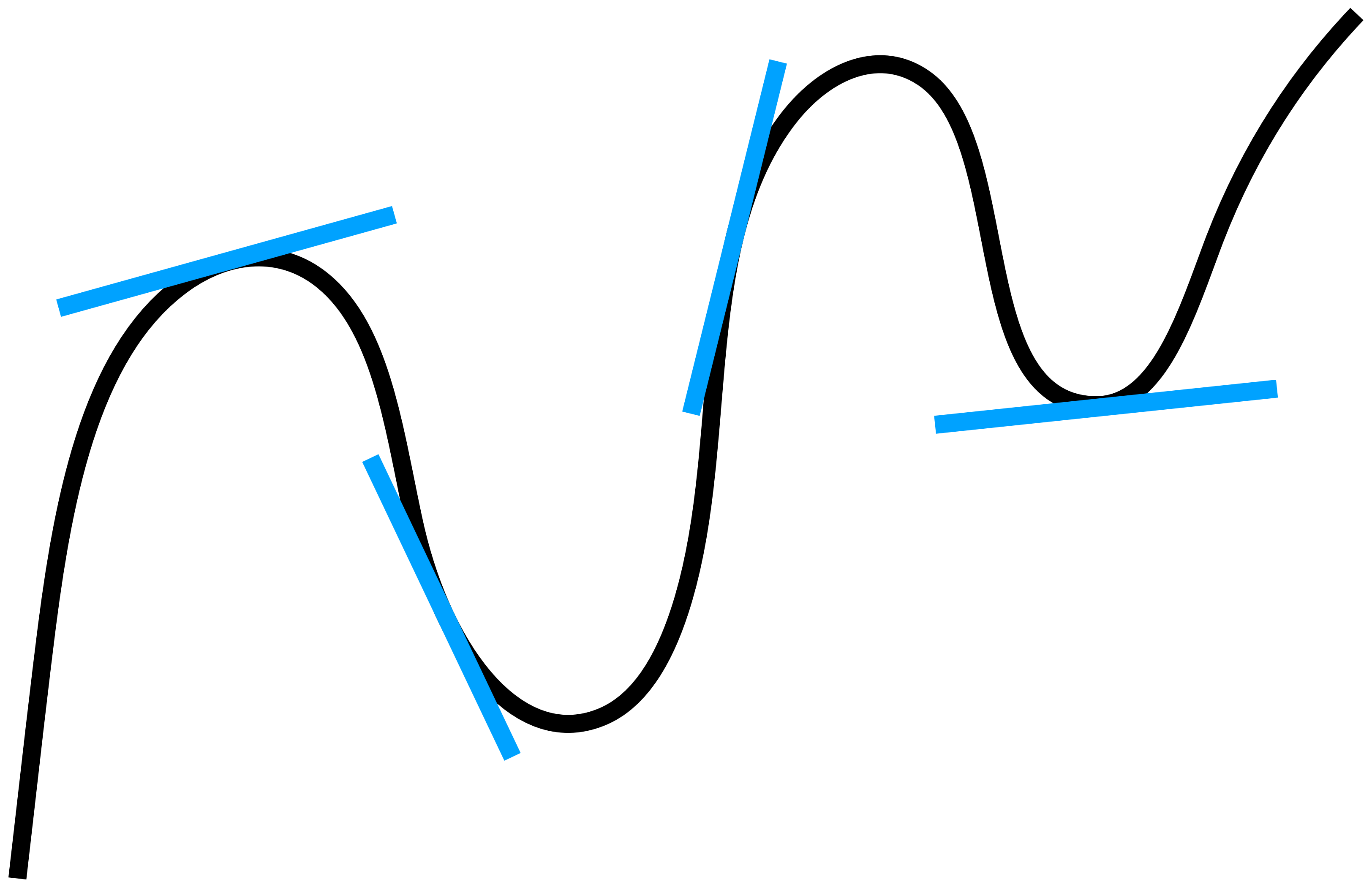
  linear regression

  …

- *Non-parametric* “memory”

  k-nearest neighbor

  …



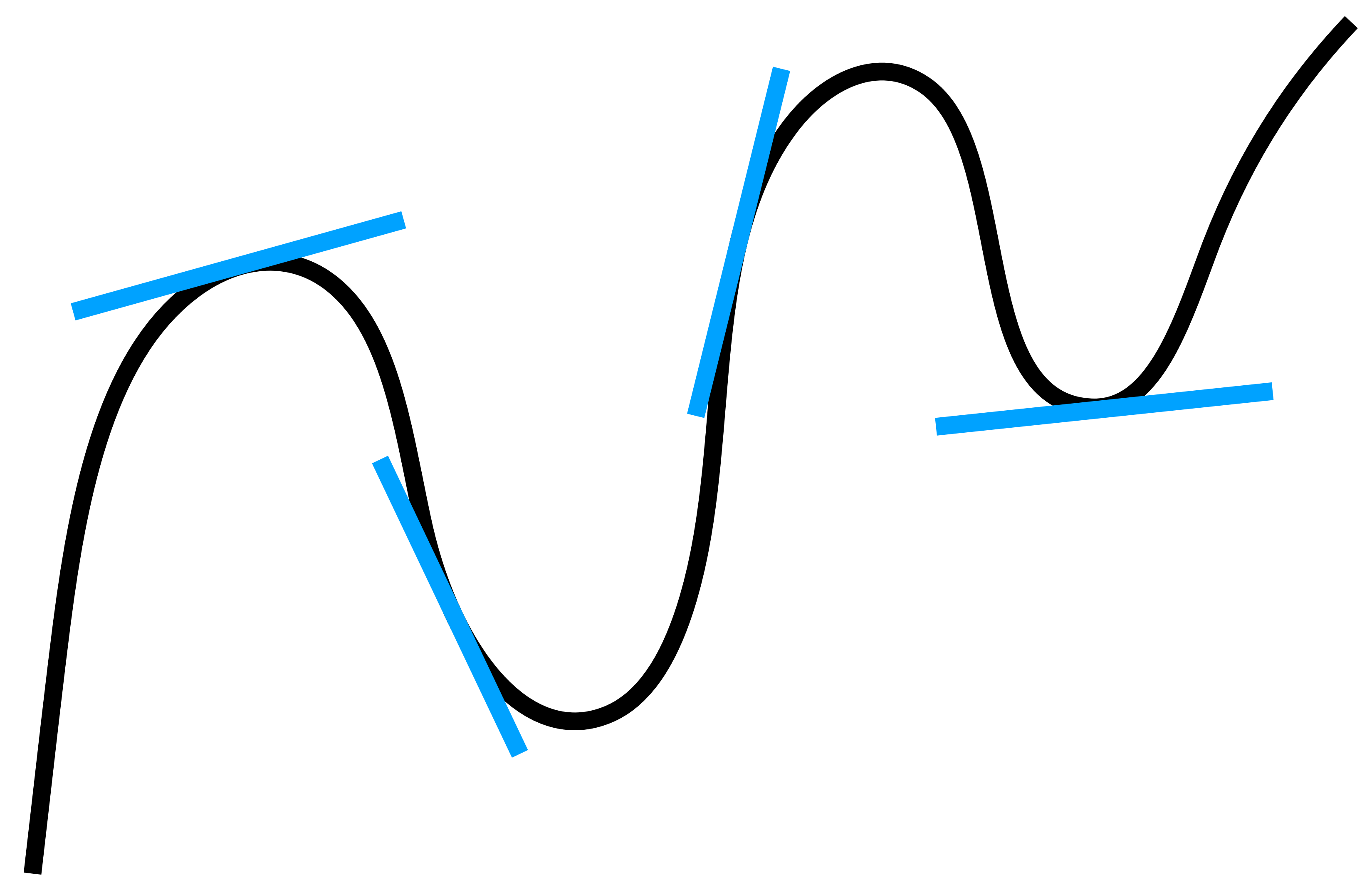
A Story of Curve Fitting

Local models have two components:

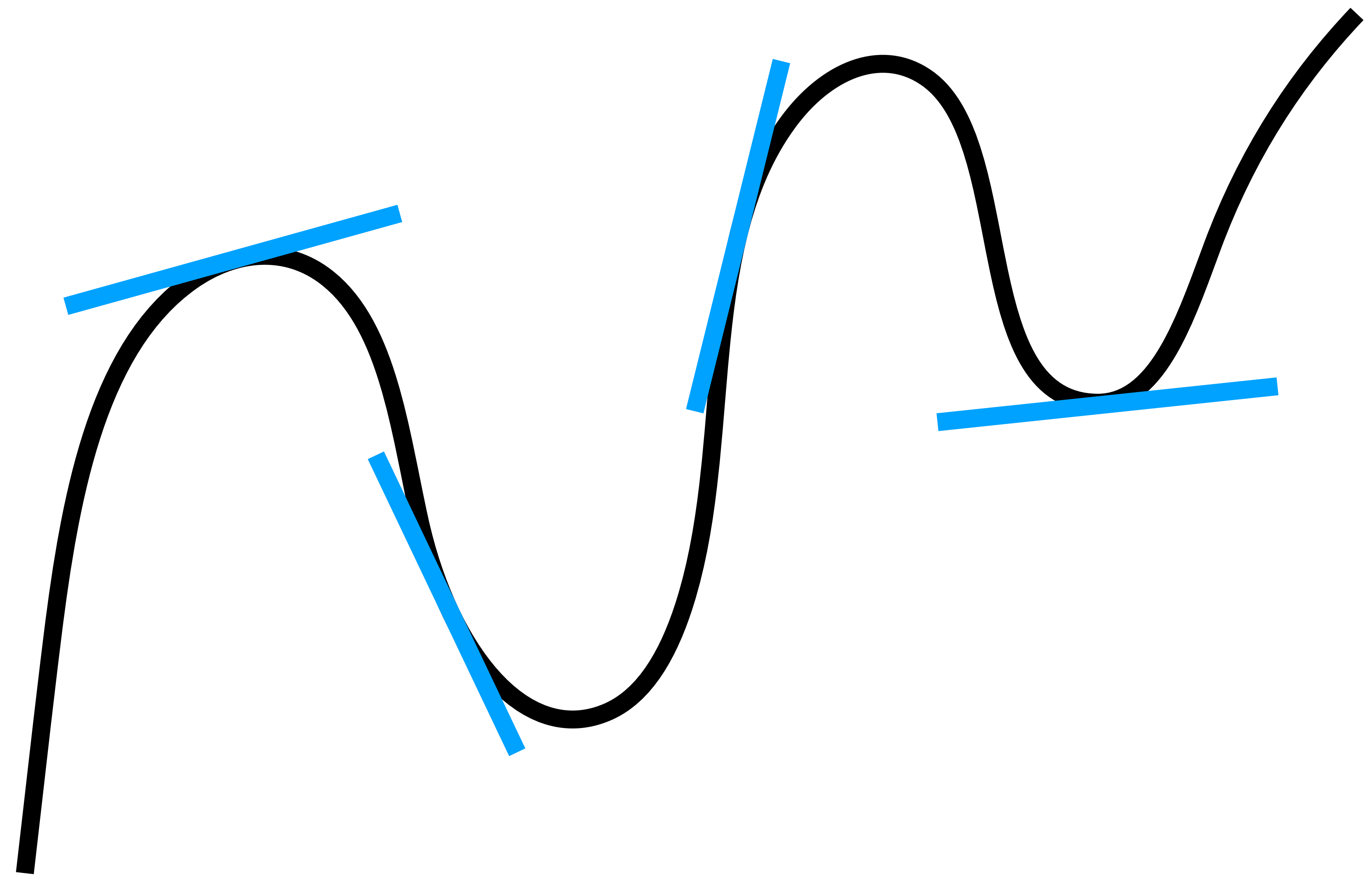
- *Parametric* “controller”
  linear regression
  …

- *Non-parametric* “memory”
  k-nearest neighbor
  …

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



A Story of Curve Fitting



Local models have two components:

- *Parametric* “controller”

  linear regression

  …

- *Non-parametric* “memory”

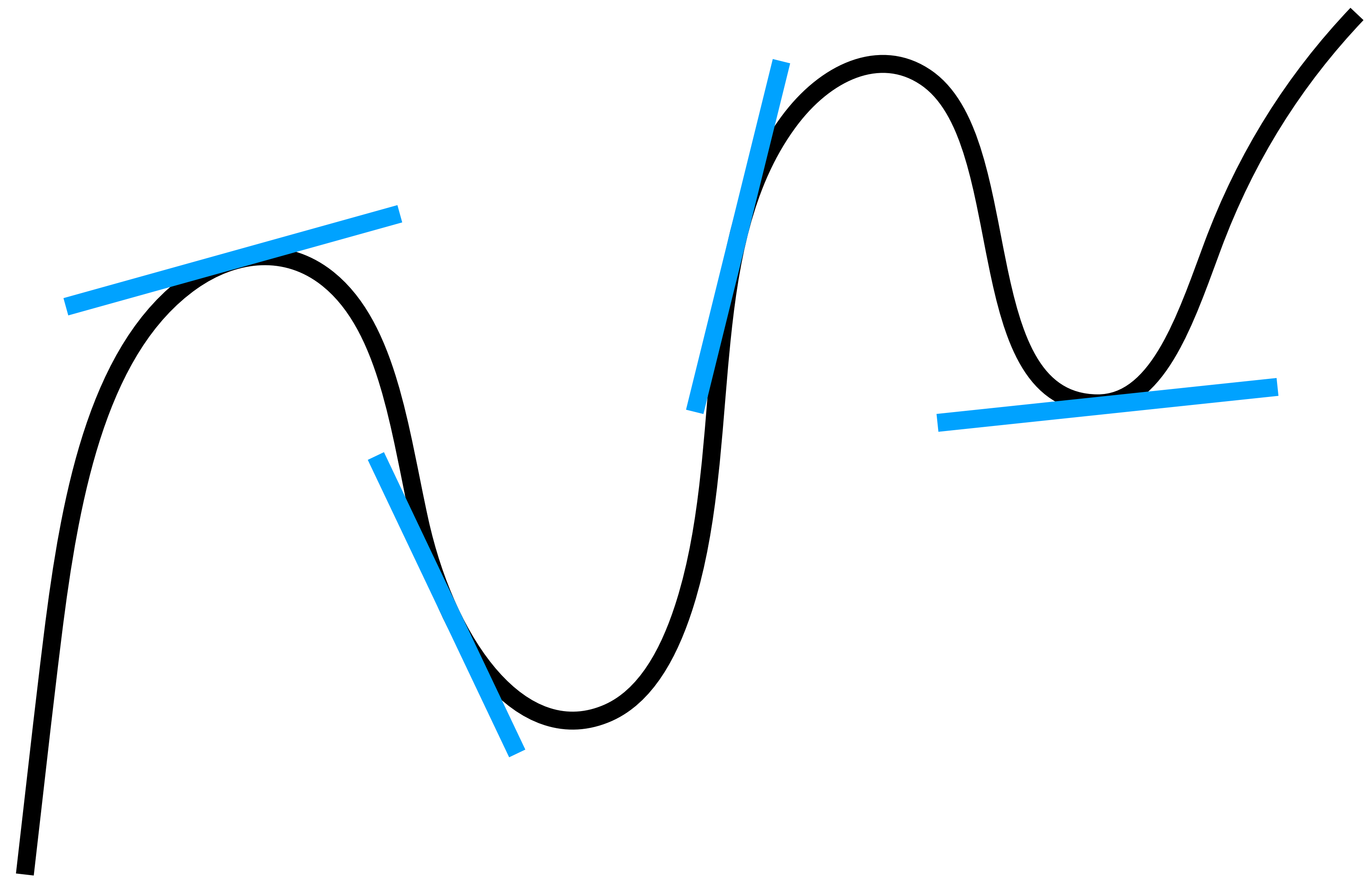
  k-nearest neighbor

  …

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

→ <u>one</u> local model needs only little data!

A Story of Curve Fitting



Local models have two components:

- *Parametric* “controller”
  linear regression
  …

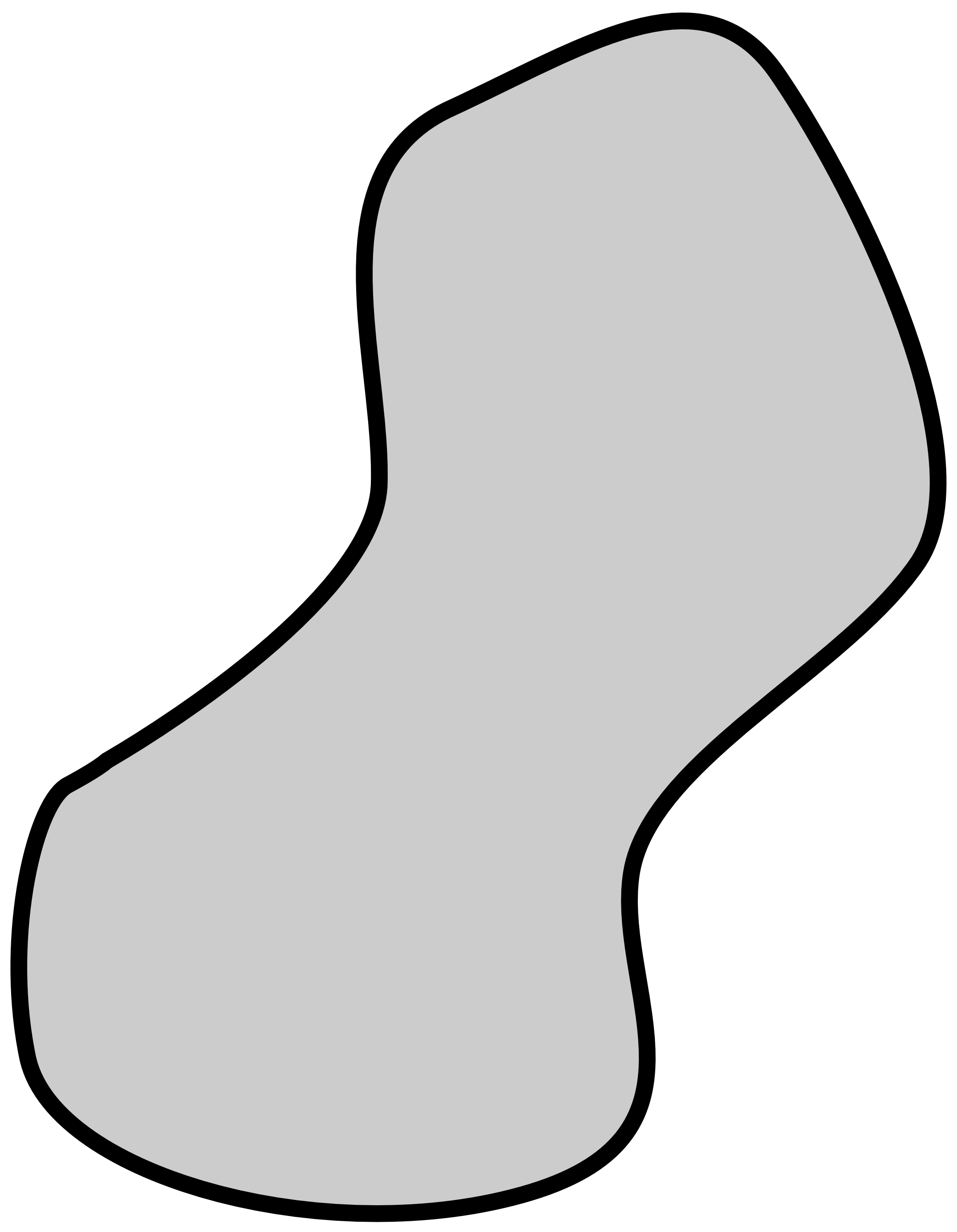
- *Non-parametric* “memory”
  k-nearest neighbor
  …

