

# Foundation Models and Agents: Methods and Applications

## From MATH4AI to AI4MATH

---

**Fanghui Liu**

[fanghui.liu@sjtu.edu.cn](mailto:fanghui.liu@sjtu.edu.cn)

*Institute of Natural Sciences, School of Mathematical Sciences  
Shanghai Jiao Tong University (SJTU)*

at IEEE Summer School 2026



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY

自然科学研究院  
Institute of Natural Sciences



数学科学学院  
SCHOOL OF MATHEMATICAL SCIENCES



上海交通大学  
科学与工程计算教育部重点实验室

**SAI** SCHOOL OF  
ARTIFICIAL INTELLIGENCE  
上海交通大学人工智能学院

## ❑ MATH4AI

- generalization under scaling...
- theory contributes to practice...LLM fine-tuning

## ❑ MATH4AI

- generalization under scaling...
- theory contributes to practice...LLM fine-tuning

## ❑ AI4MATH

- Modeling mathematical reasoning of LLMs
- Machine learning theory in Lean 4 (Infrastructure)
- Harness engineering for auto-formalization

Prefer more data and larger model to obtain better performance...



```

//this code is not working like i expect - how do i fix it?

resultObjectFor = (a, callback) => {
  const data = resultObjectFor(a)
  getItems(a)
  .then(() => {
    callback(resultObjectFor(a))
  })
}

err => {
  callback(err)
}

if (err) {
  return -resultObjectFor(a)
}

return null; //error, appenderr, -resultObjectFor(a)
}

```



**ResNet-152:**  
60.3 million parameters

**Transformer:**  
340 million  
parameters

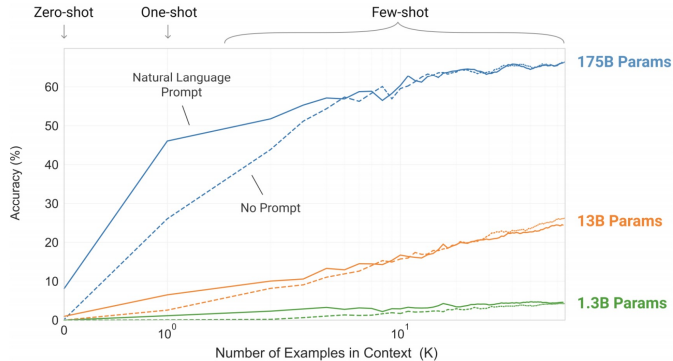
**GPT-2:**  
1.5 billion  
parameters

**GPT-3, Chat-GPT:**  
175 billion  
parameters

## Deepseek V3:

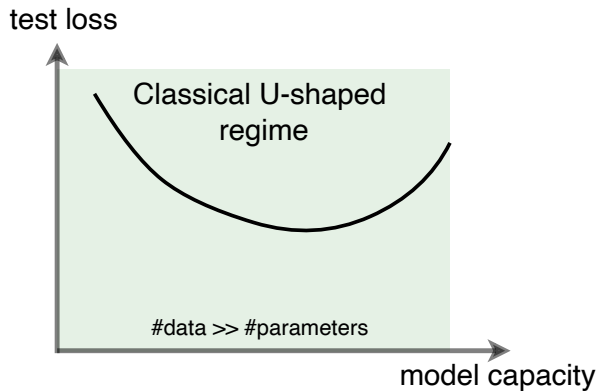


# Larger is better!

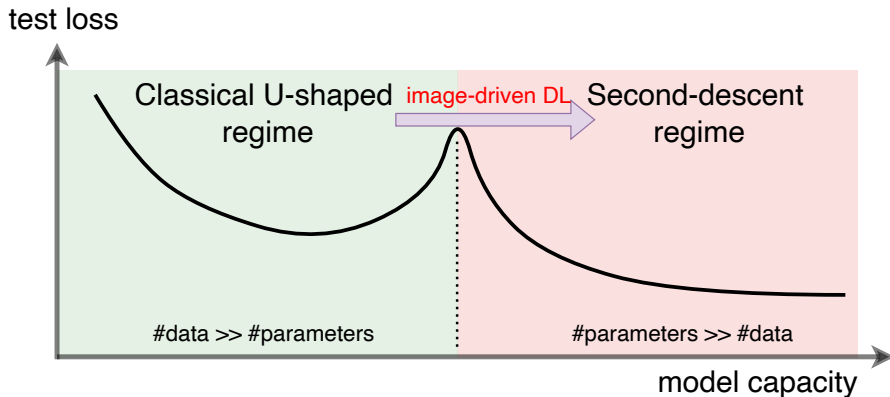


**Figure 1:** Larger models make increasingly efficient use of in-context information: source from [OpenAI](#).

## Generalization under scaling...



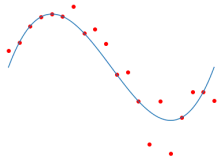
## Generalization under scaling...



Proposed explanation: double descent ([Belkin et al., 2019](#))

# A toy example: curve fitting

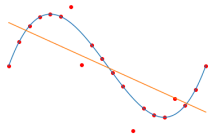
---



**(a)**  $y_i = f_{\rho}(x_i) + \varepsilon_i$

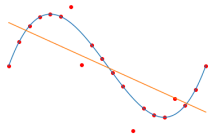
## A toy example: curve fitting

---

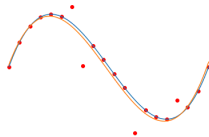


(a) under-fitting

# A toy example: curve fitting

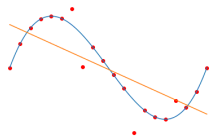


(a) under-fitting

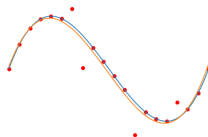


(b) sweet spot

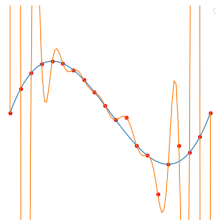
# A toy example: curve fitting



(a) under-fitting

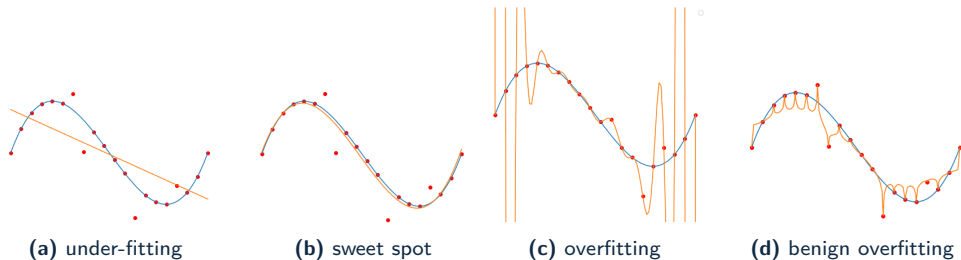


(b) sweet spot



(c) overfitting

## A toy example: curve fitting



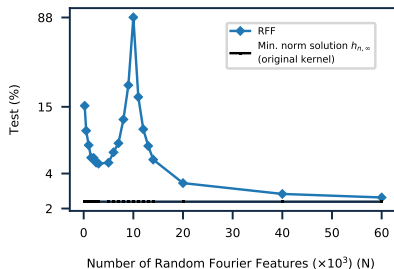
**Figure 2:** Test performance on curve fitting. Source: [OpenAI](#).



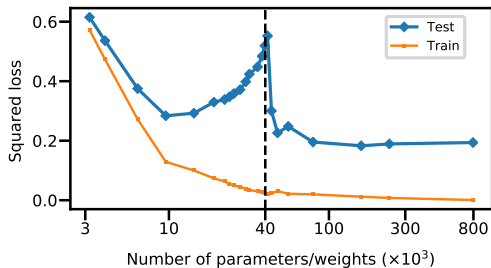
# Double descent curve in practice (I)

Typical examples:

- linear/nonlinear regression ([Hastie et al., 2022](#))
- random features, random forest, and shallow neural networks ([Belkin et al., 2019](#))