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

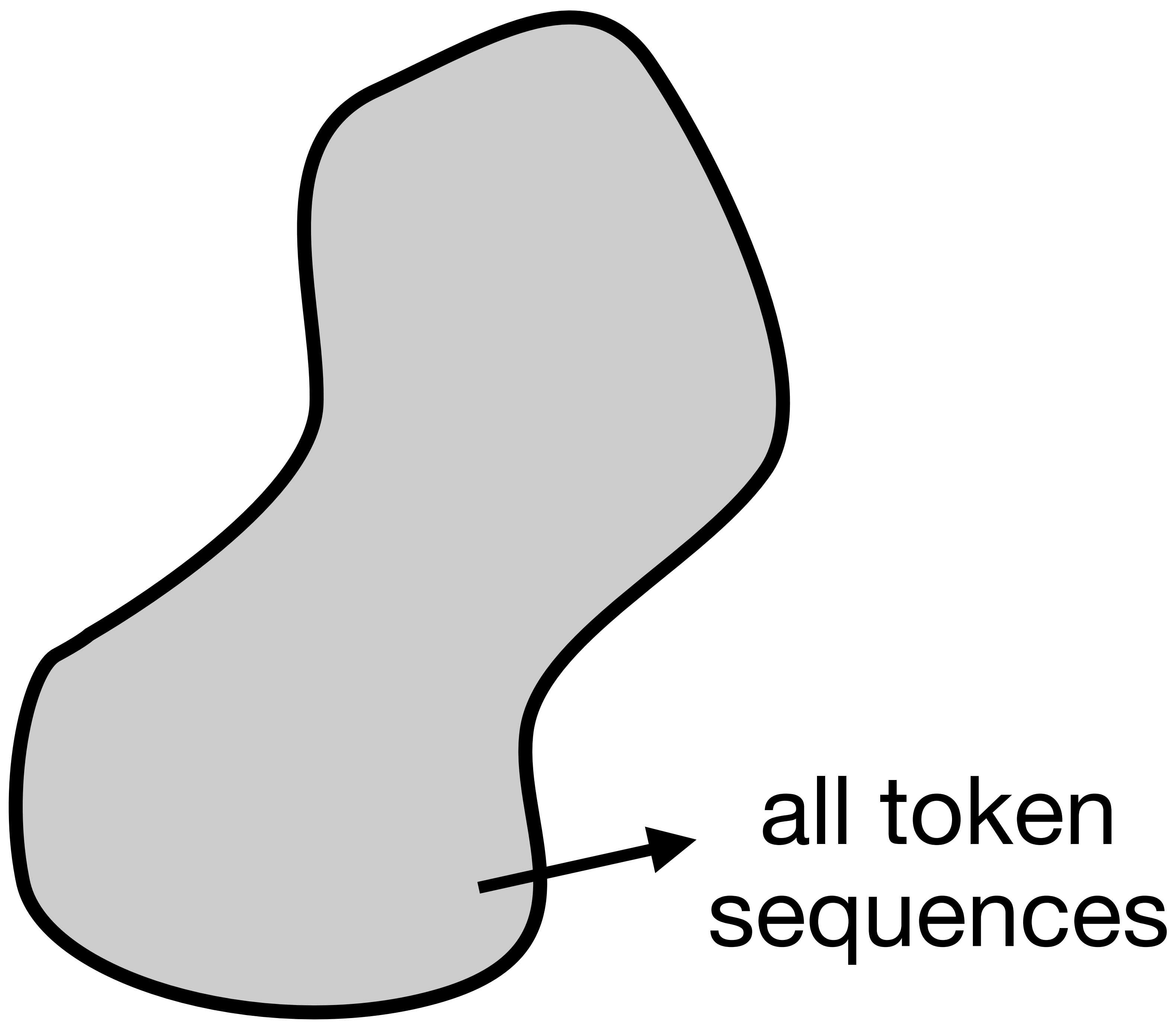
→ <u>one</u> local model needs only little data!

→ too good to be true?

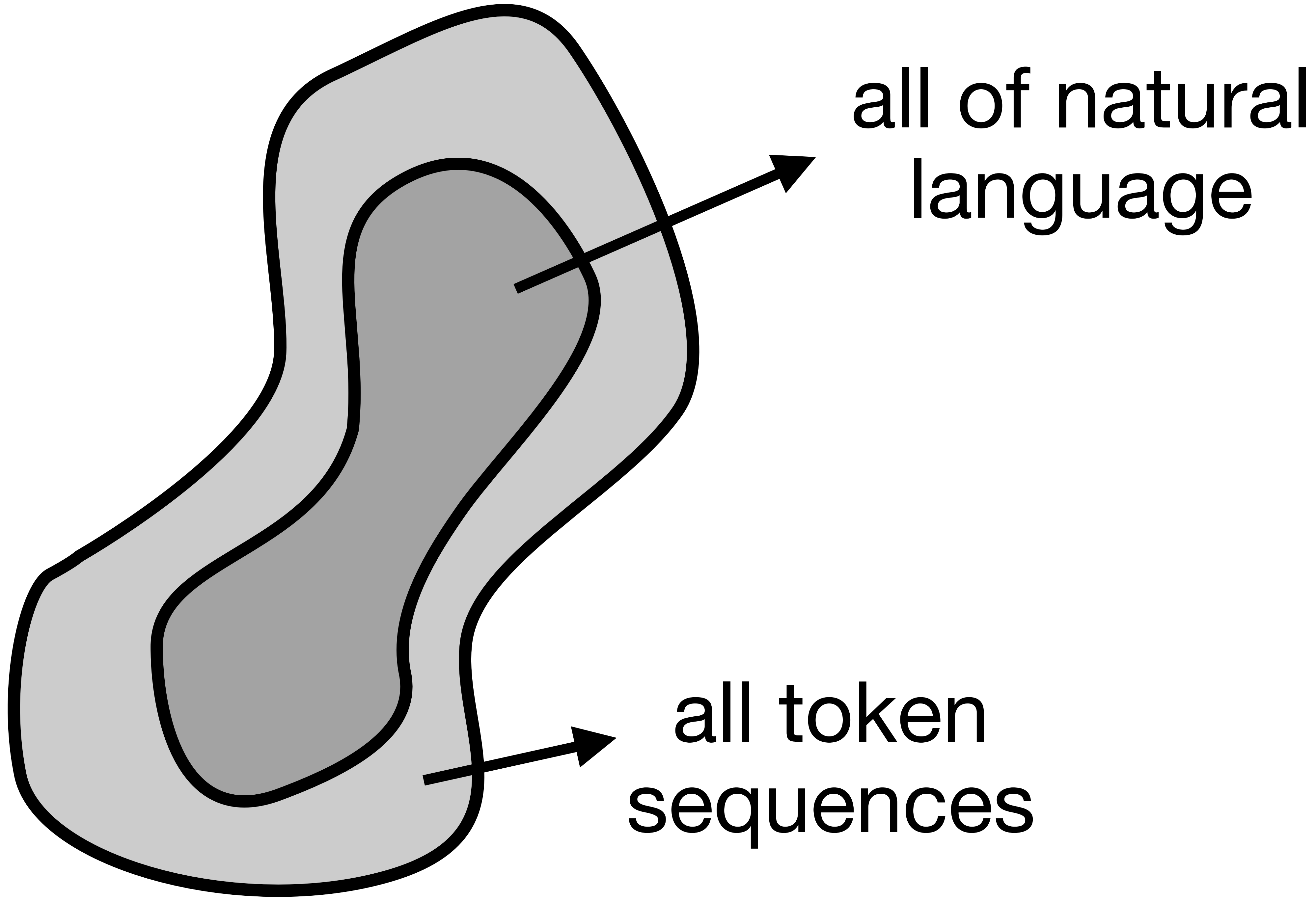
Local Learning in a Picture



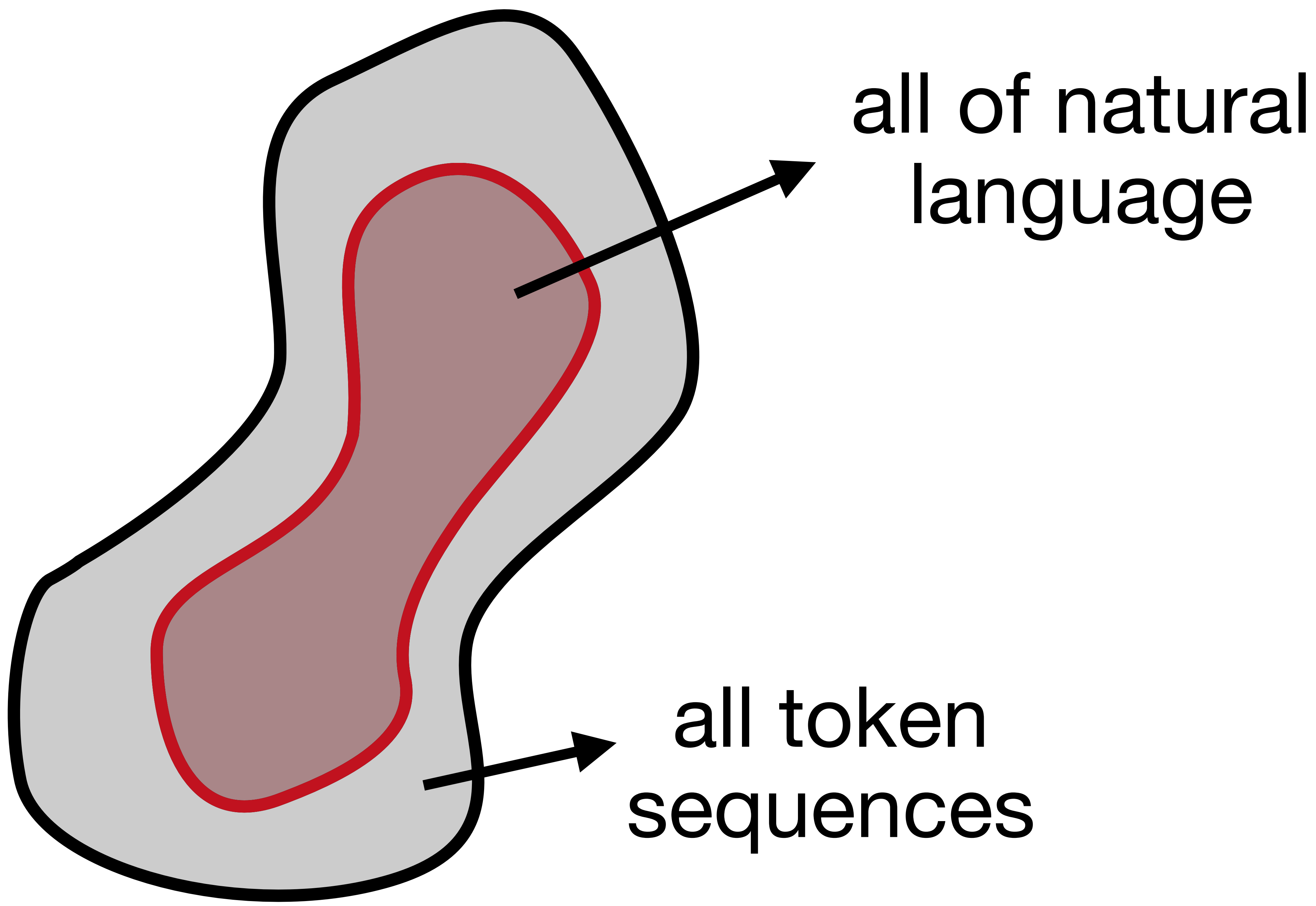
Local Learning in a Picture



Local Learning in a Picture

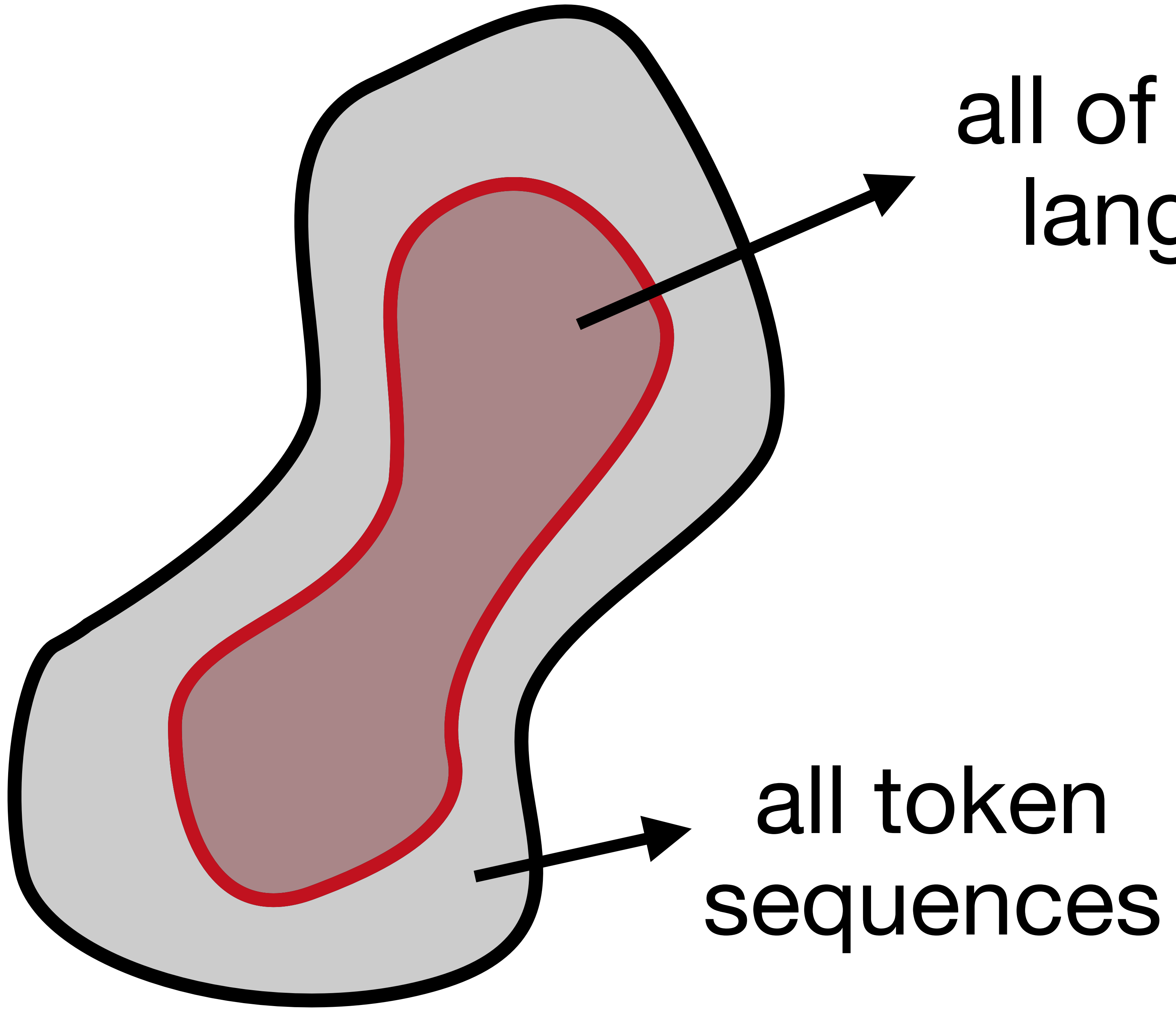


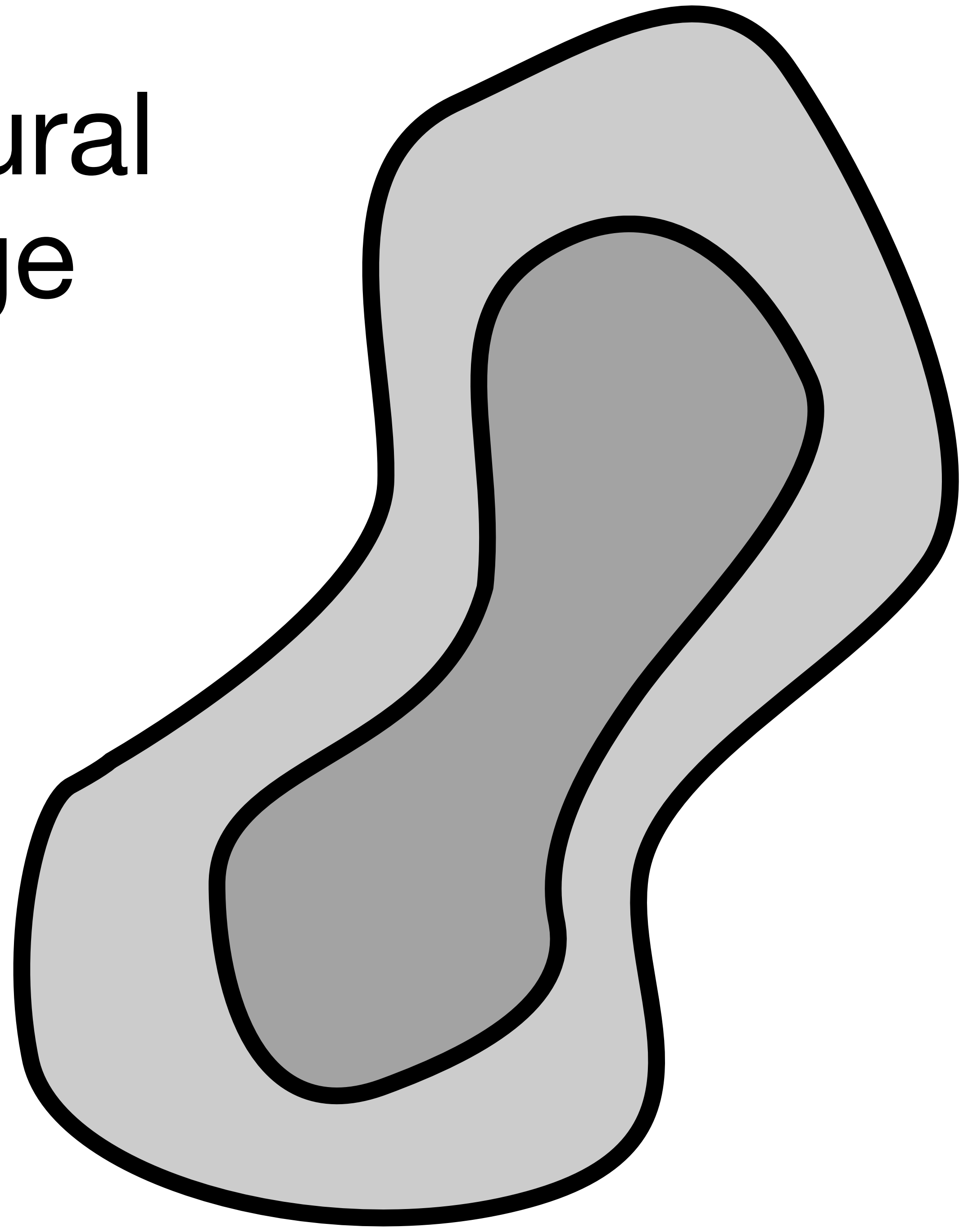
Local Learning in a Picture



inductive learning

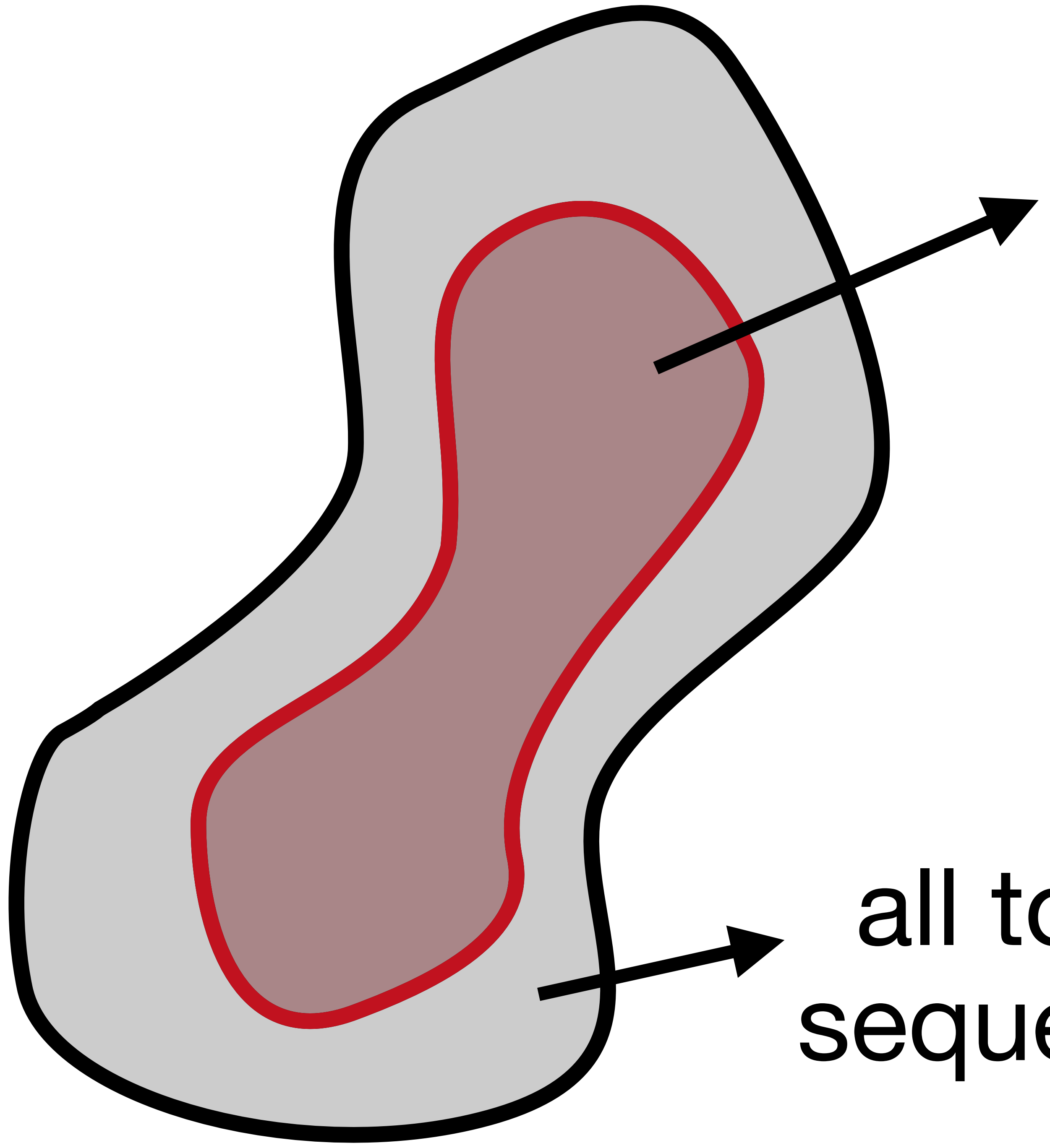
Local Learning in a Picture

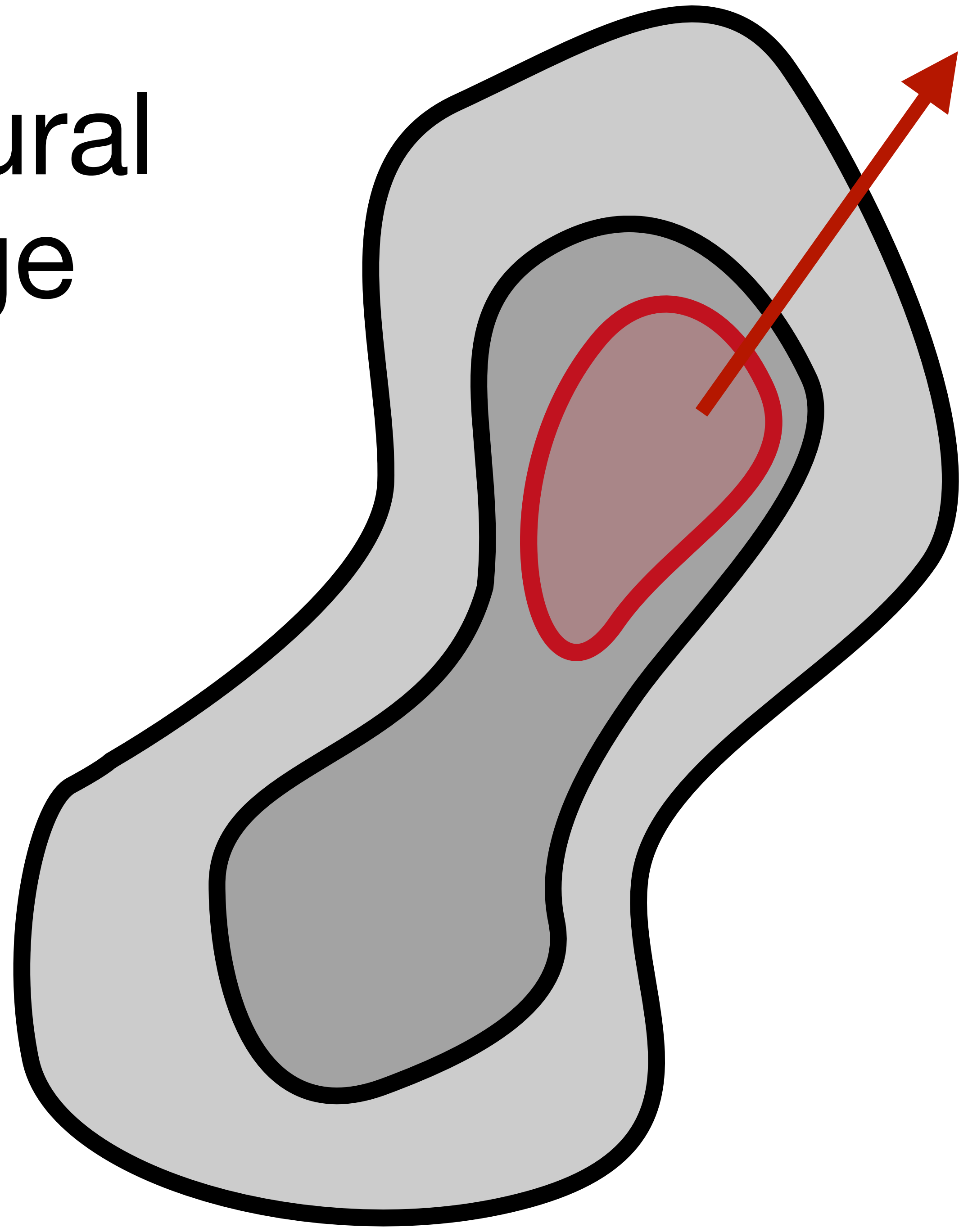




inductive learning

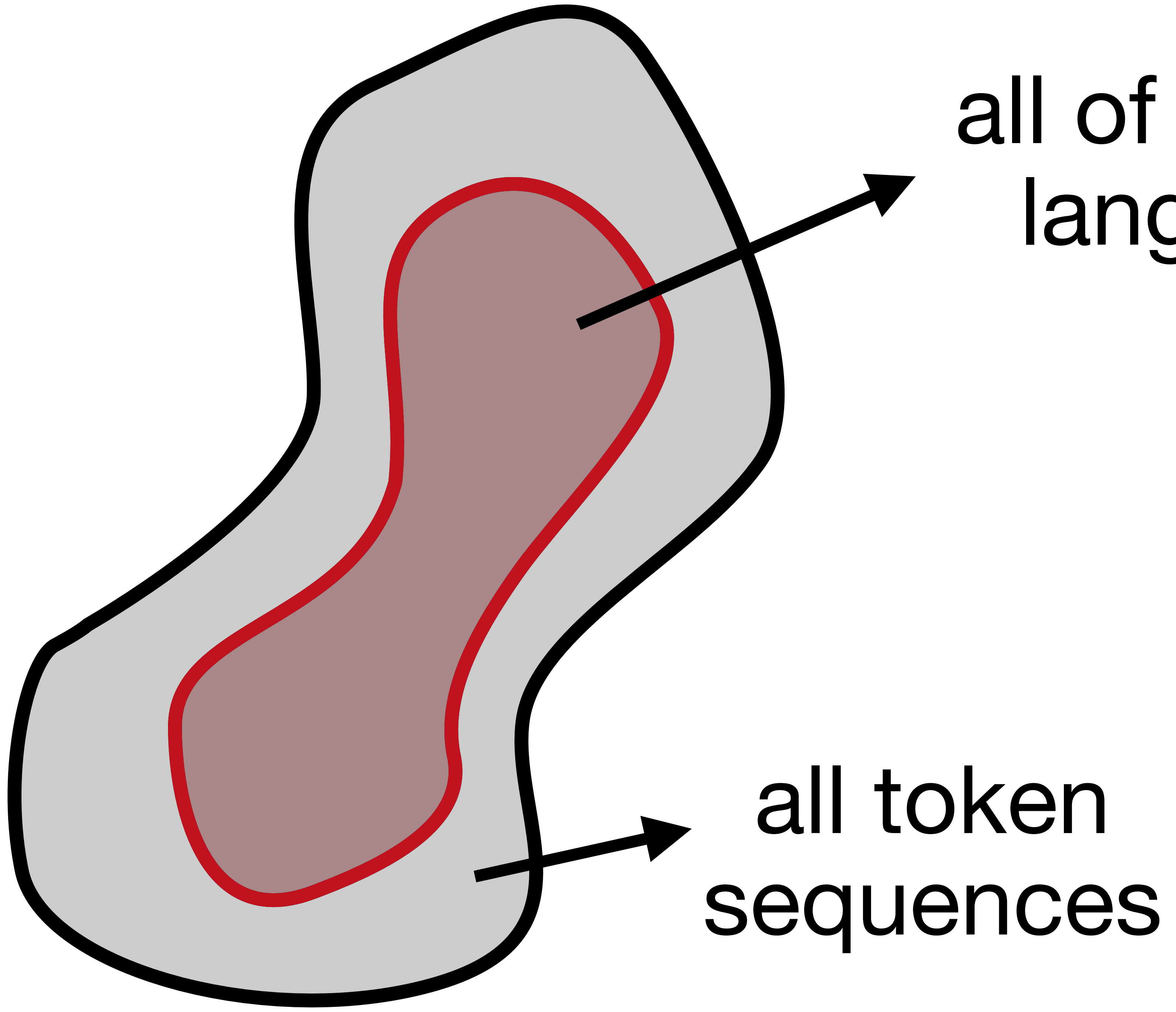
Local Learning in a Picture



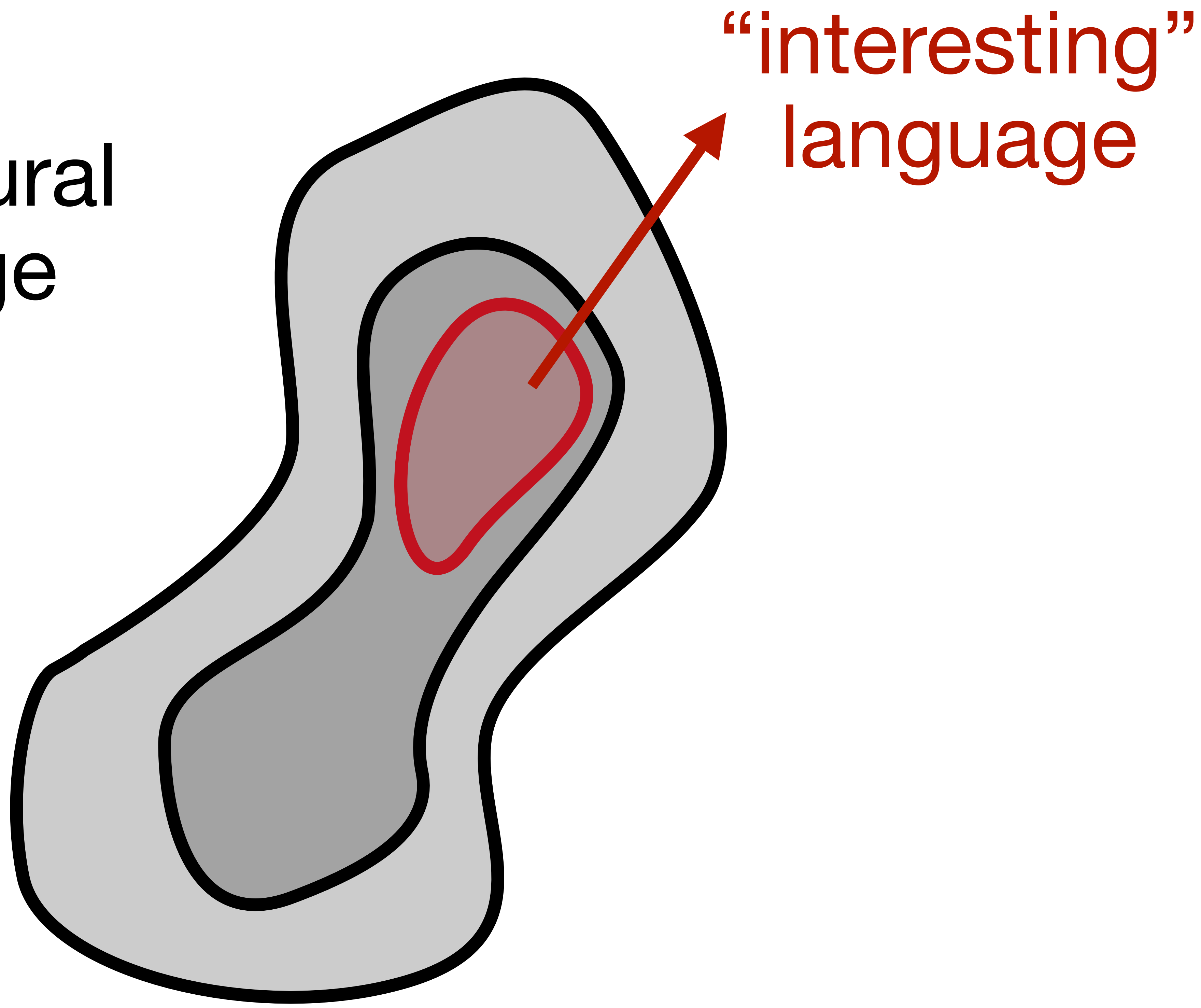


inductive learning

Local Learning in a Picture

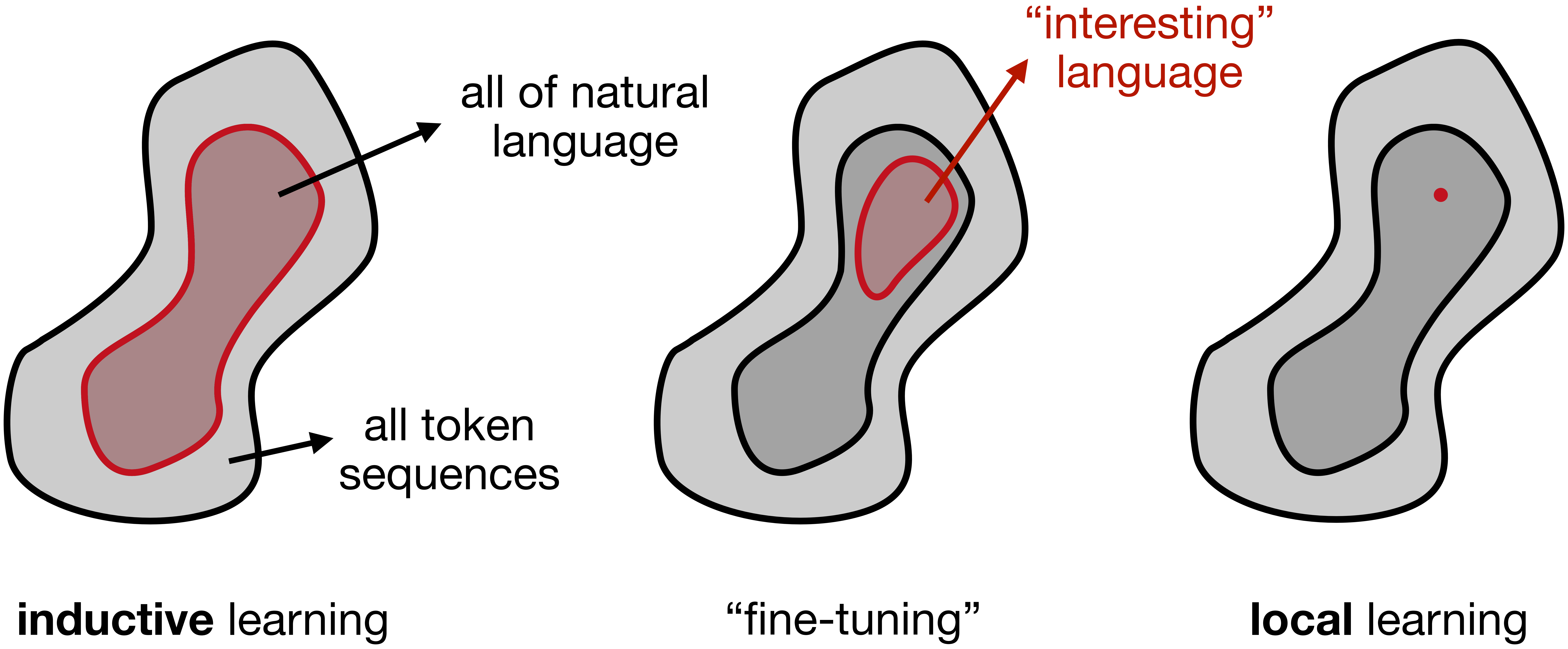


inductive learning

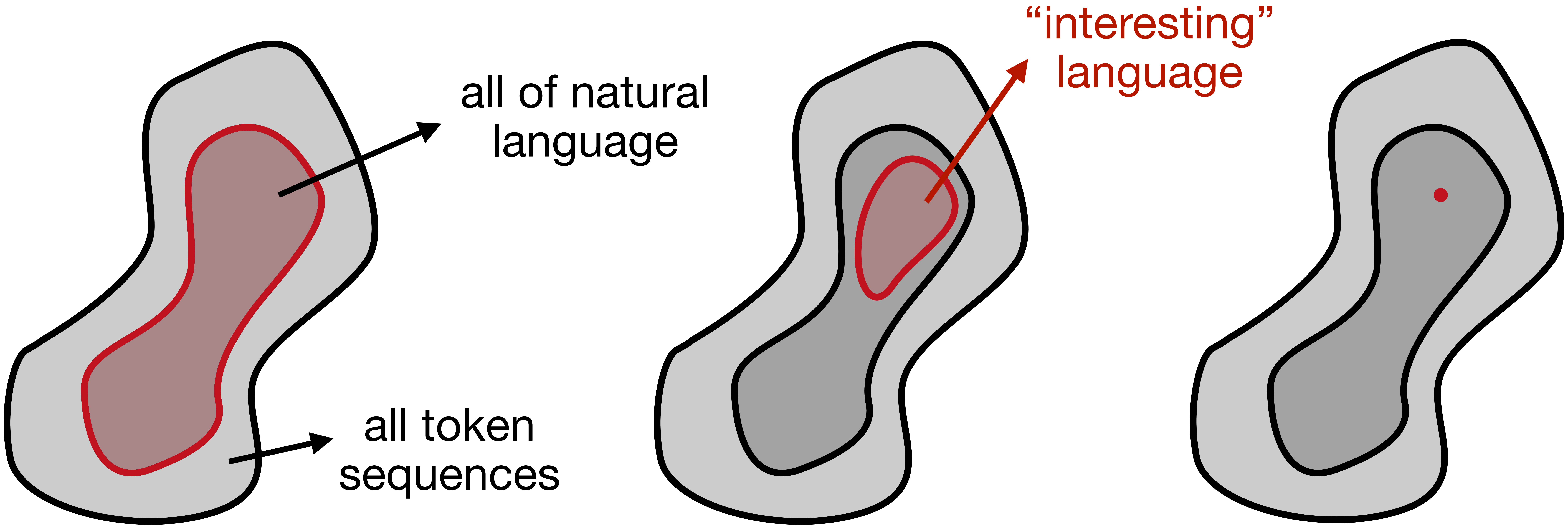


"fine-tuning"