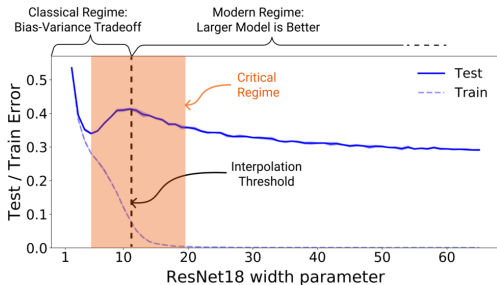
(a) Random features model



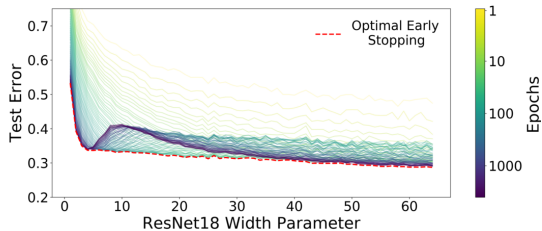
(b) A fully connected neural network

**Figure 3:** Experiments on MNIST. Source: ([Belkin et al., 2019](#)).

## Double descent curve in practice (II)



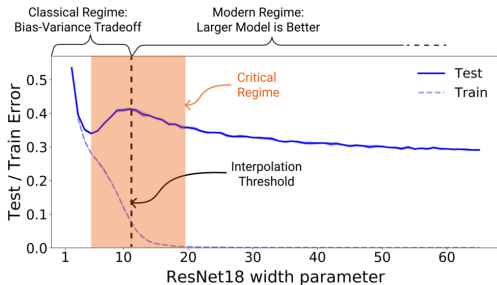
(a) ResNet18 under scaling



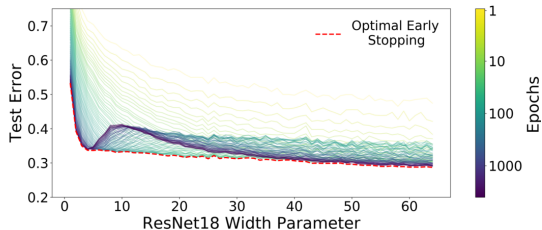
(b) Optimal early stopping

**Figure 4:** Experiment on ResNet18. Source: (Nakkiran et al., 2019) from OpenAI.

## Double descent curve in practice (II)



(a) ResNet18 under scaling



(b) Optimal early stopping

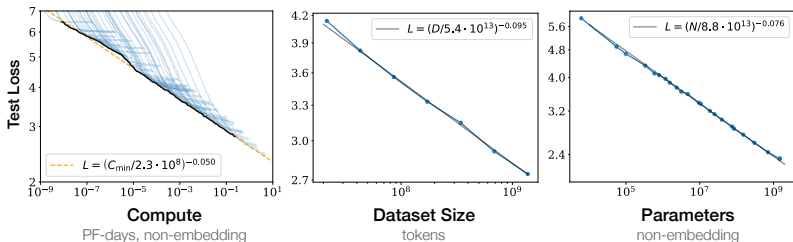
**Figure 4:** Experiment on ResNet18. Source: (Nakkiran et al., 2019) from OpenAI.

Double descent is observed under 15% label noise...

# Scaling law in the era of LLMs

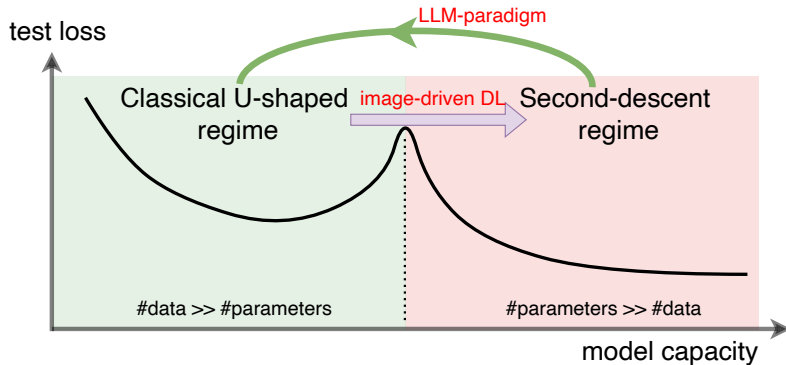
## Scaling law (Kaplan et al., 2020)

$$\text{test loss} = A \times \text{Model Size}^{-a} + B \times \text{Data Size}^{-b} + C$$



**Figure 5:** Empirical scaling laws. Source: (Kaplan et al., 2020)

# Learning paradigm in the past twenty years



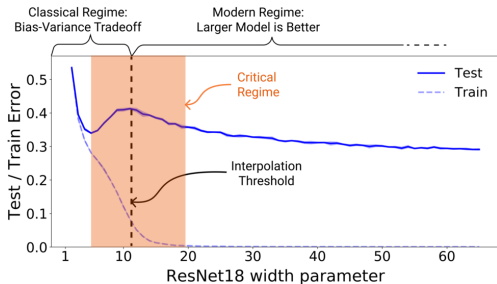
- U-shaped curve (Vapnik, 1995; Hastie et al., 2009)
- Double descent (Belkin et al., 2019)
- Scaling law (Kaplan et al., 2020)

# Model capacity

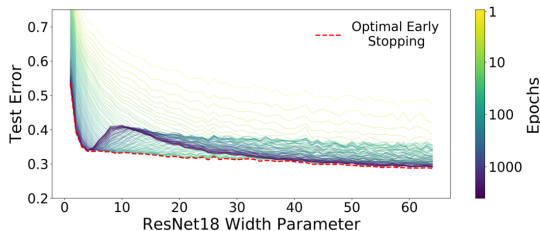
## “Remove bias-variance trade-offs from ML textbooks”

I can define **model capacity** at random and see whatever curve I want to see.

— Ben Recht, 2025



(a) Results on ResNet18 (Nakkiran et al., 2019)



(b) Optimal early stopping (Nakkiran et al., 2019).

## Model capacity: Norm-based capacity

(Bartlett, 1998)

“The size of the weights is more important than the size of the network!”

## Model capacity: Norm-based capacity

(Bartlett, 1998)

“The size of the weights is more important than the size of the network!”

- Min-norm solution (Neyshabur et al., 2015; Savarese et al., 2019; Hastie et al., 2022)
- Stable approximation...
- Applications: pruning, MoE, layerNorm, RMS Norm, EoS, mu-transfer, Muon...



## Model capacity: Norm-based capacity

(Bartlett, 1998)

“The size of the weights is more important than the size of the network!”

- Min-norm solution (Neyshabur et al., 2015; Savarese et al., 2019; Hastie et al., 2022)
- Stable approximation...
- Applications: pruning, MoE, layerNorm, RMS Norm, EoS, mu-transfer, Muon...

How do generalization curves behave under norm-based model capacity?

# Model capacity: Norm-based capacity

(Bartlett, 1998)

“The size of the weights is more important than the size of the network!”

- Min-norm solution (Neyshabur et al., 2015; Savarese et al., 2019; Hastie et al., 2022)
- Stable approximation...
- Applications: pruning, MoE, layerNorm, RMS Norm, EoS, mu-transfer, Muon...

How do generalization curves behave under norm-based model capacity?

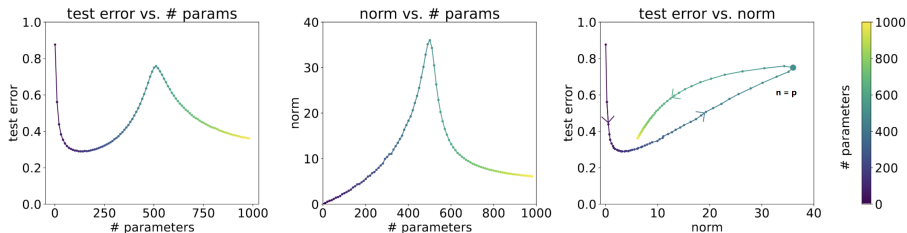
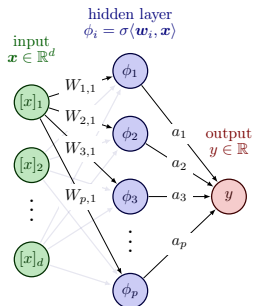


Figure 6: Stanford CS229 lecture notes (Ng and Ma, 2023, Figure 8.12).

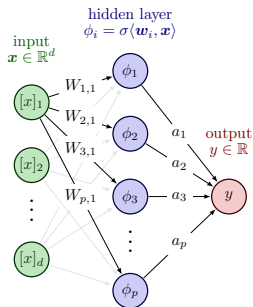
# Background: Random features ridge regression



$$f_p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{i=1}^p \mathbf{a}_i \phi(\mathbf{x}, \mathbf{w}_i), \quad \boldsymbol{\theta} := \{(\mathbf{a}_i, \mathbf{w}_i)\}_{i=1}^p$$

- $\phi : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}$ , e.g., ReLU:  
 $\phi(\mathbf{x}, \mathbf{w}) = \max(\langle \mathbf{x}, \mathbf{w} \rangle, 0)$

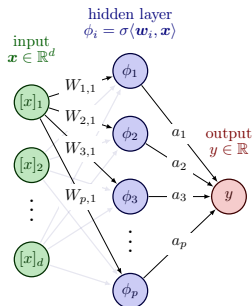
# Background: Random features ridge regression



$$f_p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{i=1}^p \mathbf{a}_i \phi(\mathbf{x}, \mathbf{w}_i), \quad \boldsymbol{\theta} := \{(\mathbf{a}_i, \mathbf{w}_i)\}_{i=1}^p$$

- $\phi : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}$ , e.g., ReLU:  
 $\phi(\mathbf{x}, \mathbf{w}) = \max(\langle \mathbf{x}, \mathbf{w} \rangle, 0)$
- Random features models (RFMs) (Rahimi and Recht, 2007; Liu et al., 2021):
  - $\{\mathbf{w}_i\}_{i=1}^p \stackrel{iid}{\sim} \mu$  for a given  $\mu \in \mathcal{P}(\mathcal{W})$
  - only train the second layer