Local Learning in a Picture



Local Learning in a Picture

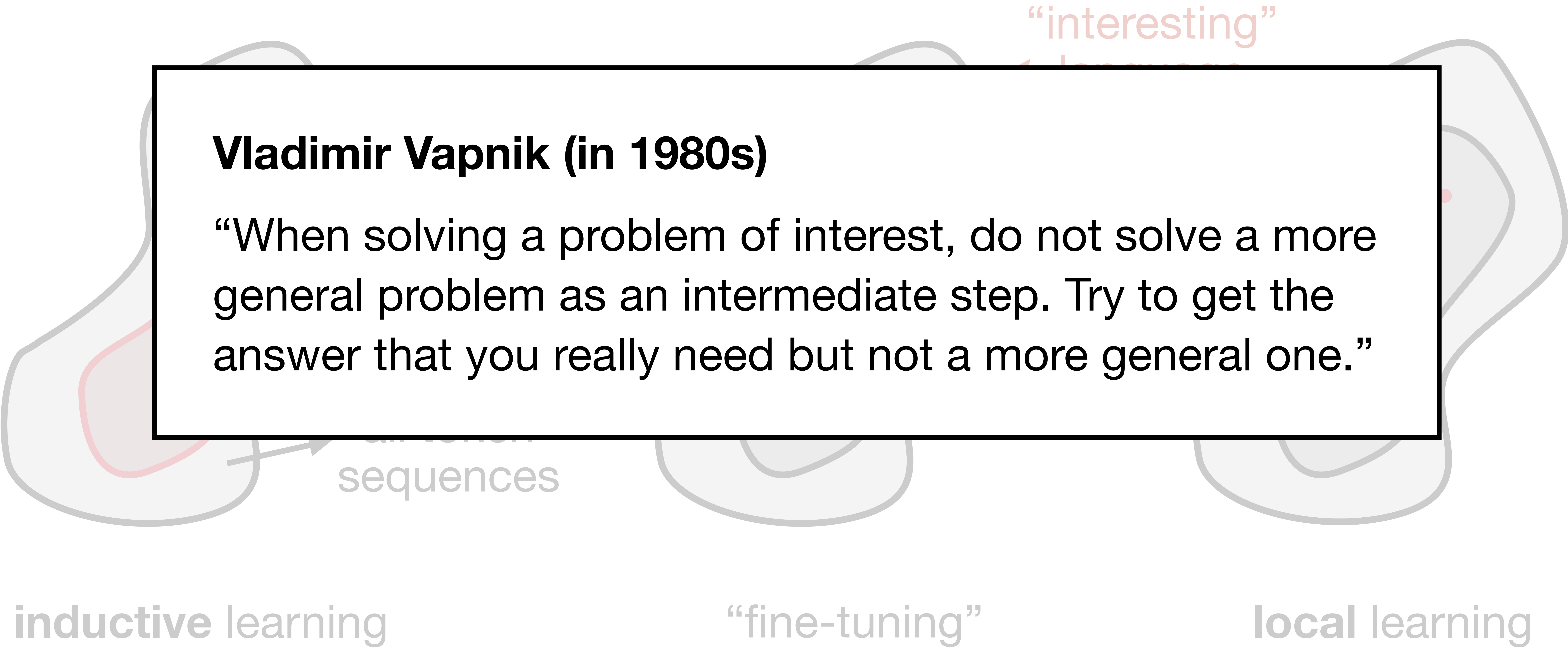


inductive learning

“fine-tuning”

local learning

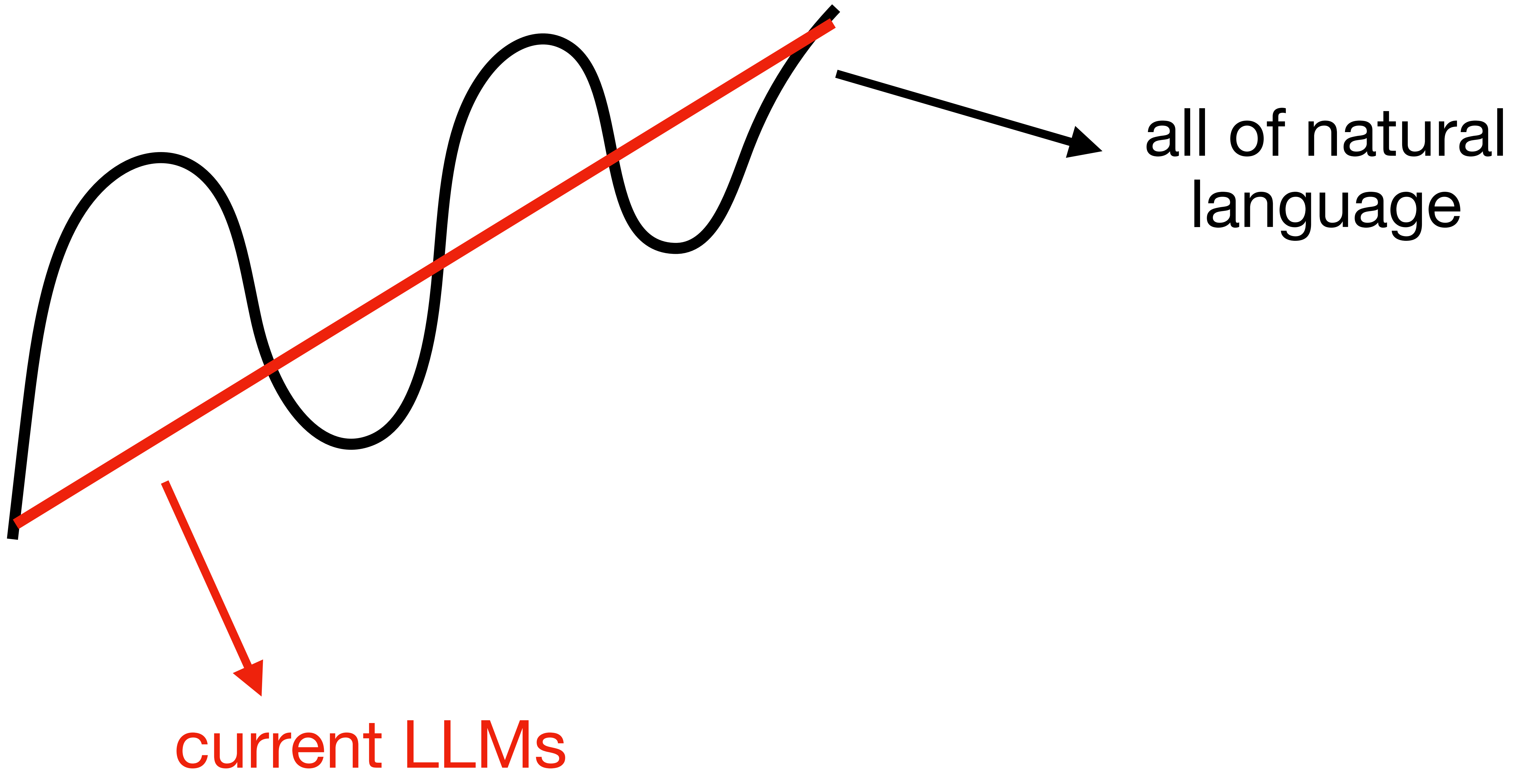
Local Learning in a Picture



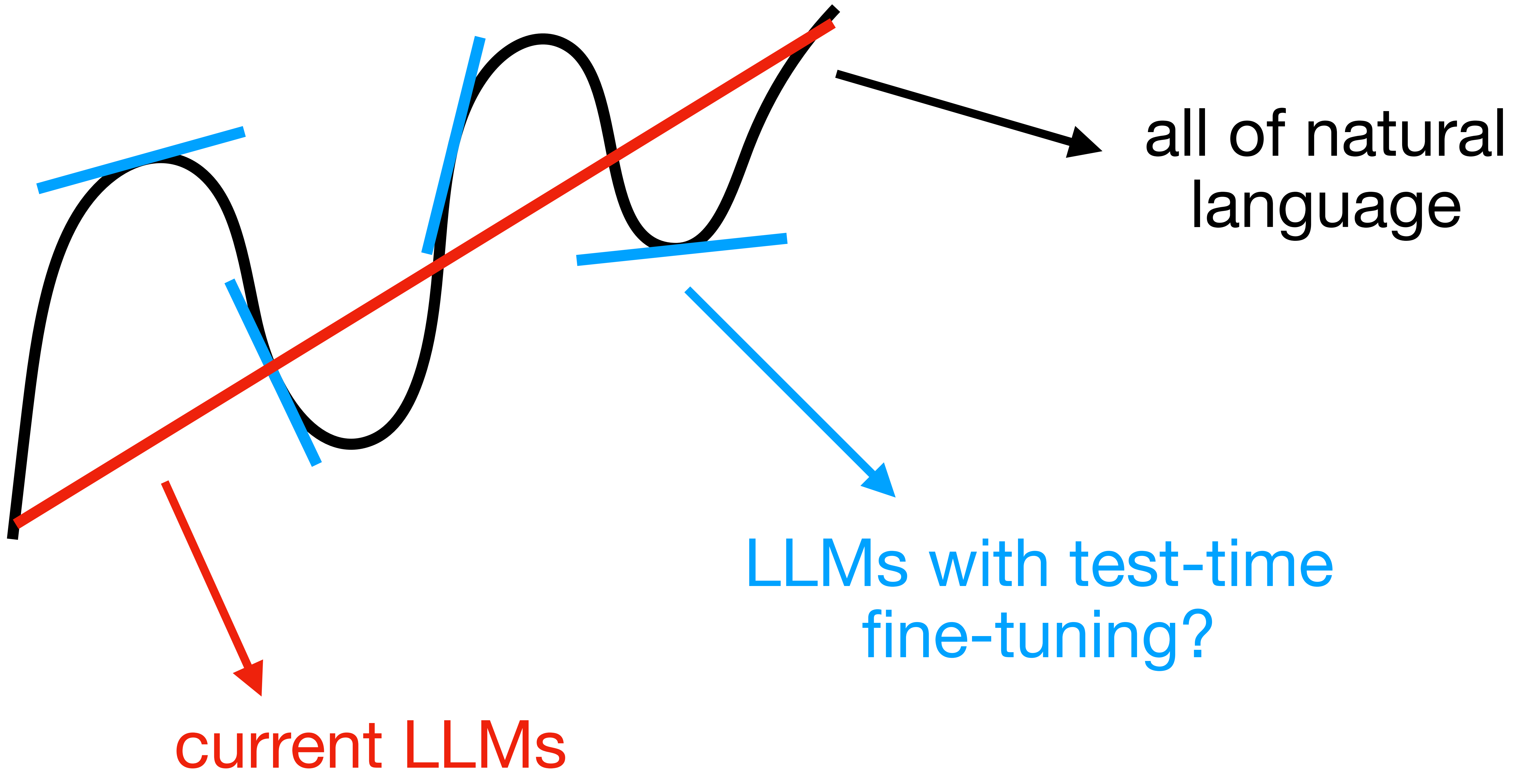
Vladimir Vapnik (in 1980s)

“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.”

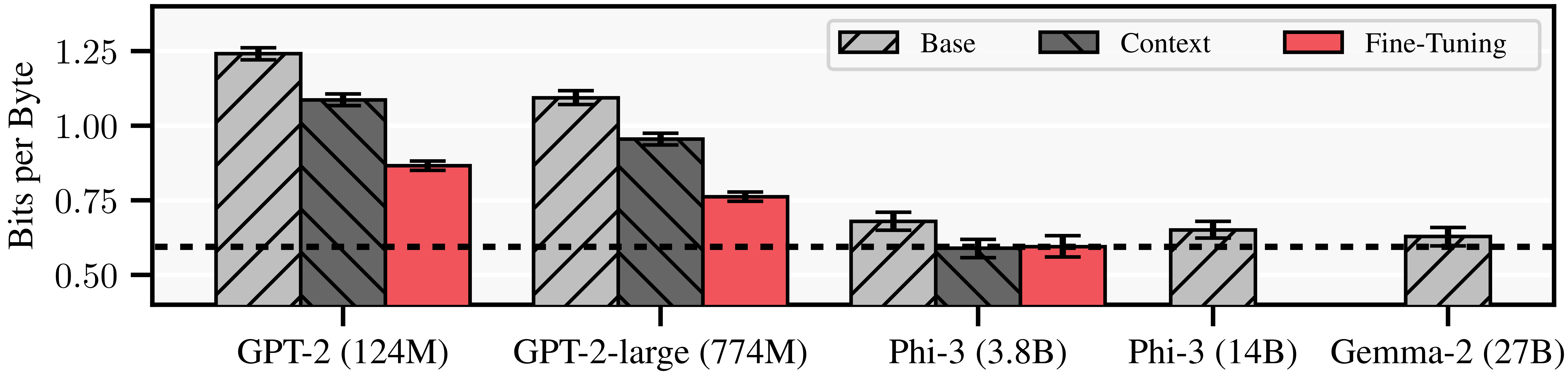
Hypothesis for LLMs



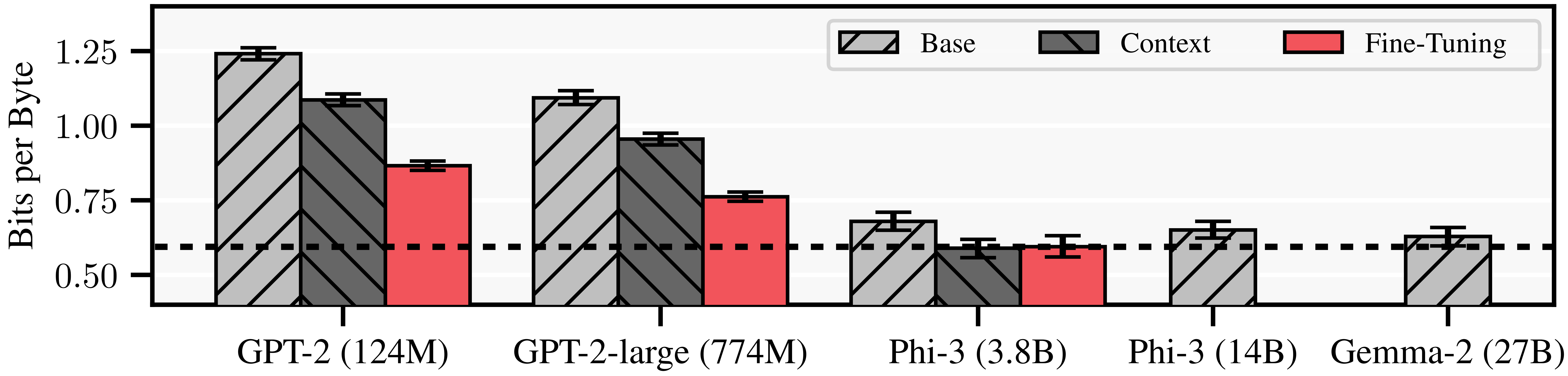
Hypothesis for LLMs



Does Local Learning work with 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

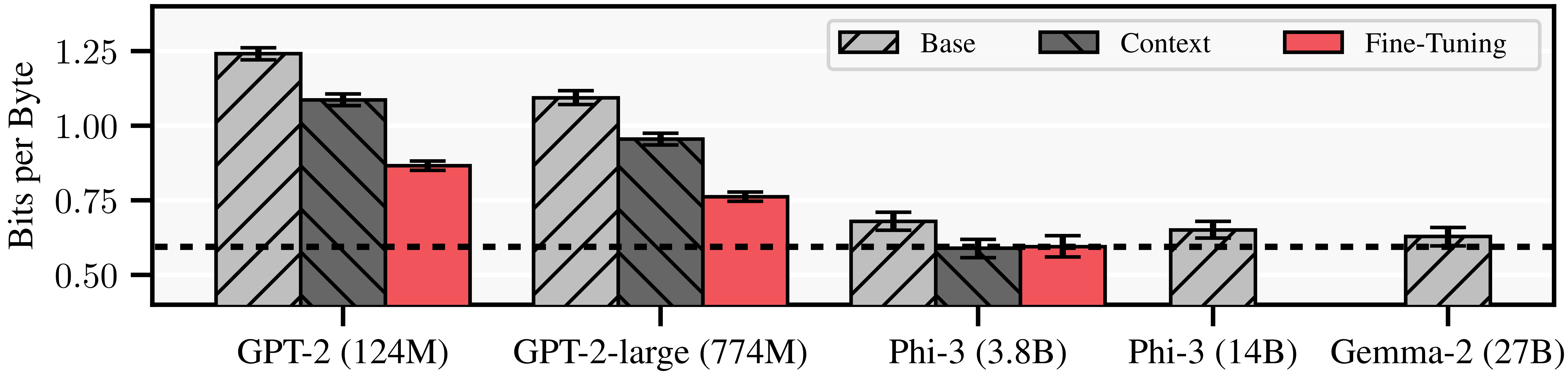
	Context	Fine-Tuning	Δ
GitHub	74.6 (2.5)	**31.0** (2.2)	↓43.6
DeepMind Math	100.2 (0.7)	**74.2** (2.3)	↓26.0
US Patents	87.4 (2.5)	**64.7** (3.8)	↓22.7
FreeLaw	87.2 (3.6)	**68.3** (4.2)	↓18.9

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

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

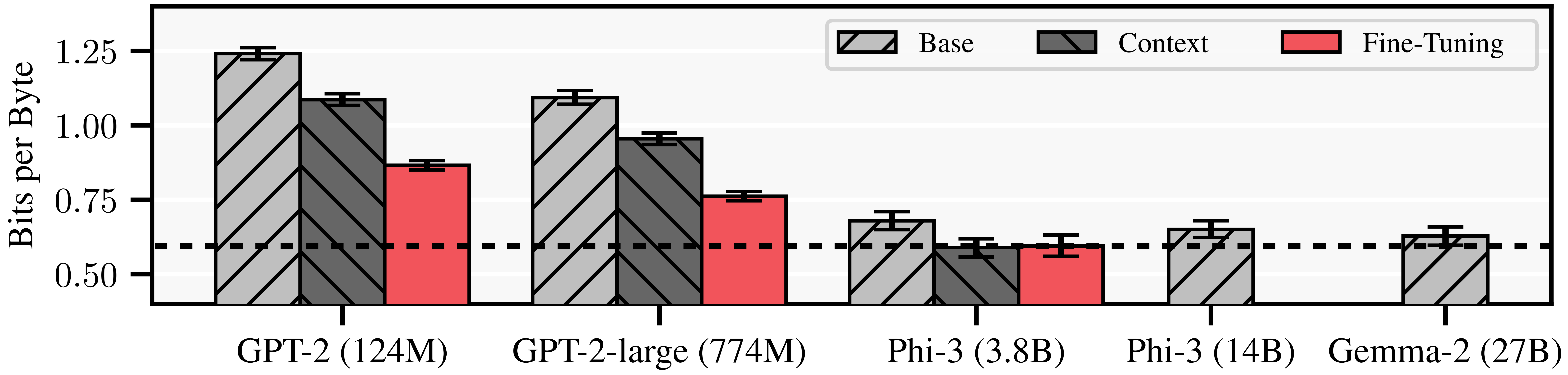
	Context	Fine-Tuning	Δ
GitHub	74.6 (2.5)	**31.0** (2.2)	↓43.6
DeepMind Math	100.2 (0.7)	**74.2** (2.3)	↓26.0
US Patents	87.4 (2.5)	**64.7** (3.8)	↓22.7
FreeLaw	87.2 (3.6)	**68.3** (4.2)	↓18.9

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

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?