# Background: Random features ridge regression



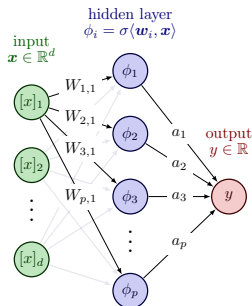
$$f_p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{i=1}^p \mathbf{a}_i \phi(\mathbf{x}, \mathbf{w}_i), \quad \boldsymbol{\theta} := \{(\mathbf{a}_i, \mathbf{w}_i)\}_{i=1}^p$$

- $\phi : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}$ , e.g., ReLU:  
 $\phi(\mathbf{x}, \mathbf{w}) = \max(\langle \mathbf{x}, \mathbf{w} \rangle, 0)$
- Random features models (RFMs) (Rahimi and Recht, 2007; Liu et al., 2021):
  - $\{\mathbf{w}_i\}_{i=1}^p \stackrel{iid}{\sim} \mu$  for a given  $\mu \in \mathcal{P}(\mathcal{W})$
  - only train the second layer

$$\hat{\mathbf{a}} := \operatorname{argmin}_{\mathbf{a} \in \mathbb{R}^p} \left\{ \sum_{i=1}^n (y_i - f(\mathbf{x}_i; \mathbf{a}))^2 + \lambda \|\mathbf{a}\|_2^2 \right\} = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I}_p)^{-1} \mathbf{Z}^\top \mathbf{y}.$$

◦  $\mathbf{Z} \in \mathbb{R}^{n \times p}$  with  $[\mathbf{Z}]_{ij} = \frac{1}{\sqrt{m}} \phi(\mathbf{x}_i, \mathbf{w}_j)$ .

# Background: Random features ridge regression



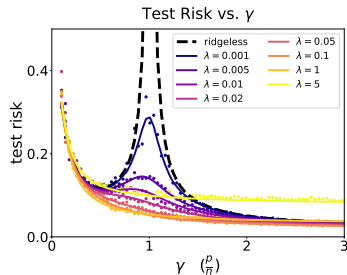
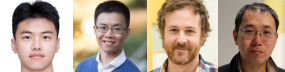
$$f_p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{i=1}^p \mathbf{a}_i \phi(\mathbf{x}, \mathbf{w}_i), \quad \boldsymbol{\theta} := \{(a_i, \mathbf{w}_i)\}_{i=1}^p$$

- $\phi : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}$ , e.g., ReLU:  
 $\phi(\mathbf{x}, \mathbf{w}) = \max(\langle \mathbf{x}, \mathbf{w} \rangle, 0)$
- Random features models (RFMs) (Rahimi and Recht, 2007; Liu et al., 2021):
  - $\{\mathbf{w}_i\}_{i=1}^p \stackrel{iid}{\sim} \mu$  for a given  $\mu \in \mathcal{P}(\mathcal{W})$
  - only train the second layer

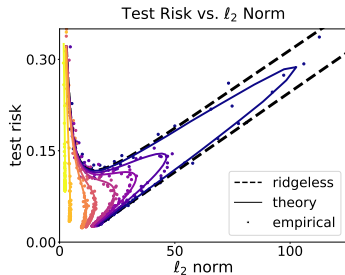
$$\hat{\mathbf{a}} := \operatorname{argmin}_{\mathbf{a} \in \mathbb{R}^p} \left\{ \sum_{i=1}^n (y_i - f(\mathbf{x}_i; \mathbf{a}))^2 + \lambda \|\mathbf{a}\|_2^2 \right\} = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I}_p)^{-1} \mathbf{Z}^\top \mathbf{y}.$$

- $\mathbf{Z} \in \mathbb{R}^{n \times p}$  with  $[\mathbf{Z}]_{ij} = \frac{1}{\sqrt{m}} \phi(\mathbf{x}_i, \mathbf{w}_j)$ .
  - Norm over the first-layer (untrained)  $\|\mathbf{W}\|_F$
  - Norm over the second-layer  $\|\hat{\mathbf{a}}\|_2^2$

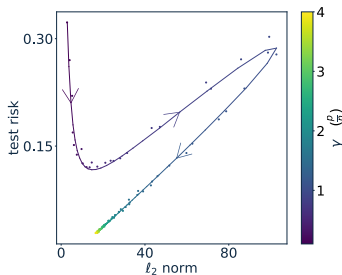
# The $\varphi$ curve (Wang, Chen, Rosasco, Liu, NeurIPS'25)



(a) Test Risk vs.  $\gamma$



(b) Test Risk vs. norm

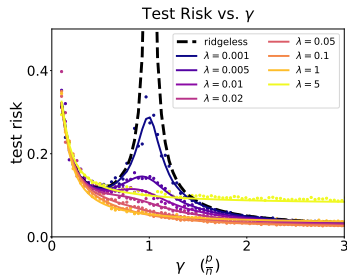
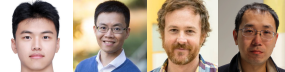


(c)  $\lambda = 0.001$

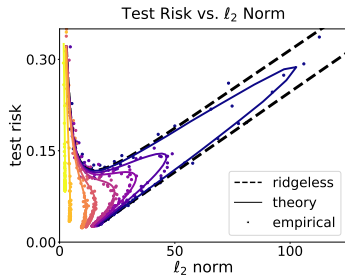
- $\gamma := p/n$ ,  $p$  : model size (width),  $n$  : data size

| $p$ ( $n = 100$ ) | 10   | 50    | 100    | 150   | 200   | 250   | 300   | 350   | 400   |
|-------------------|------|-------|--------|-------|-------|-------|-------|-------|-------|
| Test Risk         | 0.32 | 0.12  | 0.29   | 0.08  | 0.05  | 0.04  | 0.03  | 0.03  | 0.03  |
| Norm              | 2.93 | 14.66 | 102.89 | 35.26 | 24.76 | 20.67 | 18.63 | 17.47 | 16.69 |

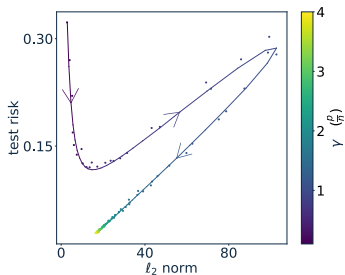
# The $\varphi$ curve (Wang, Chen, Rosasco, Liu, NeurIPS'25)



(a) Test Risk vs.  $\gamma$



(b) Test Risk vs. norm

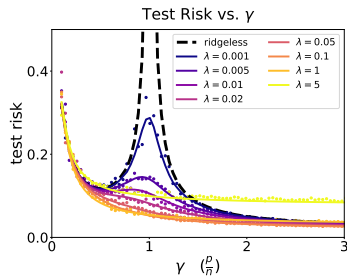
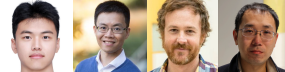


(c)  $\lambda = 0.001$

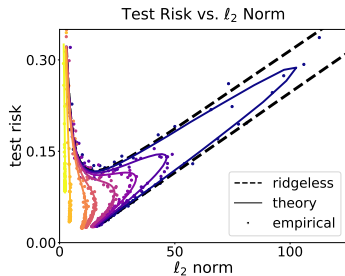
- $\gamma := p/n$ ,  $p$  : model size (width),  $n$  : data size
- Phase transition exists but double descent does not exist
- More close to **U-shaped** instead of double descent: **A  $\varphi$  paradigm**



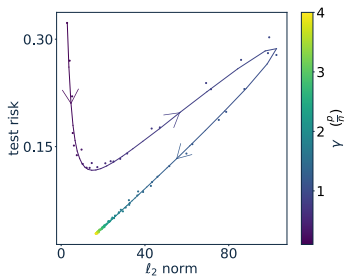
# The $\varphi$ curve (Wang, Chen, Rosasco, Liu, NeurIPS'25)



(a) Test Risk vs.  $\gamma$



(b) Test Risk vs. norm

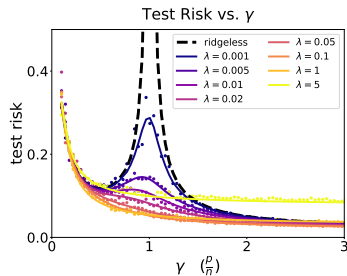
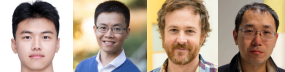


(c)  $\lambda = 0.001$

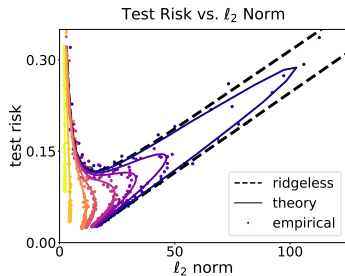
- Reshape scaling-law:

test loss =  $A \times \text{Data Size}^{-a} + B \times \text{Model Size}^{-b} + C$  with  $a, b > 0$

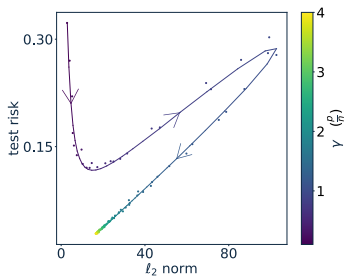
# The $\varphi$ curve (Wang, Chen, Rosasco, Liu, NeurIPS'25)