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.

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?

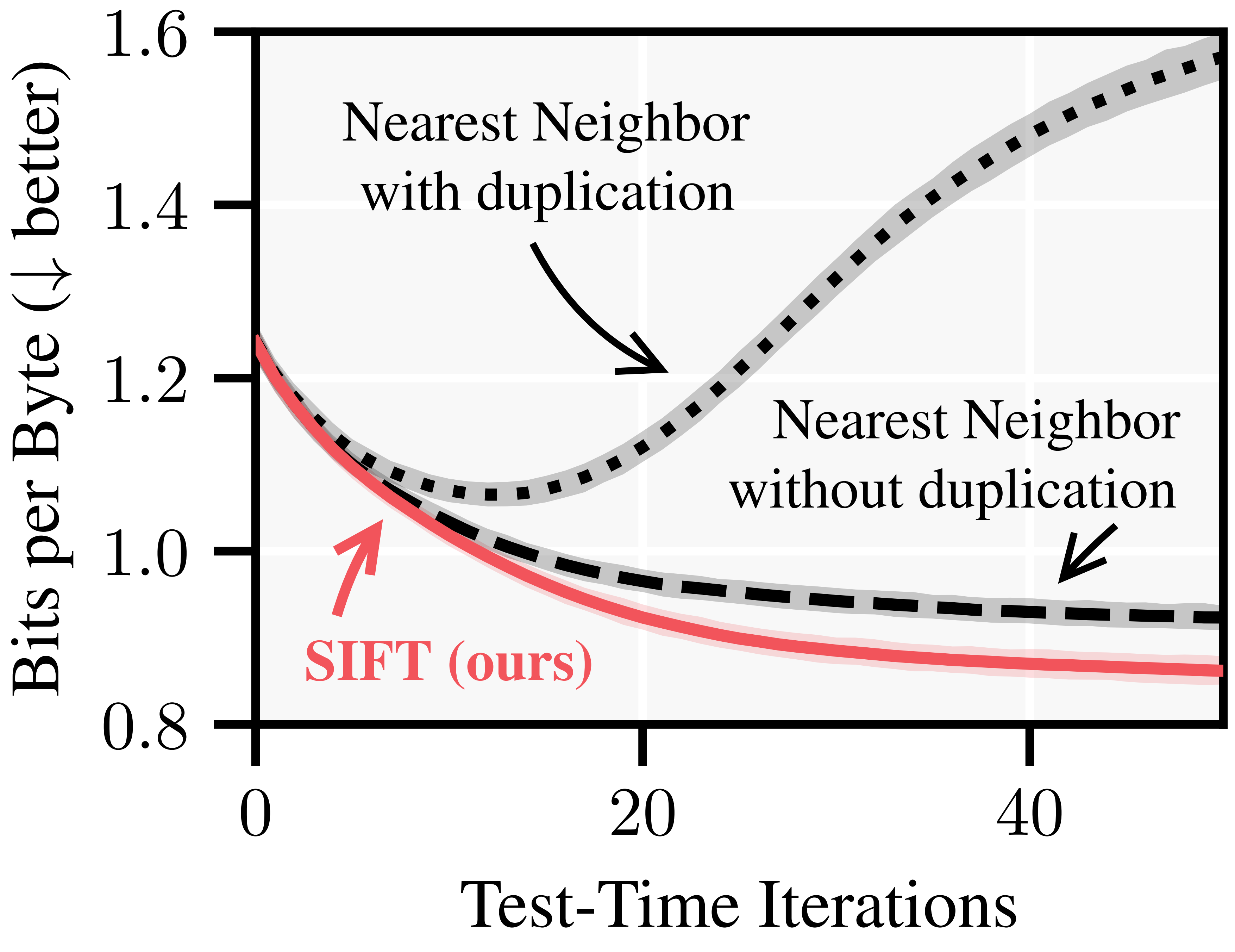
Prompt: What is the age of Michael Jordan and how many kids does he have?

Nearest Neighbor:

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

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.

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

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

❶ Estimating Uncertainty

- Making this tractable:

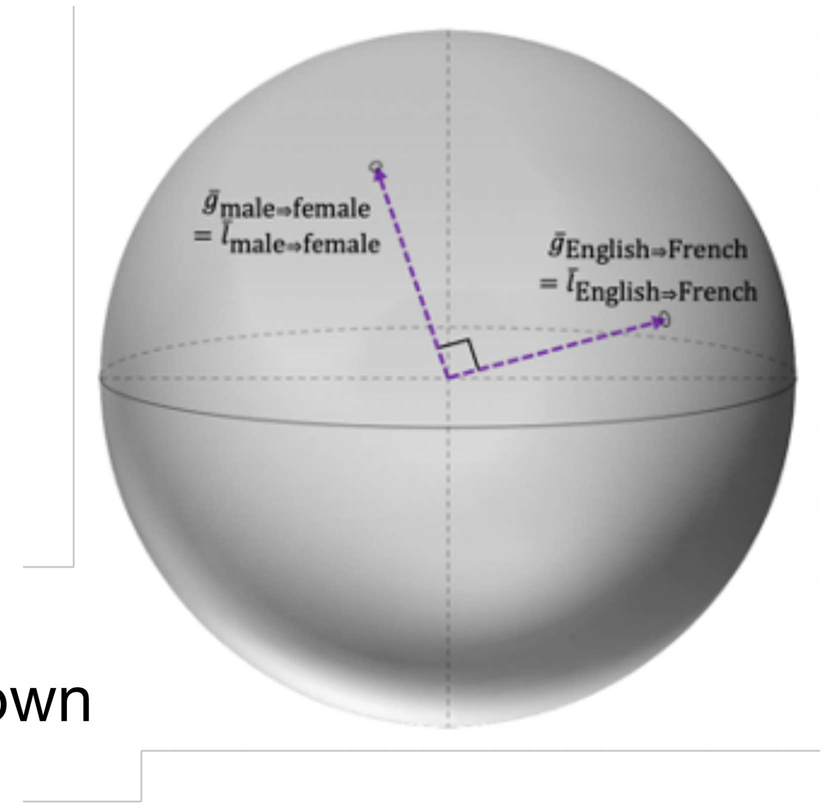
  Surrogate model: logit-linear model $s(f^\star(x))$ with $f^\star(x) = \boldsymbol{W}^\star \boldsymbol{\phi}(x)$

➊ Estimating Uncertainty

- Making this tractable:

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

➊ Estimating Uncertainty

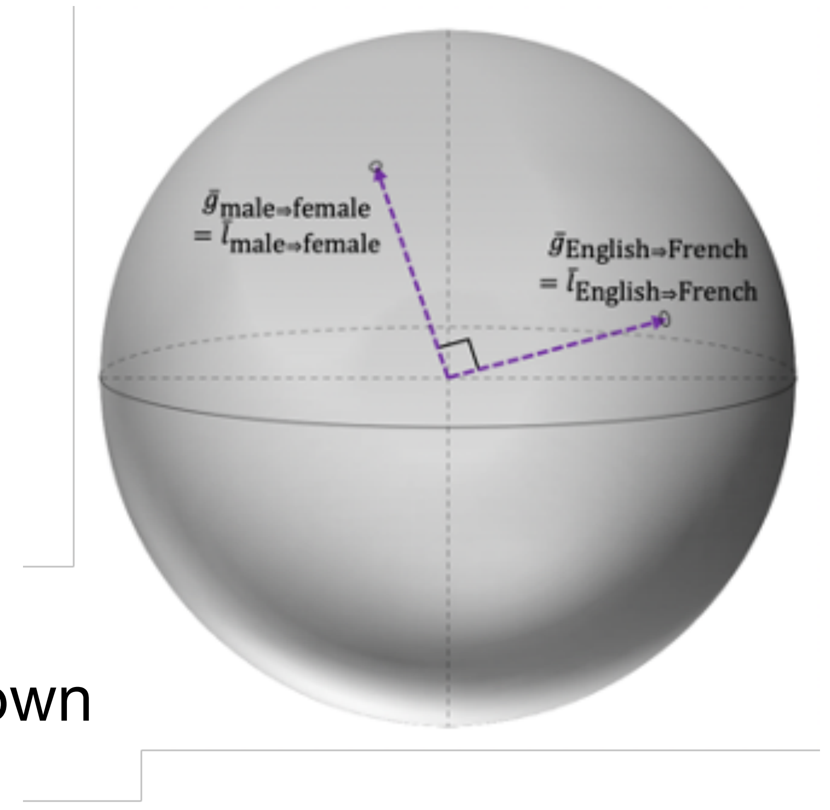


- Making this tractable:

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

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

❶ Estimating Uncertainty



- Making this tractable:

unknown known

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

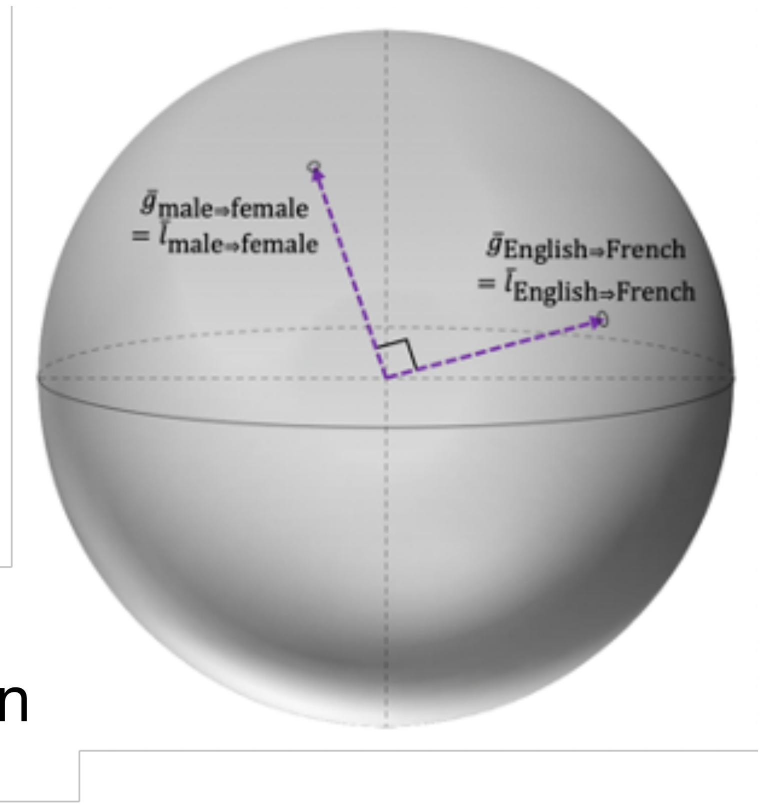
→ linear representation hypothesis [Park, Choe, Veitch; ICML '24]

$s^\star(x) = s(f^\star(x))$ — "truth"

$s^{\text{pre}}(x) = s(\boldsymbol{W}^{\text{pre}} \boldsymbol{\phi}(x))$ — pre-trained model

$s_n(x) = s(\boldsymbol{W}_n \boldsymbol{\phi}(x))$ — fine-tuned model on n pieces of data

❶ Estimating Uncertainty



- Making this tractable:

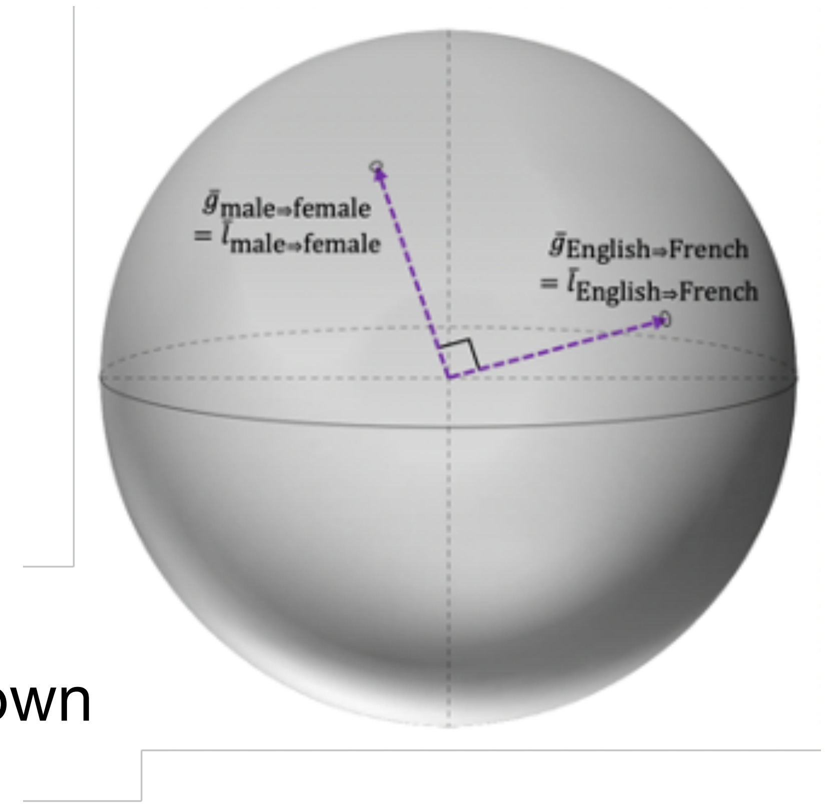
  Surrogate model: logit-linear model $s(f^\star(x))$ with $f^\star(x) = \overset{\text{unknown}}{\boldsymbol{W}^\star}\overset{\text{known}}{\boldsymbol{\phi}}(x)$

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

$s^\star(x) = s(f^\star(x))$	$s^{\text{pre}}(x) = s(\boldsymbol{W}^{\text{pre}}\boldsymbol{\phi}(x))$	$s_n(x) = s(\boldsymbol{W}_n\boldsymbol{\phi}(x))$
“truth”	pre-trained model	fine-tuned model on n pieces of data

- *Confidence sets:* $\mathbb{P}(\forall n \geq 1, x \in \mathcal{X} : d_{\text{TV}}(s_n(x), s^\star(x)) \leq \beta_n(\delta)\ \sigma_n(x)) \geq 1 - \delta$

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]

$s^\star(x) = s(f^\star(x))$ — "truth"

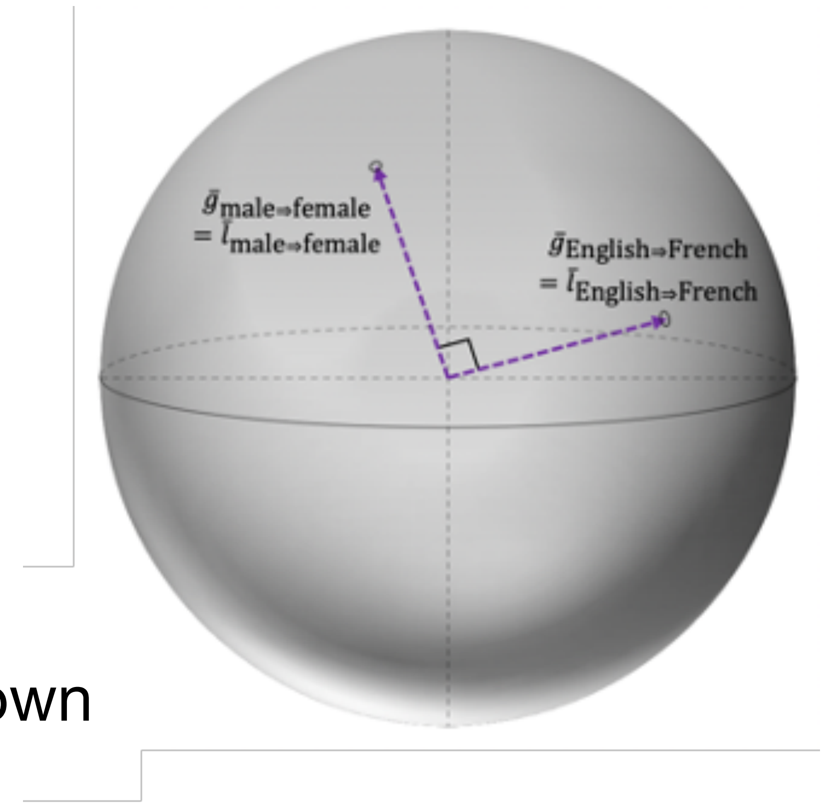
$s^{\text{pre}}(x) = s(\boldsymbol{W}^{\text{pre}} \boldsymbol{\phi}(x))$ — pre-trained model

$s_n(x) = s(\boldsymbol{W}_n \boldsymbol{\phi}(x))$ — fine-tuned model on n pieces of data

- *Confidence sets:* $\mathbb{P}(\forall n \geq 1, x \in \mathcal{X} : d_{\text{TV}}(s_n(x), s^\star(x)) \leq \beta_n(\delta)\ \sigma_n(x)) \geq \boxed{1 - \delta}$

  significance

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]

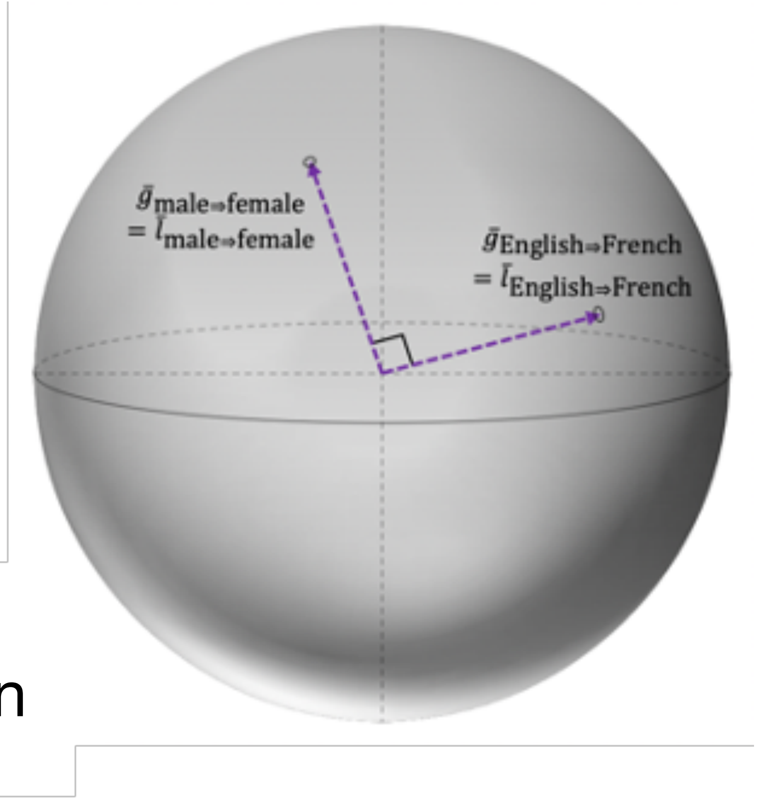
$s^\star(x) = s(f^\star(x))$ "truth"

$s^{\text{pre}}(x) = s(\boldsymbol{W}^{\text{pre}} \boldsymbol{\phi}(x))$ pre-trained model

$s_n(x) = s(\boldsymbol{W}_n \boldsymbol{\phi}(x))$ fine-tuned model on n pieces of data

- *Confidence sets:* $\mathbb{P}(\forall n \geq 1, x \in \mathcal{X} : \underbrace{d_{\text{TV}}(s_n(x), s^\star(x))}_{\text{error}} \leq \underbrace{\beta_n(\delta)}_{\text{scaling}} \underbrace{\sigma_n(x)}_{\text{key object}}) \geq \underbrace{1 - \delta}_{\text{significance}}$