(a) Test Risk vs.  $\gamma$



(b) Test Risk vs. norm



(c)  $\lambda = 0.001$

## Reshape scaling-law:

test loss =  $A \times \text{Data Size}^{-a} + B \times \text{Model Size}^{-b} + C$  with  $a, b > 0$

test loss =  $A \times \text{Data Size}^{-a} \times \text{Norm Capacity}^{-b}$  with  $a > 0$  and  $b \in \mathbb{R}$

## Control norm by tuning $\lambda$ : L-curve (Hansen, 1992)

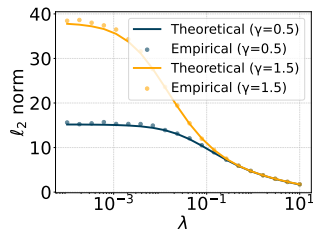
Explicit (model size) vs. Implicit (norm)

One-to-one mapping between norm and  $\lambda$

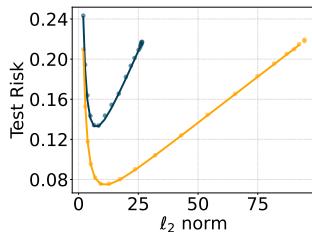
# Control norm by tuning $\lambda$ : L-curve (Hansen, 1992)

## Explicit (model size) vs. Implicit (norm)

One-to-one mapping between norm and  $\lambda$



(a) Norm vs.  $\lambda$  (varying  $\lambda$ )



(b) Risk vs. Norm (varying  $\lambda$ )

## Control norm by tuning $\lambda$ : L-curve (Hansen, 1992)

Explicit (model size) vs. Implicit (norm)

One-to-one mapping between norm and  $\lambda$

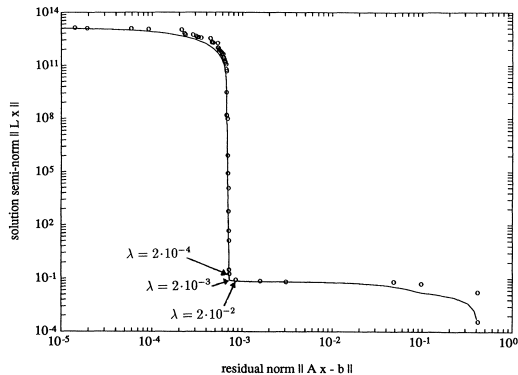


Figure 7: Source from Hansen (1992).

## An example of linear regression: Textbook level and beyond

---

- $n$  i.i.d. samples  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$  with  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}$
- $y = \langle \beta_*, \mathbf{x} \rangle + \varepsilon$ ,  $\mathbb{E}(\varepsilon) = 0$  and  $\mathbb{V}(\varepsilon) = \sigma^2$ , covariance matrix  $\Sigma = \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$
- ridge regression:  $\hat{\beta} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$

## An example of linear regression: Textbook level and beyond

- $n$  i.i.d. samples  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$  with  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}$
- $y = \langle \beta_*, \mathbf{x} \rangle + \varepsilon$ ,  $\mathbb{E}(\varepsilon) = 0$  and  $\mathbb{V}(\varepsilon) = \sigma^2$ , covariance matrix  $\Sigma = \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$
- ridge regression:  $\hat{\beta} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$

### Target: precise analysis

The expected test risk  $\mathbb{E}_\varepsilon \|\hat{\beta} - \beta_*\|_\Sigma^2$  vs. the norm  $\mathbb{E}_\varepsilon \|\hat{\beta}\|_2^2$

## An example of linear regression: Textbook level and beyond

- $n$  i.i.d. samples  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$  with  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}$
- $y = \langle \beta_*, \mathbf{x} \rangle + \varepsilon$ ,  $\mathbb{E}(\varepsilon) = 0$  and  $\mathbb{V}(\varepsilon) = \sigma^2$ , covariance matrix  $\Sigma = \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$
- ridge regression:  $\hat{\beta} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$

### Target: precise analysis

The expected test risk  $\mathbb{E}_\varepsilon \|\hat{\beta} - \beta_*\|_\Sigma^2$  vs. the norm  $\mathbb{E}_\varepsilon \|\hat{\beta}\|_2^2$

- Deterministic equivalence (Cheng and Montanari, 2024; Misiakiewicz and Saeed, 2024; Bach, 2024)

The empirical spectral measure converges to a deterministic limit.



## An example of linear regression: Textbook level and beyond

- $n$  i.i.d. samples  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$  with  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}$
- $y = \langle \beta_*, \mathbf{x} \rangle + \varepsilon$ ,  $\mathbb{E}(\varepsilon) = 0$  and  $\mathbb{V}(\varepsilon) = \sigma^2$ , covariance matrix  $\Sigma = \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$
- ridge regression:  $\hat{\beta} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$

### Target: precise analysis

The expected test risk  $\mathbb{E}_\varepsilon \|\hat{\beta} - \beta_*\|_\Sigma^2$  vs. the norm  $\mathbb{E}_\varepsilon \|\hat{\beta}\|_2^2$

- Deterministic equivalence (Cheng and Montanari, 2024; Misiakiewicz and Saeed, 2024; Bach, 2024)

$$\text{Tr}(\mathbf{X}^\top \mathbf{X} (\mathbf{X}^\top \mathbf{X} + \lambda)^{-1}) \sim \text{Tr}(\Sigma (\Sigma + \lambda_*)^{-1}), \text{ w.h.p.}$$

- $\sim$  can be **asymptotic** or **non-asymptotic** at the rate of  $\mathcal{O}(1/\sqrt{n})$ .
- $\lambda_*$  is the non-negative solution to the self-consistent equation  $n - \frac{\lambda}{\lambda_*} = \text{Tr}(\Sigma (\Sigma + \lambda_*)^{-1})$ .

## An example of linear regression: Textbook level and beyond

- $n$  i.i.d. samples  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$  with  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}$
- $y = \langle \beta_*, \mathbf{x} \rangle + \varepsilon$ ,  $\mathbb{E}(\varepsilon) = 0$  and  $\mathbb{V}(\varepsilon) = \sigma^2$ , covariance matrix  $\Sigma = \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$
- ridge regression:  $\hat{\beta} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$

### Target: precise analysis

The expected test risk  $\mathbb{E}_\varepsilon \|\hat{\beta} - \beta_*\|_\Sigma^2$  vs. the norm  $\mathbb{E}_\varepsilon \|\hat{\beta}\|_2^2$

- Deterministic equivalence (Cheng and Montanari, 2024; Misiakiewicz and Saeed, 2024; Bach, 2024)
- Bias-variance decomposition on the test risk
  - $\mathcal{B}_{\mathcal{R}, \lambda}^{\text{LS}} = \lambda^2 \langle \beta_*, (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \Sigma (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \beta_* \rangle$
  - $\mathcal{V}_{\mathcal{R}, \lambda}^{\text{LS}} = \sigma^2 \text{Tr}(\Sigma \mathbf{X}^\top \mathbf{X} (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-2})$

### Theorem (asymptotic/non-asymptotic results)

We have a bias-variance decomposition  $\mathbb{E}_\varepsilon \|\hat{\beta}\|_2^2 = \mathcal{B}_{\mathcal{N},\lambda} + \mathcal{V}_{\mathcal{N},\lambda}$ .

# Our results

## Theorem (asymptotic/non-asymptotic results)

We have a bias-variance decomposition  $\mathbb{E}_\varepsilon \|\hat{\beta}\|_2^2 = \mathcal{B}_{\mathcal{N},\lambda} + \mathcal{V}_{\mathcal{N},\lambda}$ .

For *well-behaved* data and  $\Sigma$ , we have  $\mathcal{B}_{\mathcal{N},\lambda} \sim \mathcal{B}_{\mathcal{N},\lambda}$  and  $\mathcal{V}_{\mathcal{N},\lambda} \sim \mathcal{V}_{\mathcal{N},\lambda}$ , w.h.p.

$$\mathcal{B}_{\mathcal{N},\lambda} := \langle \beta_*, \Sigma^2 (\Sigma + \lambda_*)^{-2} \beta_* \rangle + \underbrace{\frac{\text{Tr}(\Sigma(\Sigma + \lambda_*)^{-2})}{n} \frac{\lambda_*^2 \langle \beta_*, \Sigma(\Sigma + \lambda_*)^{-2} \beta_* \rangle}{1 - \frac{1}{n} \text{Tr}(\Sigma^2 (\Sigma + \lambda_*)^{-2})}}_{\mathcal{B}_{\mathcal{R},\lambda}},$$

$$\mathcal{V}_{\mathcal{N},\lambda} := \frac{\sigma^2 \text{Tr}(\Sigma(\Sigma + \lambda_*)^{-2})}{n - \text{Tr}(\Sigma^2 (\Sigma + \lambda_*)^{-2})}.$$

# Our results

## Theorem (asymptotic/non-asymptotic results)

We have a bias-variance decomposition  $\mathbb{E}_\varepsilon \|\hat{\beta}\|_2^2 = \mathcal{B}_{\mathcal{N},\lambda} + \mathcal{V}_{\mathcal{N},\lambda}$ .