❶ Estimating Uncertainty



- Making this tractable:

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

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

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

- *Confidence sets:* $\mathbb{P}(\forall n \geq 1, x \in \mathcal{X} : \underbrace{d_{\text{TV}}(s_n(x), s^\star(x))}_{\text{error}} \leq \underbrace{\beta_n(\delta)}_{\text{scaling}} \underbrace{\sigma_n(x)}_{\text{key object}}) \geq \underbrace{1 - \delta}_{\text{significance}}$

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

❷ Minimizing "Posterior" Uncertainty

❷ Minimizing “Posterior” Uncertainty

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

❷ Minimizing "Posterior" Uncertainty

- Choose data that minimizes uncertainty of the model after seeing this data:

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

prompt

❷ Minimizing "Posterior" Uncertainty

- Choose data that minimizes uncertainty of the model after seeing this data:

$$x_{n+1} = \operatorname*{argmin}_x \; \sigma_{X_n \cup \{x\}}(x^\star)$$ prompt

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

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

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

- *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

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

A probabilistic interpretation of SIFT

A probabilistic interpretation of SIFT

probabilistic model
with **belief** about f
("controller")

A probabilistic interpretation of SIFT

probabilistic model
with **belief** about f
(“controller”)

“memory”

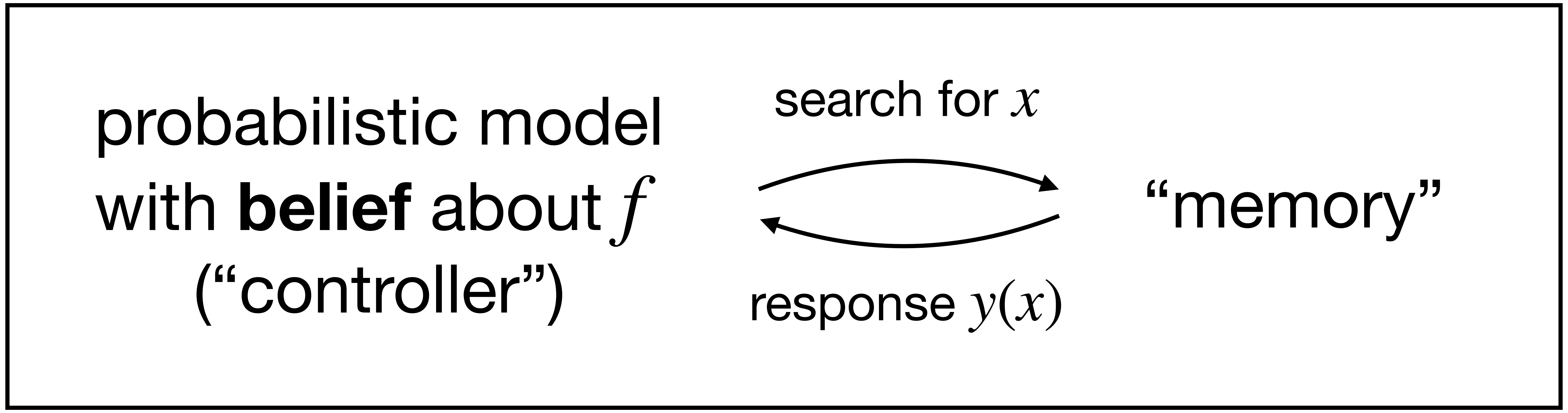
A probabilistic interpretation of SIFT

probabilistic model
with **belief** about f
(“controller”)

$\xrightarrow{\text{search for } x}$

“memory”

A probabilistic interpretation of SIFT



A probabilistic interpretation of SIFT

probabilistic model
with **belief** about f
("controller")

search for x

response $y(x)$

"memory"

$$x_{n+1} = \operatorname*{argmax}_x \ \mathrm{I}(f(x^\star); y(x) \mid y_{1:n})$$

A probabilistic interpretation of SIFT

probabilistic model
with **belief** about f
(“controller”)

search for x

response $y(x)$

“memory”

$$
\begin{aligned}
x_{n+1} &= \operatorname*{argmax}_x \; \mathrm{I}(f(x^\star); y(x) \mid y_{1:n}) \\
&= \operatorname*{argmax}_x \; \mathrm{I}(f(x^\star); y(x)) - \mathrm{I}(f(x^\star); y(x); y_{1:n})
\end{aligned}
$$

A probabilistic interpretation of SIFT

probabilistic model
with **belief** about f
("controller")

search for x

response $y(x)$

"memory"

$$
\begin{aligned}
x_{n+1} &= \operatorname*{argmax}_x \; \mathrm{I}(f(x^\star); y(x) \mid y_{1:n}) \\
&= \operatorname*{argmax}_x \; \underbrace{\mathrm{I}(f(x^\star); y(x))}_{\text{relevance}} - \mathrm{I}(f(x^\star); y(x); y_{1:n})
\end{aligned}
$$

A probabilistic interpretation of SIFT

probabilistic model
with **belief** about f
("controller")

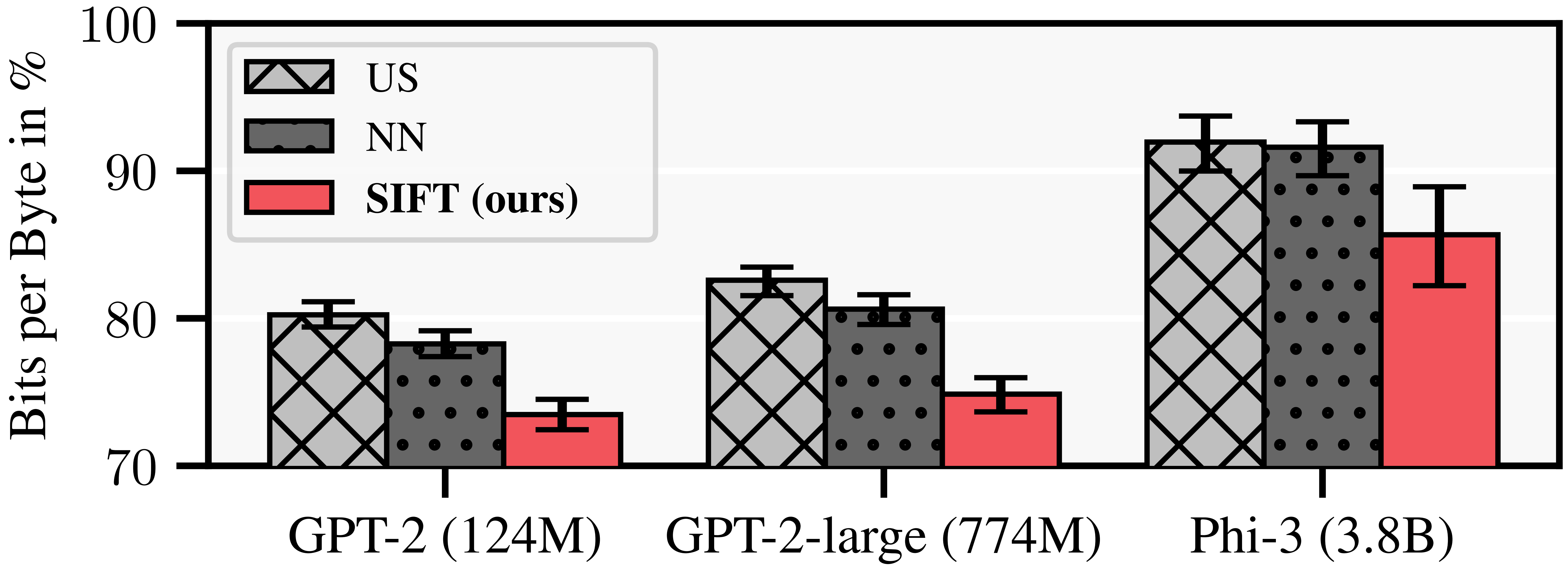
search for x

response $y(x)$

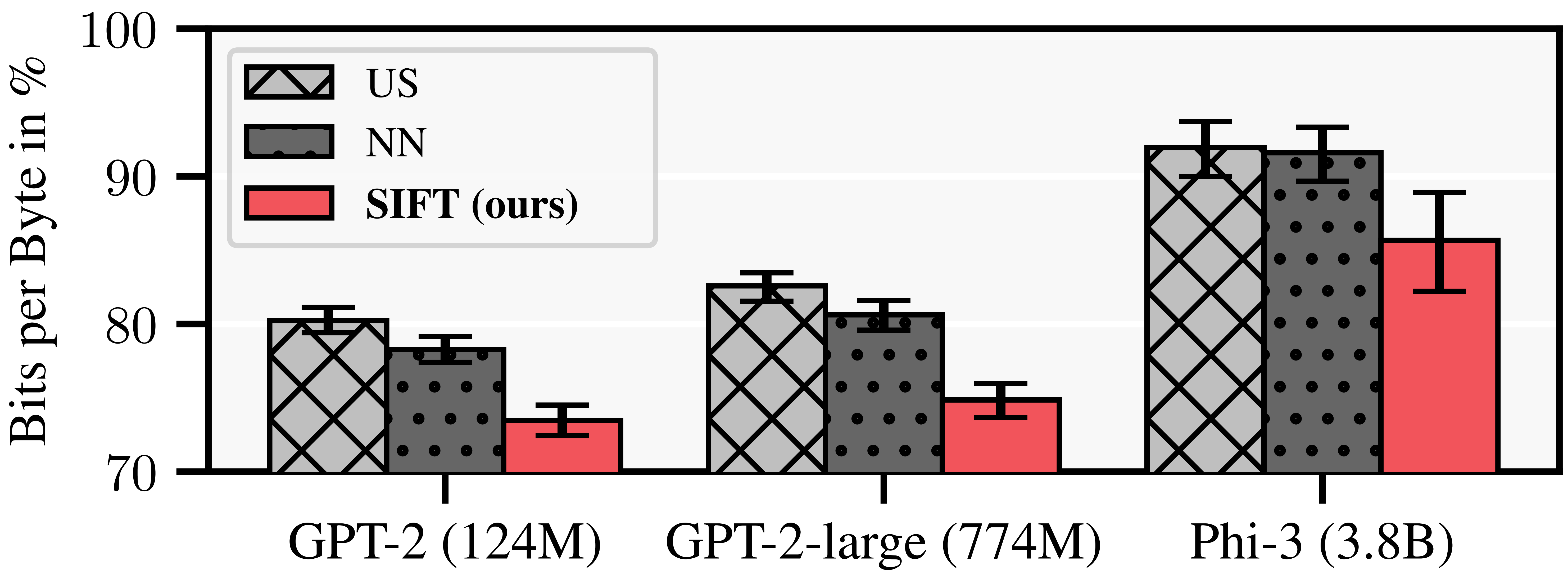
"memory"

$$
\begin{aligned}
x_{n+1} &= \operatorname*{argmax}_x \; \mathrm{I}(f(x^\star); y(x) \mid y_{1:n}) \\
&= \operatorname*{argmax}_x \; \underbrace{\mathrm{I}(f(x^\star); y(x))}_{\text{relevance}} - \underbrace{\mathrm{I}(f(x^\star); y(x); y_{1:n})}_{\text{redundancy}}
\end{aligned}
$$

Does SIFT work?

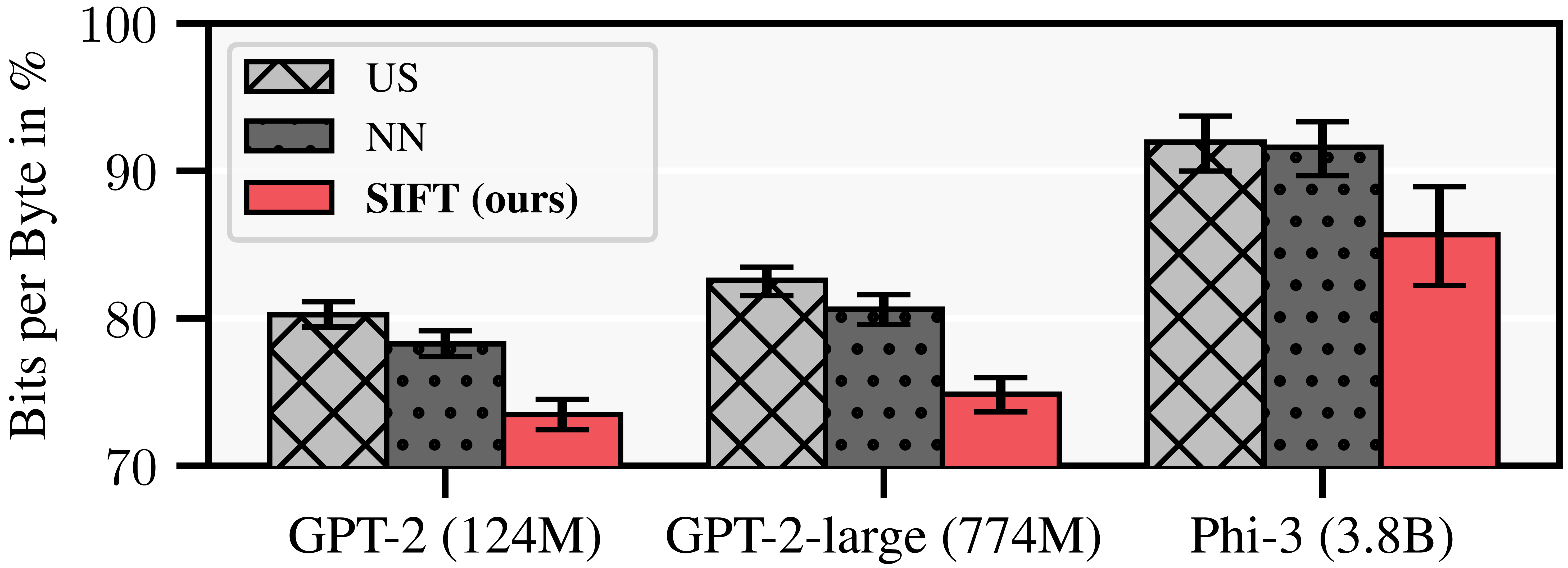


Does SIFT work?



→ larger gains with stronger base models!

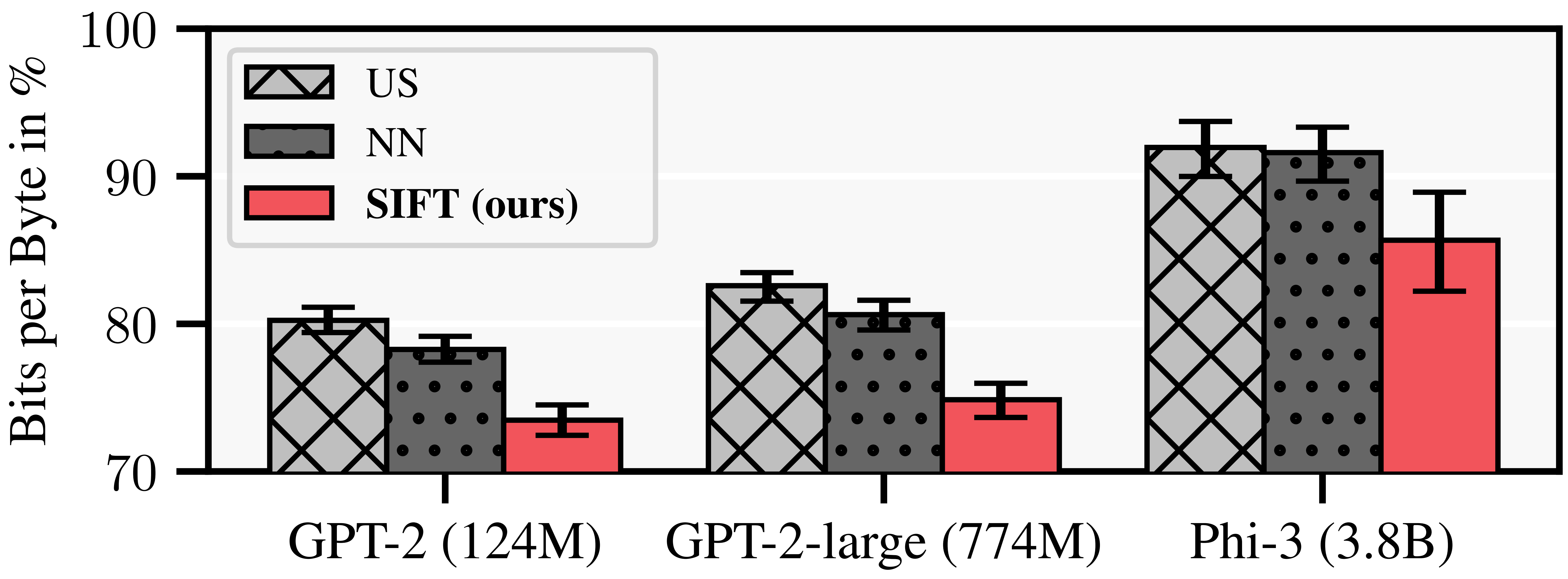
Does SIFT work?



→ larger gains with stronger base models!

→ larger gains with larger “memory”!

Does SIFT work?

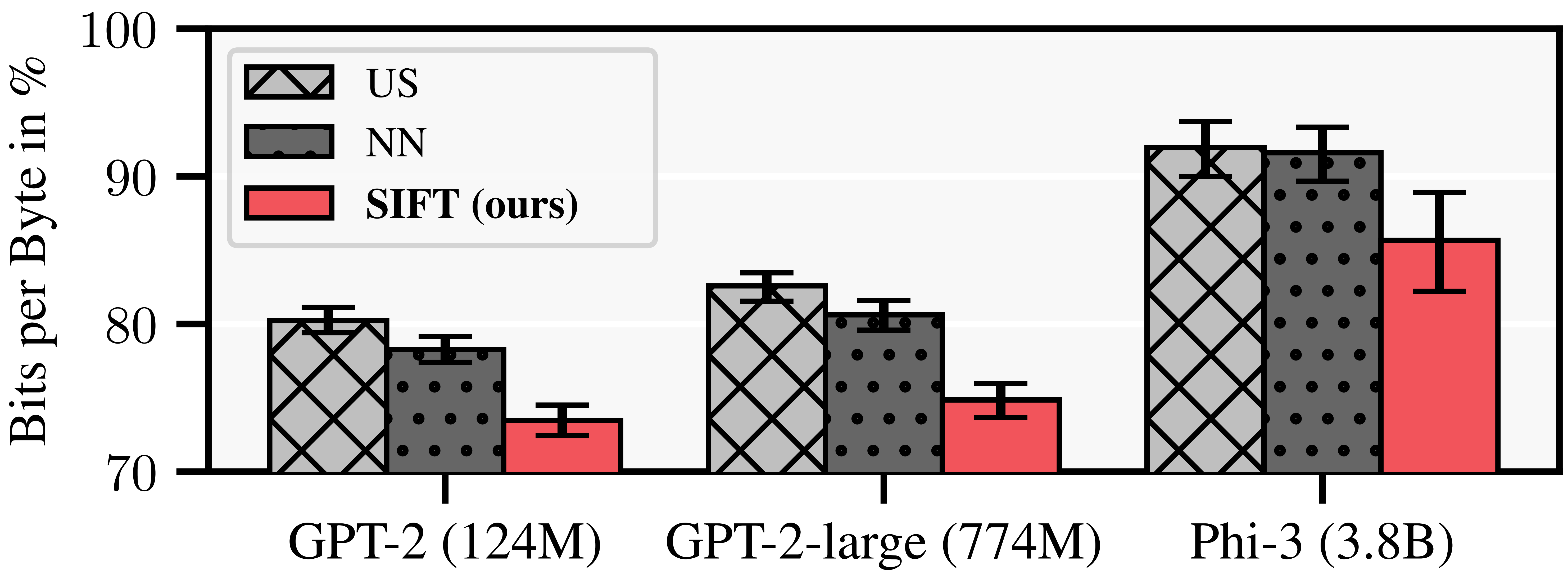


	US	NN	NN-F	SIFT	Δ
NIH Grants	93.1 (1.1)	84.9 (2.1)	91.6 (16.7)	**53.8** (8.9)	↓31.1
US Patents	85.6 (1.5)	80.3 (1.9)	108.8 (6.6)	**62.9** (3.5)	↓17.4
GitHub	45.6 (2.2)	42.1 (2.0)	53.2 (4.0)	**30.0** (2.2)	↓12.1
Enron Emails	**68.6** (9.8)	**64.4** (10.1)	91.6 (20.6)	**53.1** (11.4)	↓11.3
Wikipedia	67.5 (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!

→ larger gains with larger “memory”!

Does SIFT work?



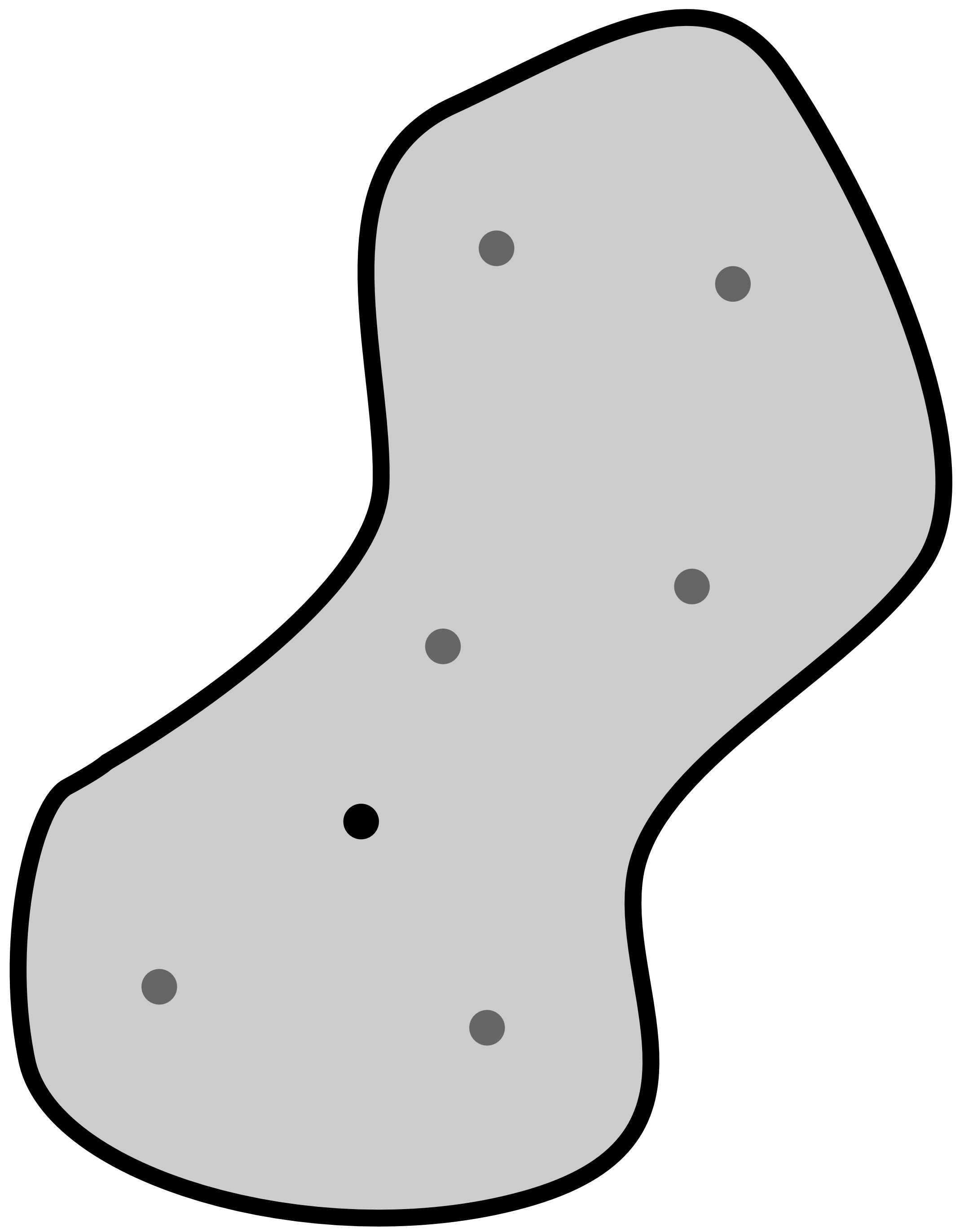
	US	NN	NN-F	SIFT	Δ
NIH Grants	93.1 (1.1)	84.9 (2.1)	91.6 (16.7)	**53.8** (8.9)	↓31.1
US Patents	85.6 (1.5)	80.3 (1.9)	108.8 (6.6)	**62.9** (3.5)	↓17.4
GitHub	45.6 (2.2)	42.1 (2.0)	53.2 (4.0)	**30.0** (2.2)	↓12.1
Enron Emails	**68.6** (9.8)	**64.4** (10.1)	91.6 (20.6)	**53.1** (11.4)	↓11.3
Wikipedia	67.5 (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!

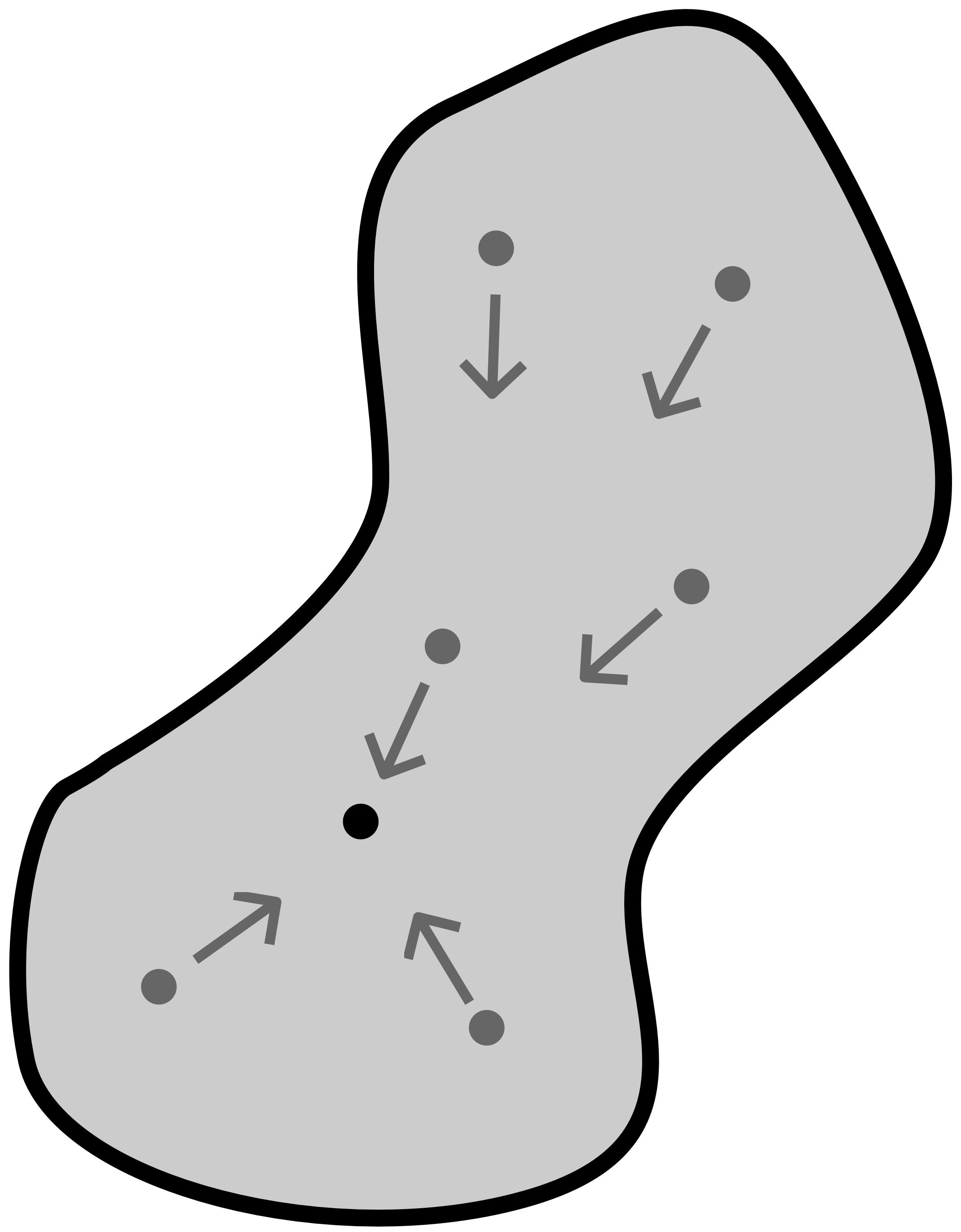
→ larger gains with larger “memory”!

Can we learn representations over time?

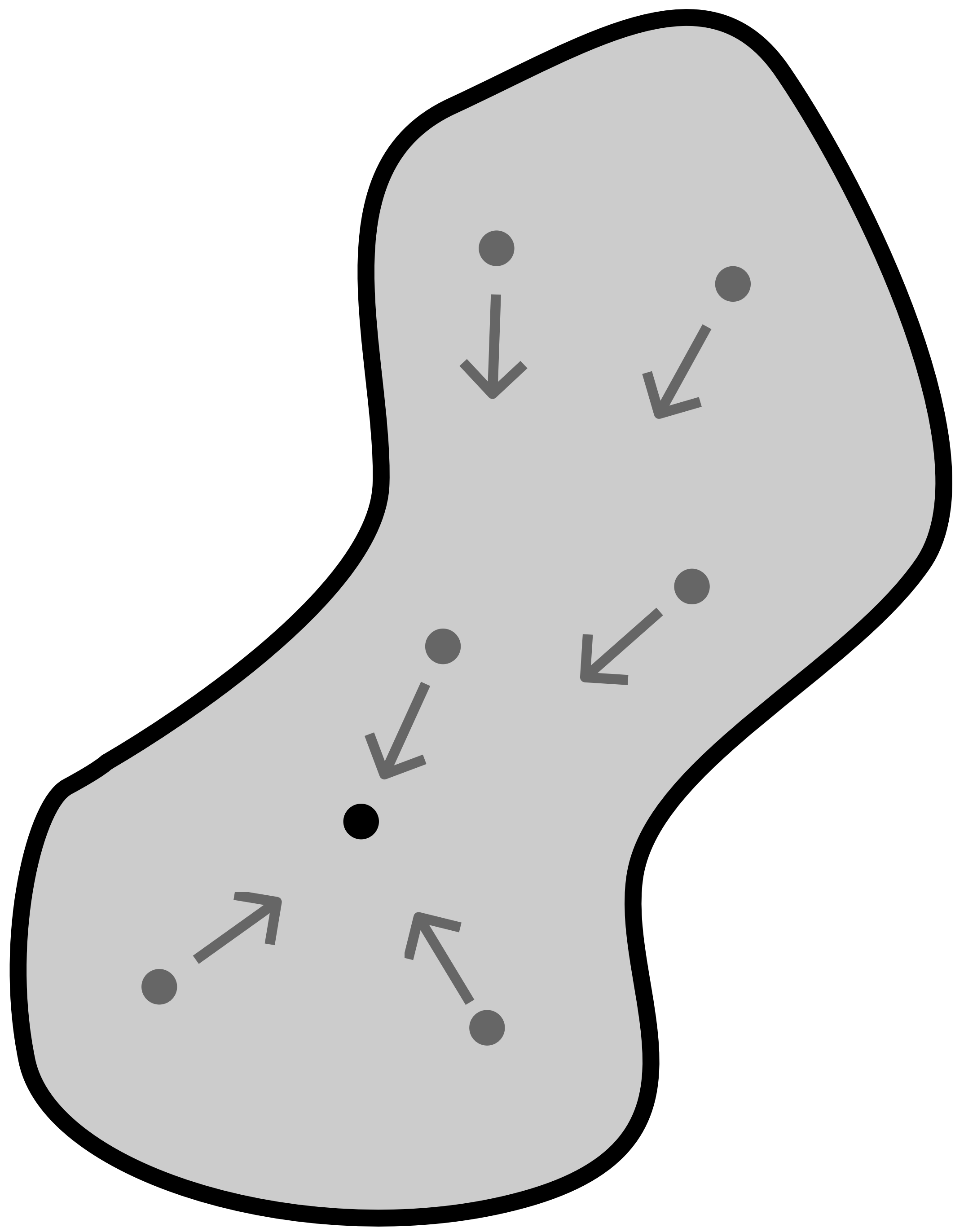
Can we learn representations over time?



Can we learn representations over time?


representations

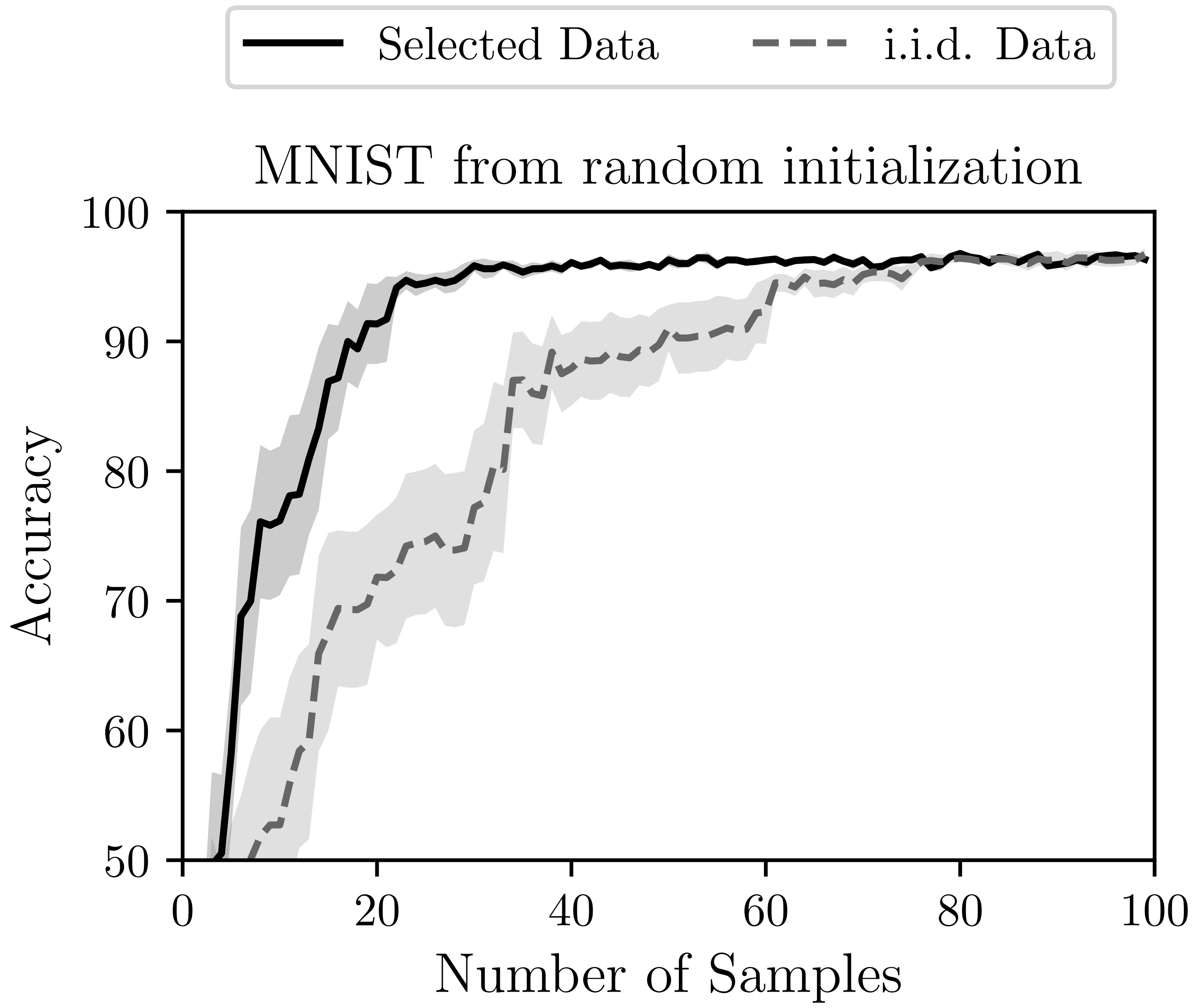
Can we learn representations over time?



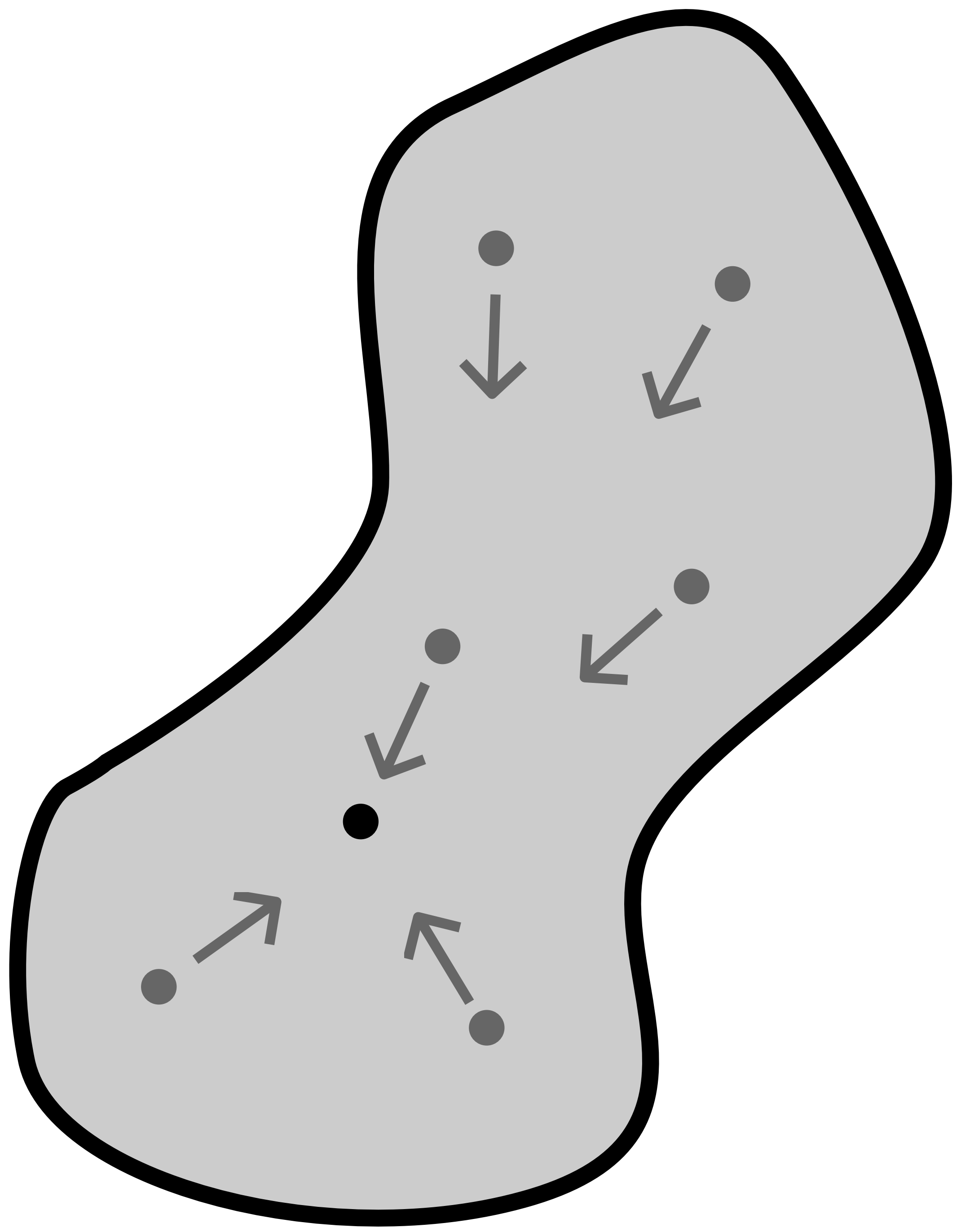
representations

Strong representations can be bootstrapped!