For *well-behaved* data and  $\Sigma$ , we have  $\mathcal{B}_{\mathcal{N},\lambda} \sim \mathcal{B}_{\mathcal{N},\lambda}$  and  $\mathcal{V}_{\mathcal{N},\lambda} \sim \mathcal{V}_{\mathcal{N},\lambda}$ , w.h.p.

$$\mathcal{B}_{\mathcal{N},\lambda} := \langle \beta_*, \Sigma^2 (\Sigma + \lambda_*)^{-2} \beta_* \rangle + \underbrace{\frac{\text{Tr}(\Sigma(\Sigma + \lambda_*)^{-2})}{n} \frac{\lambda_*^2 \langle \beta_*, \Sigma(\Sigma + \lambda_*)^{-2} \beta_* \rangle}{1 - \frac{1}{n} \text{Tr}(\Sigma^2 (\Sigma + \lambda_*)^{-2})}}_{\mathcal{B}_{\mathcal{R},\lambda}},$$

$$\mathcal{V}_{\mathcal{N},\lambda} := \frac{\sigma^2 \text{Tr}(\Sigma(\Sigma + \lambda_*)^{-2})}{n - \text{Tr}(\Sigma^2 (\Sigma + \lambda_*)^{-2})}.$$

**Remark:** Which model capacity suffices to characterize the test risk?

- Norm-based capacity: ✓ 😊
- effective dimension-style  $\text{Tr}(\Sigma(\Sigma + \lambda I)^{-1})$ : ✗ 😞

## Example: Relationship under isotropic features ( $\Sigma = I_d$ )

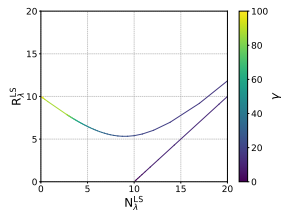
---

- Test risk  $R_\lambda$  and norm  $N_\lambda$  formulates a cubic curve (complex but precise).

## Example: Relationship under isotropic features ( $\Sigma = I_d$ )

- Test risk  $R_\lambda$  and norm  $N_\lambda$  formulates a cubic curve (complex but precise).
- min-norm interpolator ( $\lambda = 0$ ):

$$R_0 = \begin{cases} N_0 - \|\beta_*\|_2^2; & \text{in under-parameterized regimes} \\ \sqrt{[N_0 - (\|\beta_*\|_2^2 - \sigma^2)]^2 + 4\|\beta_*\|_2^2\sigma^2} - \sigma^2. & \end{cases}$$



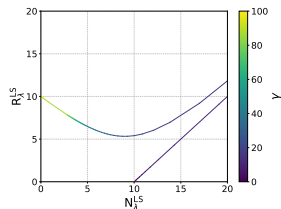
## Example: Relationship under isotropic features ( $\Sigma = I_d$ )

- Test risk  $R_\lambda$  and norm  $N_\lambda$  formulates a cubic curve (complex but precise).
- min-norm interpolator ( $\lambda = 0$ ):

$$R_0 = \begin{cases} N_0 - \|\beta_*\|_2^2; & \text{in under-parameterized regimes} \\ \sqrt{[N_0 - (\|\beta_*\|_2^2 - \sigma^2)]^2 + 4\|\beta_*\|_2^2\sigma^2} - \sigma^2. & \end{cases}$$

Why? ○ Variance is the same

- $\lambda_* = 0$  (under-parameterized)
- $\lambda_* = \frac{d-n}{n}$  (over-parameterized)





## Example: Relationship under isotropic features ( $\Sigma = I_d$ )

□ Test risk  $R_\lambda$  and norm  $N_\lambda$  formulates a cubic curve (complex but precise).

- min-norm interpolator ( $\lambda = 0$ ):

$$R_0 = \begin{cases} N_0 - \|\beta_*\|_2^2; & \text{in under-parameterized regimes} \\ \sqrt{[N_0 - (\|\beta_*\|_2^2 - \sigma^2)]^2 + 4\|\beta_*\|_2^2\sigma^2} - \sigma^2. & \end{cases}$$

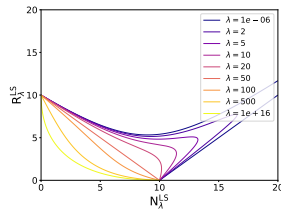
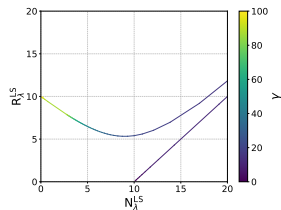
Why? ○ Variance is the same

○  $\lambda_* = 0$  (under-parameterized)

○  $\lambda_* = \frac{d-n}{n}$  (over-parameterized)

- optimal regularization  $\lambda = \frac{d\sigma^2}{\|\beta^*\|_2^2}$  (Wu and Xu, 2020):

$$R_\lambda = \|\beta_*\|_2^2 - N_\lambda$$



## Example: Relationship under isotropic features ( $\Sigma = I_d$ )

□ Test risk  $R_\lambda$  and norm  $N_\lambda$  formulates a cubic curve (complex but precise).

- min-norm interpolator ( $\lambda = 0$ ):

$$R_0 = \begin{cases} N_0 - \|\beta_*\|_2^2; & \text{in under-parameterized regimes} \\ \sqrt{[N_0 - (\|\beta_*\|_2^2 - \sigma^2)]^2 + 4\|\beta_*\|_2^2\sigma^2} - \sigma^2. & \end{cases}$$

Why? ○ Variance is the same

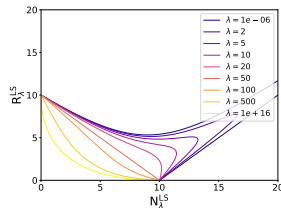
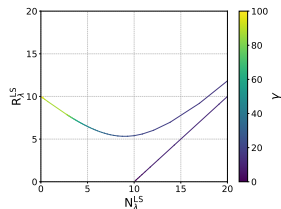
○  $\lambda_* = 0$  (under-parameterized)

○  $\lambda_* = \frac{d-n}{n}$  (over-parameterized)

- optimal regularization  $\lambda = \frac{d\sigma^2}{\|\beta^*\|_2^2}$  (Wu and Xu, 2020):

$$R_\lambda = \|\beta_*\|_2^2 - N_\lambda$$

- $\lambda \rightarrow \infty$ :  $R_\lambda = (\|\beta_*\|_2 - \sqrt{N_\lambda})^2$

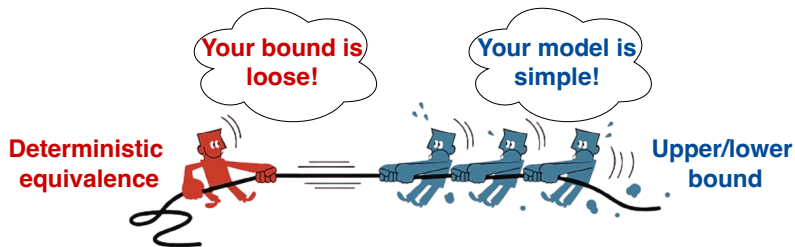


## Precise analysis via deterministic equivalence

- ❑ Precisely describe the learning curve.
  - phase transitions, (non-)monotonicity, etc.
- ❑ Enables *accurate comparison* between estimators/algorithms.
  - **Foundations of scaling law**: data or parameter under the same budget, etc.

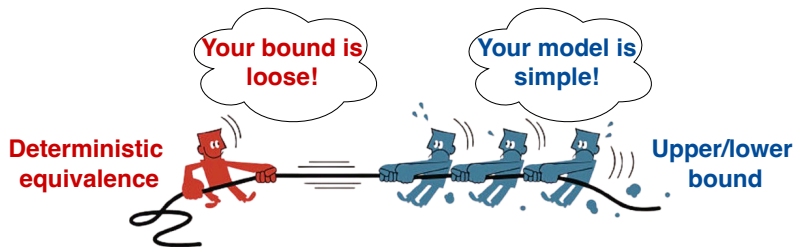
# Precise analysis via deterministic equivalence

- ❑ Precisely describe the learning curve.
  - phase transitions, (non-)monotonicity, etc.
- ❑ Enables *accurate comparison* between estimators/algorithms.
  - **Foundations of scaling law:** data or parameter under the same budget, etc.



# Precise analysis via deterministic equivalence

- ❑ Precisely describe the learning curve.
  - phase transitions, (non-)monotonicity, etc.
- ❑ Enables *accurate comparison* between estimators/algorithms.
  - **Foundations of scaling law:** data or parameter under the same budget, etc.



Is  $\ell_2$  norm-based capacity **best** for characterizing generalization?

# Which model capacity is suitable (for neural networks)?

**Table 1:** Complexity measures compared in the empirical study (Jiang et al., 2020).

| name                     | definition                          | rank correlation |
|--------------------------|-------------------------------------|------------------|
| Parameter Frobenius norm | $\sum_{i=1}^L \ \mathbf{W}_i\ _F^2$ | 0.073            |

# Which model capacity is suitable (for neural networks)?

**Table 1:** Complexity measures compared in the empirical study (Jiang et al., 2020).

| name                                 | definition                                                                                                                     | rank correlation |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|------------------|
| Parameter Frobenius norm             | $\sum_{i=1}^L \ \mathbf{W}_i\ _F^2$                                                                                            | 0.073            |
| Frobenius distance to initialization | $\sum_{i=1}^L \ \mathbf{W}_i - \mathbf{W}_i^0\ _F^2$                                                                           | -0.263           |
| Spectral complexity                  | $\prod_{i=1}^L \ \mathbf{W}_i\  \left( \sum_{i=1}^L \frac{\ \mathbf{W}_i\ _{2,1}^{3/2}}{\ \mathbf{W}_i\ ^{3/2}} \right)^{2/3}$ | <b>-0.537</b>    |
| Fisher-Rao                           | $\frac{(L+1)^2}{n} \sum_{i=1}^n \langle \mathbf{W}, \nabla_{\mathbf{W}} \ell(h_{\mathbf{W}}(\mathbf{x}_i), y_i) \rangle$       | 0.078            |

# Which model capacity is suitable (for neural networks)?

**Table 1:** Complexity measures compared in the empirical study (Jiang et al., 2020).

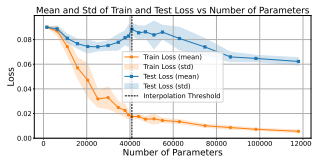
| name                                 | definition                                                                                                                     | rank correlation |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|------------------|
| Parameter Frobenius norm             | $\sum_{i=1}^L \ \mathbf{w}_i\ _F^2$                                                                                            | 0.073            |
| Frobenius distance to initialization | $\sum_{i=1}^L \ \mathbf{w}_i - \mathbf{w}_i^0\ _F^2$                                                                           | -0.263           |
| Spectral complexity                  | $\prod_{i=1}^L \ \mathbf{w}_i\  \left( \sum_{i=1}^L \frac{\ \mathbf{w}_i\ _{2,1}^{3/2}}{\ \mathbf{w}_i\ ^{3/2}} \right)^{2/3}$ | <b>-0.537</b>    |
| Fisher-Rao                           | $\frac{(L+1)^2}{n} \sum_{i=1}^n \langle \mathbf{W}, \nabla_{\mathbf{W}} \ell(h_{\mathbf{W}}(\mathbf{x}_i), y_i) \rangle$       | 0.078            |
| Path-norm                            | $\sum_{(i_0, \dots, i_L)} \prod_{j=1}^L (\mathbf{w}_{i_j, i_{j-1}})^2$                                                         | <b>0.373</b>     |



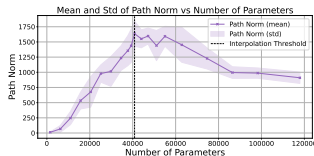
# Which model capacity is suitable (for neural networks)?

**Table 1:** Complexity measures compared in the empirical study (Jiang et al., 2020).

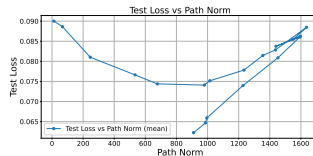
| name                                 | definition                                                                                                                     | rank correlation |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|------------------|
| Parameter Frobenius norm             | $\sum_{i=1}^L \ \mathbf{w}_i\ _F^2$                                                                                            | 0.073            |
| Frobenius distance to initialization | $\sum_{i=1}^L \ \mathbf{w}_i - \mathbf{w}_i^0\ _F^2$                                                                           | -0.263           |
| Spectral complexity                  | $\prod_{i=1}^L \ \mathbf{w}_i\  \left( \sum_{i=1}^L \frac{\ \mathbf{w}_i\ _{2,1}^{3/2}}{\ \mathbf{w}_i\ ^{3/2}} \right)^{2/3}$ | <b>-0.537</b>    |
| Fisher-Rao                           | $\frac{(L+1)^2}{n} \sum_{i=1}^n \langle \mathbf{W}, \nabla_{\mathbf{W}} \ell(h_{\mathbf{W}}(\mathbf{x}_i), y_i) \rangle$       | 0.078            |
| Path-norm                            | $\sum_{(i_0, \dots, i_L)} \prod_{j=1}^L (\mathbf{w}_{i_j, i_{j-1}})^2$                                                         | <b>0.373</b>     |



(a) Test (training) Loss vs.  $p$



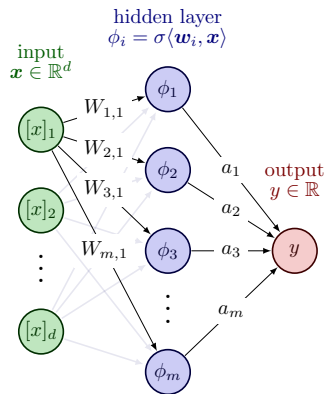
(b) Path-norm vs.  $p$



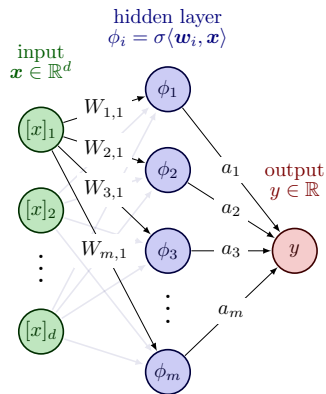
(c) Test Loss vs. Path-norm

**Figure 8:** Experiments on two-layer neural networks. Source: Wang et al. (2025).

## Two-layer neural networks, path norm



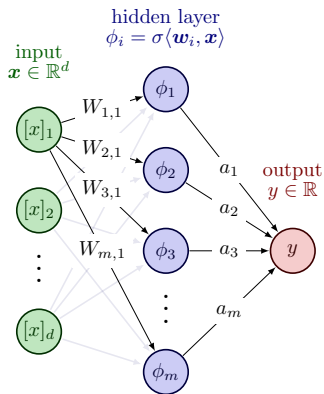
## Two-layer neural networks, path norm



$\ell_1$ -path norm (Neyshabur et al., 2015)

$$\|\theta\|_{\mathcal{P}} := \frac{1}{m} \sum_{k=1}^m |a_k| \|\mathbf{w}_k\|_1$$

## Two-layer neural networks, path norm



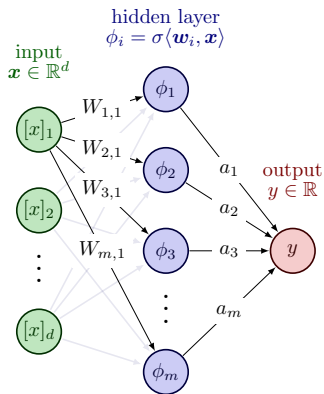
$\ell_1$ -path norm (Neyshabur et al., 2015)

$$\|\boldsymbol{\theta}\|_{\mathcal{P}} := \frac{1}{m} \sum_{k=1}^m |a_k| \|\mathbf{w}_k\|_1$$

- equivalent to Barron spaces  $\mathcal{B}$  (Barron, 1993; E et al., 2021)

$$\mathcal{B} := \cup_{\mu \in \mathcal{P}(\mathcal{W})} \{f_a : \|\mathbf{a}\|_{L^2(\mu)} < \infty\}$$

## Two-layer neural networks, path norm



$\ell_1$ -path norm (Neyshabur et al., 2015)

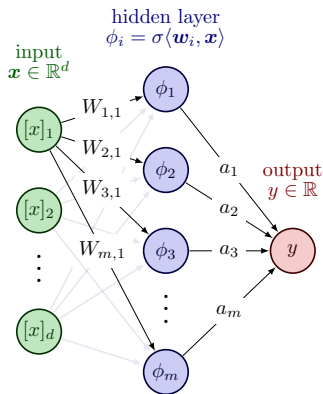
$$\|\boldsymbol{\theta}\|_{\mathcal{P}} := \frac{1}{m} \sum_{k=1}^m |a_k| \|\mathbf{w}_k\|_1$$

- equivalent to Barron spaces  $\mathcal{B}$  (Barron, 1993; E et al., 2021)

$$\mathcal{B} := \cup_{\mu \in \mathcal{P}(\mathcal{W})} \{f_a : \|\mathbf{a}\|_{L^2(\mu)} < \infty\}$$

- Variation in only a few directions (Parhi and Nowak, 2022)

# Two-layer neural networks, path norm



$\ell_1$ -path norm (Neyshabur et al., 2015)

$$\|\theta\|_{\mathcal{P}} := \frac{1}{m} \sum_{k=1}^m |a_k| \|\mathbf{w}_k\|_1$$

- equivalent to Barron spaces  $\mathcal{B}$  (Barron, 1993; E et al., 2021)

$$\mathcal{B} := \bigcup_{\mu \in \mathcal{P}(\mathcal{W})} \{f_a : \|\mathbf{a}\|_{L^2(\mu)} < \infty\}$$