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



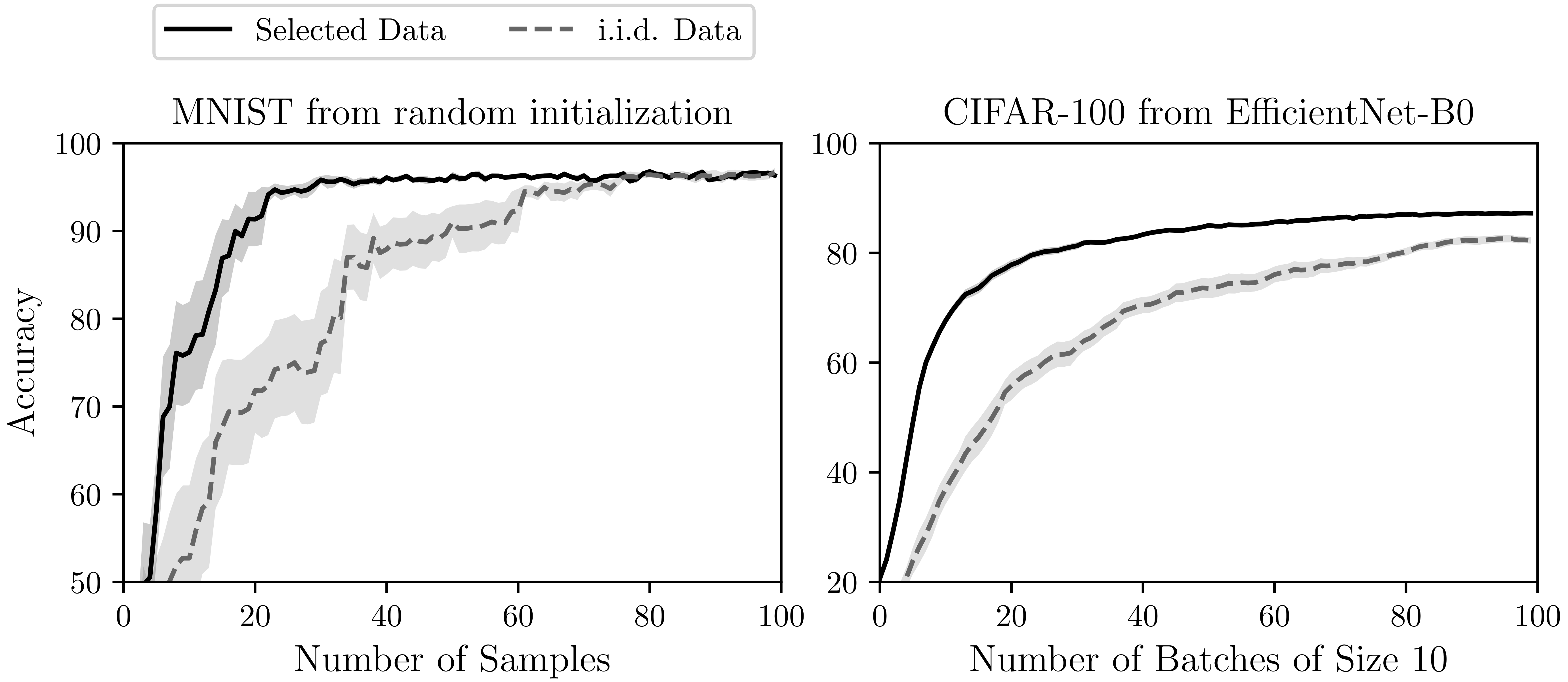
Can we learn representations over time?



representations

Strong representations can be bootstrapped!

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



Summary

Summary

Local models

solve one problem at a time

Summary

Local models

solve one problem at a time

Inductive models (most current SOTA models)

attempt to solve all possible problems at once

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”!

- **Transductive Active Learning: Theory and Applications**

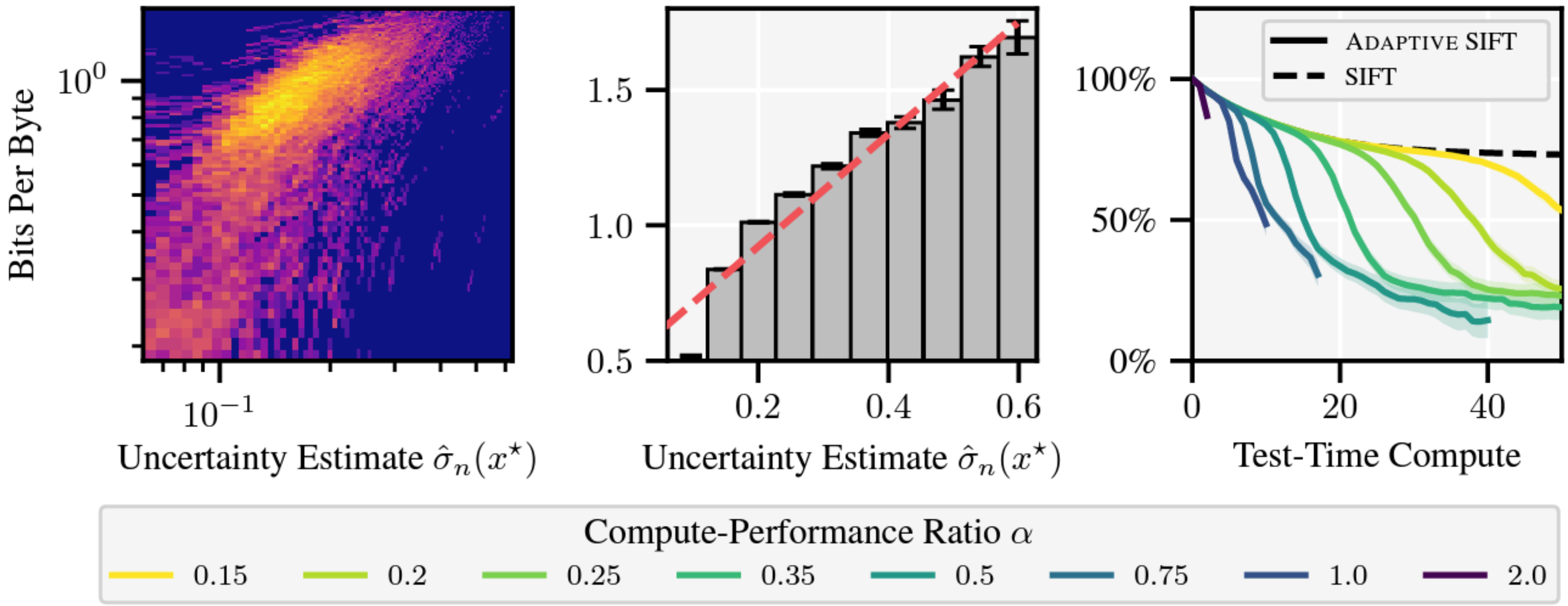
  NeurIPS ’24

- **Efficiently Learning at Test-Time: Active Fine-Tuning of LLMs**

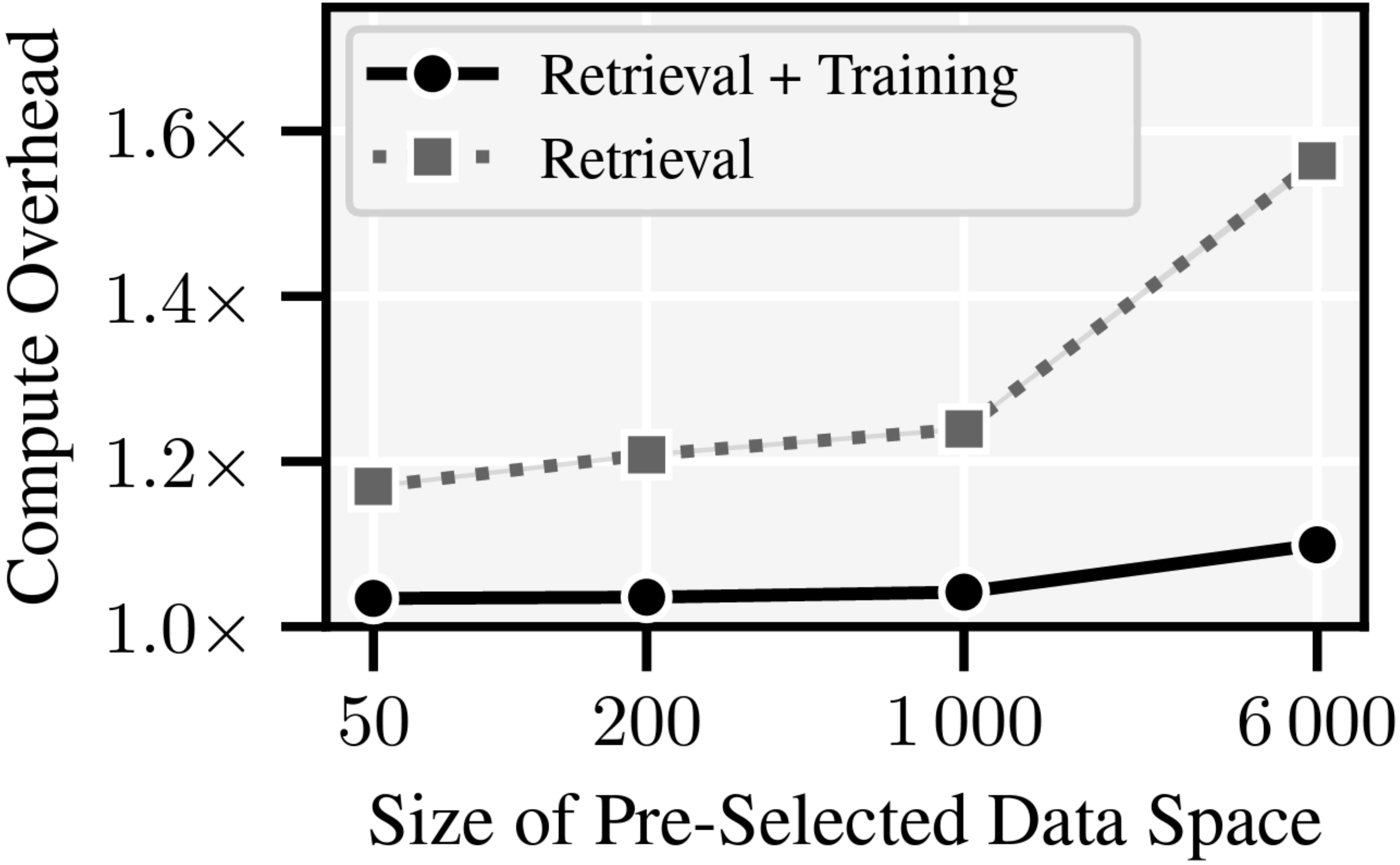
  NeurIPS ’24 Workshops

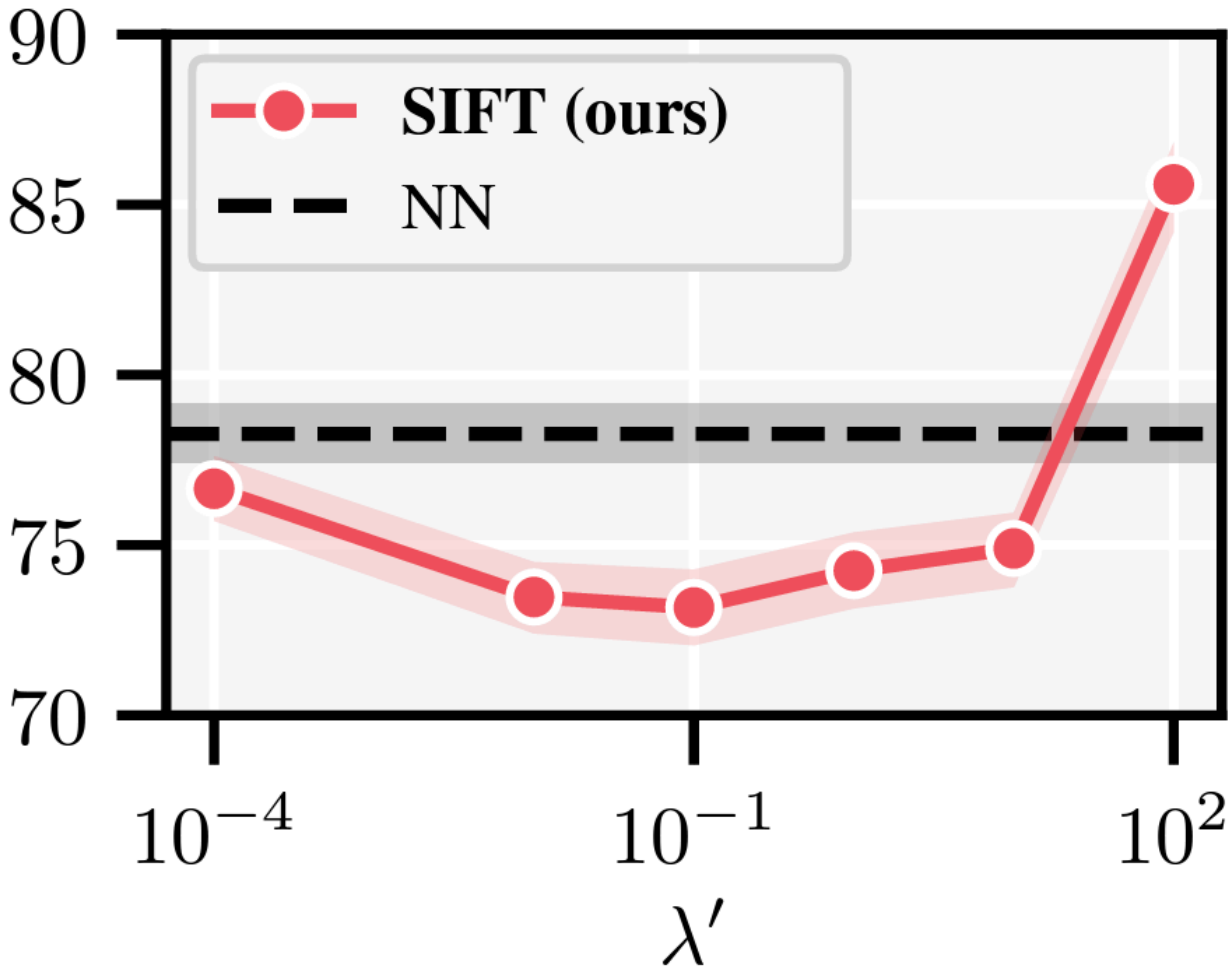
- **Active Fine-Tuning of Generalist Policies**

  Preprint

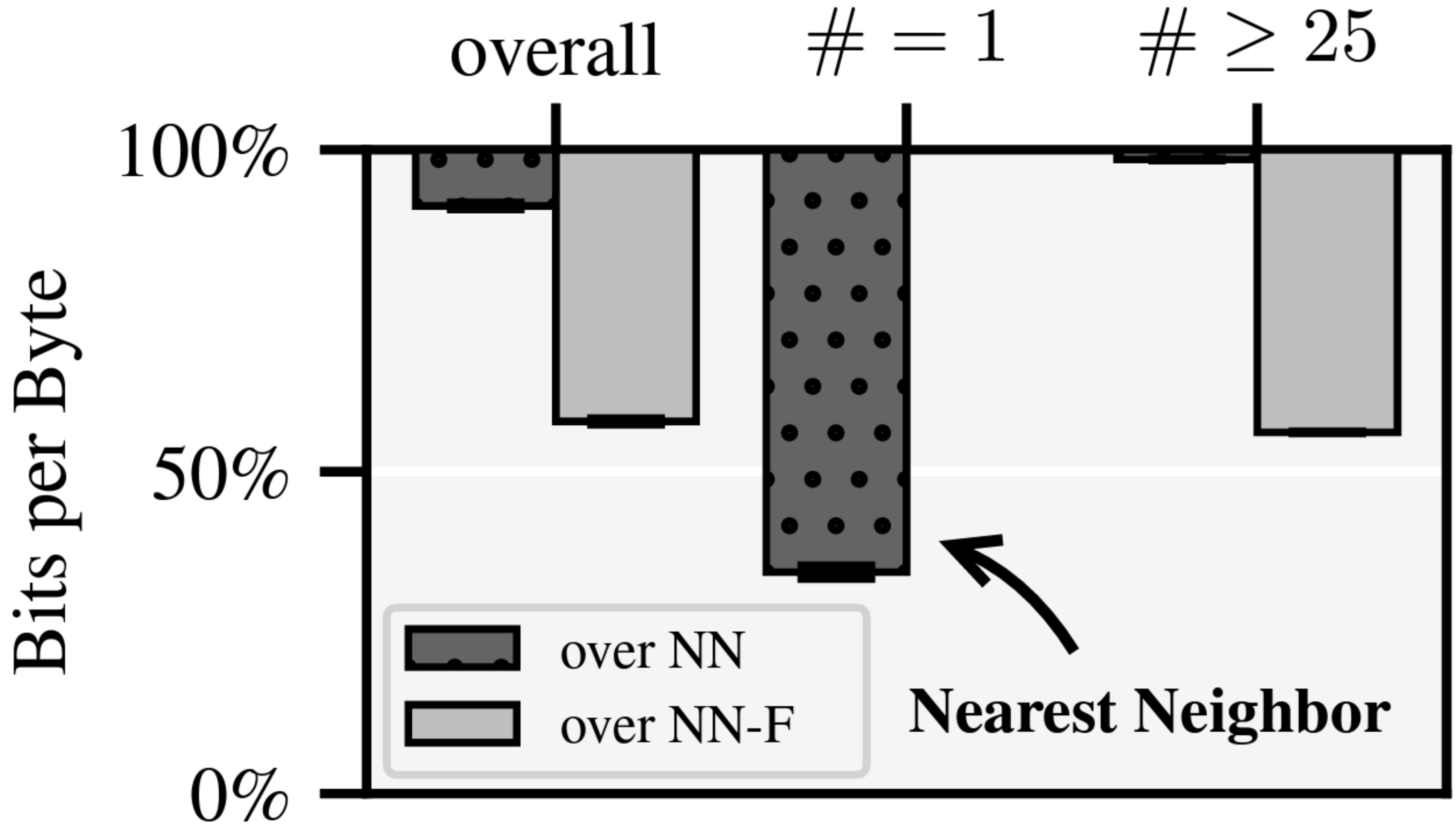


Bits Per Byte
10^0
10^{-1}
Uncertainty Estimate $\hat{\sigma}_n(x^\star)$
1.5
1.0
0.5
0.2
0.4
0.6
Uncertainty Estimate $\hat{\sigma}_n(x^\star)$
Adaptive SIFT
SIFT
100%
50%
0%
0
20
40
Test-Time Compute
Compute-Performance Ratio α
0.15
0.2
0.25
0.35
0.5
0.75
1.0
2.0


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



90
85
80
75
70
SIFT (ours)
NN
10^{-4}
10^{-1}
10^{2}
λ'


overall
$\# = 1$
$\# \geq 25$
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.