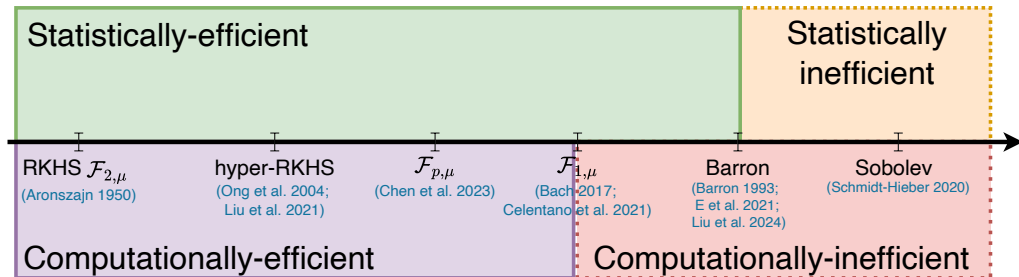
- Variation in only a few directions (Parhi and Nowak, 2022)

## Theorem (Informal results from Liu et al. (2024))

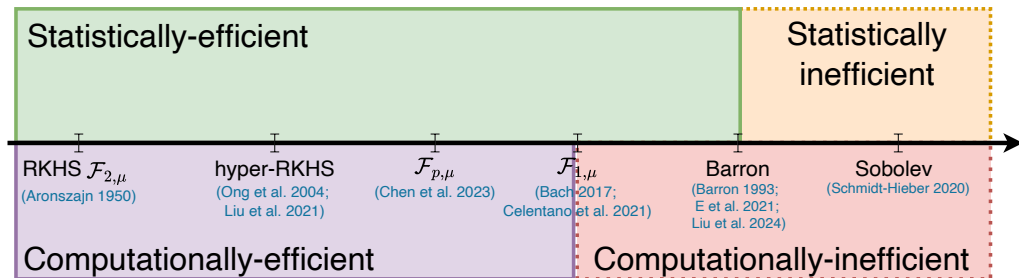
To achieve  $\epsilon$ -excess risk, kernel methods require  $\Omega(\epsilon^{-d})$  samples but

- Two-layer neural networks require  $\mathcal{O}_d(\epsilon^{-\frac{2d+2}{d+2}})$  samples. [smaller than  $\epsilon^{-2}$ ]

# Which function class can be efficiently learned by neural networks



# Which function class can be efficiently learned by neural networks



## □ Single/multi-index models

- TCS: ReLU neurons (Chen and Narayanan, 2023)
- Low-dimensional polynomials (Arous et al., 2021; Damian et al., 2022; Abbe et al., 2022; Lee et al., 2024; Dandi et al., 2024)



**How does theory contribute to  
practice?**

---

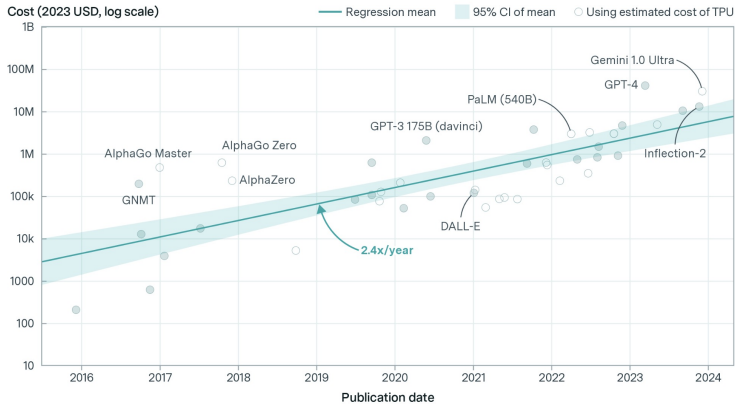
# How does theory contribute to practice?

---

## ❑ LoRA in LLM fine-tuning...

- understanding: subspace alignment
- theory-grounded algorithm for efficiency improvement
- clarify misconceptions

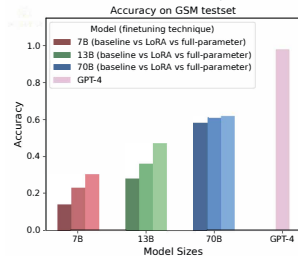
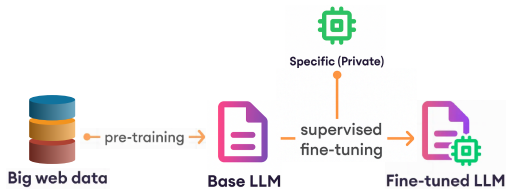
# Cost for Pre-training



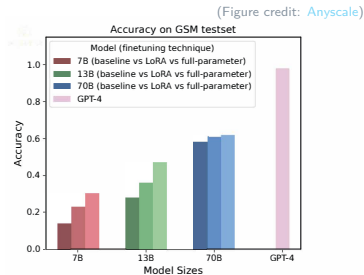
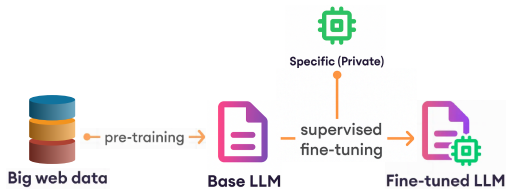
(Figure credit: [Epoch AI](#))

# From pre-training to (parameter-efficient) fine-tuning

(Figure credit: [Anyscale](#))



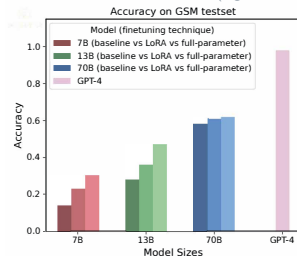
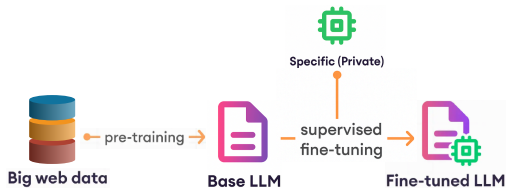
# From pre-training to (parameter-efficient) fine-tuning



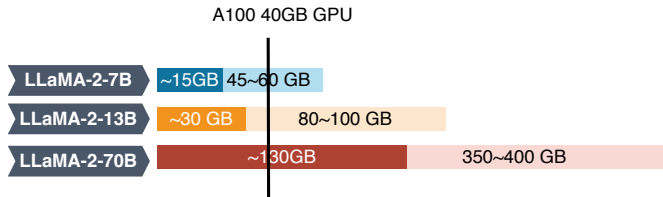
Full fine-tuning vs. parameter-efficient fine-tuning

# From pre-training to (parameter-efficient) fine-tuning

(Figure credit: Anyscale)

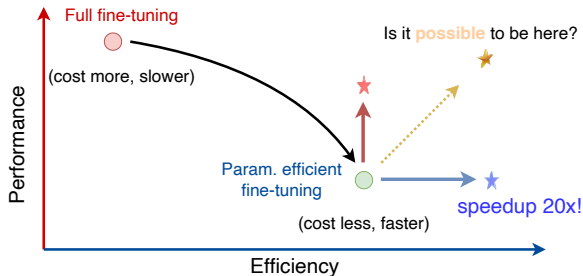
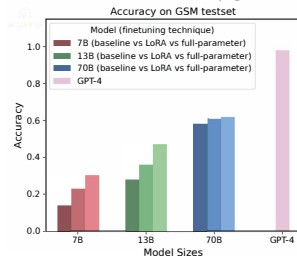
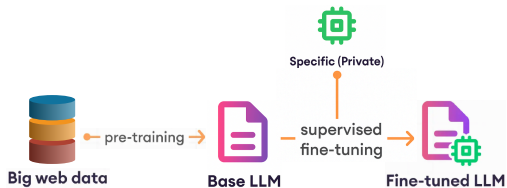


## Full fine-tuning vs. parameter-efficient fine-tuning



# From pre-training to (parameter-efficient) fine-tuning

(Figure credit: Anyscale)



# LoRA: Low-rank adaption

Published as a conference paper at ICLR 2022

## LoRA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS

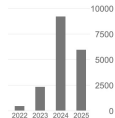
Edward Hu\*   Yelong Shen\*   Phillip Wallis   Zeyuan Allen-Zhu  
Yuanzhi Li   Shean Wang   Lu Wang   Weizhu Chen

Microsoft Corporation

edward.hu@mila.quebec

{yeshe, phwallis, zeyuana, swang, luw, wzchen}@microsoft.com

yuanzhil@andrew.cmu.edu





# LoRA: Low-rank adaption

Published as a conference paper at ICLR 2022

## LoRA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS

Edward Hu\*  
Yuanzhi Li

Yelong Shen\*  
Shean Wang

Phillip Wallis  
Lu Wang

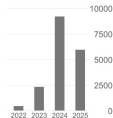
Zeyuan Allen-Zhu  
Weizhu Chen

Microsoft Corporation

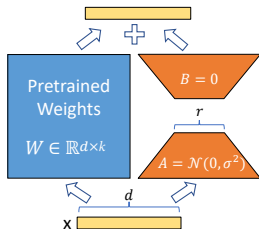
edward.hu@mila.quebec

{yeshe, phwallis, zeyuana, swang, luw, wzchen}@microsoft.com

yuanzhi@andrew.cmu.edu



$$W^{\text{FT}} = W^{\text{pre}} + \Delta \in \mathbb{R}^{d \times k}$$



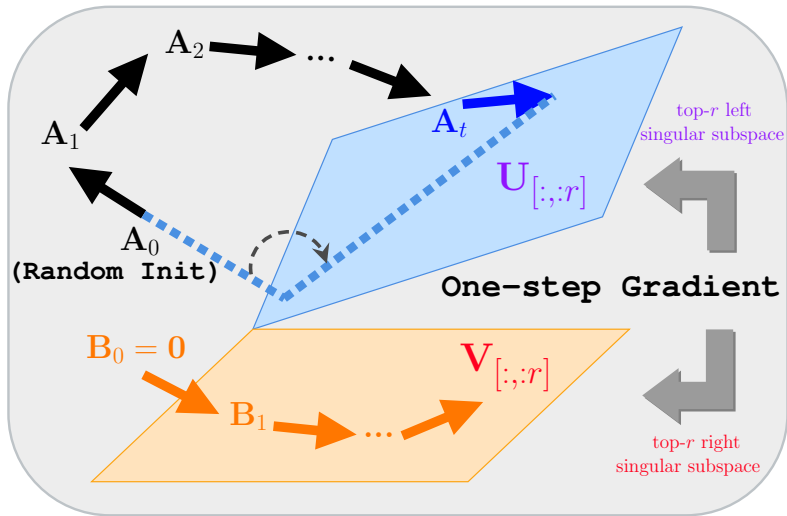
### ■ Formulation:

$$\Delta \approx AB \text{ with } A \in \mathbb{R}^{d \times r} \text{ and } B \in \mathbb{R}^{r \times k}$$

### ■ Initialization:

$$[A_0]_{ij} \sim \mathcal{N}(0, \alpha^2) \quad \text{and} \quad [B_0]_{ij} = 0. \\ (\text{LoRA-init.})$$

# Pipeline



## Problem setting and assumptions

□ (Downstream) data : with unknown low-rank feature shift  $\text{rank}(\Delta) = r^*$

$$\tilde{y} = \begin{cases} (\mathbf{W}^\natural + \Delta)^\top \tilde{\mathbf{x}}, & \tilde{\mathbf{x}} \stackrel{i.i.d.}{\sim} \text{sub-Gaussian}, & \text{linear} \\ \sigma[(\mathbf{W}^\natural + \Delta)^\top \tilde{\mathbf{x}}], & \tilde{\mathbf{x}} \stackrel{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}_d) & \text{nonlinear} \end{cases}.$$

## Problem setting and assumptions

- (Downstream) data : with unknown low-rank feature shift  $\text{rank}(\Delta) = r^*$

$$\tilde{y} = \begin{cases} (\mathbf{W}^\natural + \Delta)^\top \tilde{\mathbf{x}}, & \tilde{\mathbf{x}} \stackrel{i.i.d.}{\sim} \text{sub-Gaussian}, & \text{linear} \\ \sigma[(\mathbf{W}^\natural + \Delta)^\top \tilde{\mathbf{x}}], & \tilde{\mathbf{x}} \stackrel{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}_d) & \text{nonlinear} \end{cases}.$$

- Model : LoRA starting from **known** pre-trained  $\mathbf{W}^\natural \in \mathbb{R}^{d \times k}$

$$f(\mathbf{x}; \mathbf{A}, \mathbf{B}) := \begin{cases} (\mathbf{W}^\natural + \mathbf{A}\mathbf{B})^\top \tilde{\mathbf{x}} & \text{linear} \\ \sigma[(\mathbf{W}^\natural + \mathbf{A}\mathbf{B})^\top \tilde{\mathbf{x}}] & \text{nonlinear} \end{cases}.$$

## Problem setting and assumptions

- (Downstream) data : with unknown low-rank feature shift  $\text{rank}(\Delta) = r^*$

$$\tilde{\mathbf{y}} = \begin{cases} (\mathbf{W}^\natural + \Delta)^\top \tilde{\mathbf{x}}, & \tilde{\mathbf{x}} \stackrel{i.i.d.}{\sim} \text{sub-Gaussian}, & \text{linear} \\ \sigma[(\mathbf{W}^\natural + \Delta)^\top \tilde{\mathbf{x}}], & \tilde{\mathbf{x}} \stackrel{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}_d) & \text{nonlinear} \end{cases}.$$

- Model : LoRA starting from **known** pre-trained  $\mathbf{W}^\natural \in \mathbb{R}^{d \times k}$

$$f(\mathbf{x}; \mathbf{A}, \mathbf{B}) := \begin{cases} (\mathbf{W}^\natural + \mathbf{A}\mathbf{B})^\top \tilde{\mathbf{x}} & \text{linear} \\ \sigma[(\mathbf{W}^\natural + \mathbf{A}\mathbf{B})^\top \tilde{\mathbf{x}}] & \text{nonlinear} \end{cases}.$$

- Algorithm : Squared loss, gradient descent

$$\begin{cases} L(\mathbf{W}) = \frac{1}{2N} \sum_{i=1}^N \|f(\mathbf{x}_i; \mathbf{W}) - \tilde{\mathbf{y}}_i\|_2^2, & \text{Full fine-tuning} \\ L(\mathbf{A}, \mathbf{B}) = \frac{1}{2N} \sum_{i=1}^N \|f(\mathbf{x}_i; \mathbf{A}, \mathbf{B}) - \tilde{\mathbf{y}}_i\|_2^2, & \text{LoRA} \end{cases}.$$

## Problem setting and assumptions

- (Downstream) data : with unknown low-rank feature shift  $\text{rank}(\Delta) = r^*$

$$\tilde{\mathbf{y}} = \begin{cases} (\mathbf{W}^\natural + \Delta)^\top \tilde{\mathbf{x}}, & \tilde{\mathbf{x}} \stackrel{i.i.d.}{\sim} \text{sub-Gaussian}, & \text{linear} \\ \sigma[(\mathbf{W}^\natural + \Delta)^\top \tilde{\mathbf{x}}], & \tilde{\mathbf{x}} \stackrel{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}_d) & \text{nonlinear} \end{cases}.$$

- Model : LoRA starting from **known** pre-trained  $\mathbf{W}^\natural \in \mathbb{R}^{d \times k}$

$$f(\mathbf{x}; \mathbf{A}, \mathbf{B}) := \begin{cases} (\mathbf{W}^\natural + \mathbf{A}\mathbf{B})^\top \tilde{\mathbf{x}} & \text{linear} \\ \sigma[(\mathbf{W}^\natural + \mathbf{A}\mathbf{B})^\top \tilde{\mathbf{x}}] & \text{nonlinear} \end{cases}.$$

- Algorithm : Squared loss, gradient descent

$$\begin{cases} L(\mathbf{W}) = \frac{1}{2N} \sum_{i=1}^N \|f(\mathbf{x}_i; \mathbf{W}) - \tilde{\mathbf{y}}_i\|_2^2, & \text{Full fine-tuning} \\ L(\mathbf{A}, \mathbf{B}) = \frac{1}{2N} \sum_{i=1}^N \|f(\mathbf{x}_i; \mathbf{A}, \mathbf{B}) - \tilde{\mathbf{y}}_i\|_2^2, & \text{LoRA} \end{cases}.$$

- Evaluate  $\|\mathbf{A}_t \mathbf{B}_t - \Delta\|_F$  : optimization and generalization!

$$\mathbb{E}_{\tilde{\mathbf{x}}} \left\| \tilde{\mathbf{y}} - \sigma(\mathbf{W}^\natural + \mathbf{A}_t \mathbf{B}_t)^\top \tilde{\mathbf{x}} \right\|_2^2 \lesssim \|\mathbf{A}_t \mathbf{B}_t - \Delta\|_F^2$$

## Our results: Alignment on $B_t$

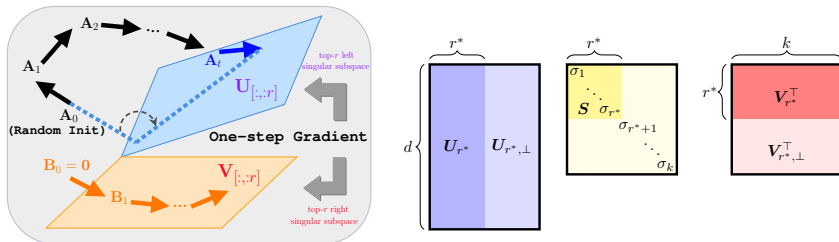
□ one-step full gradient:  $\mathbf{G} \in \mathbb{R}^{d \times k}$  and  $\text{rank}(\mathbf{G}) = r^*$

$$\mathbf{G} := -\nabla_{\mathbf{W}} L(\mathbf{W}^\natural) = \frac{1}{N} \widetilde{\mathbf{X}}^\top (\widetilde{\mathbf{Y}} - \widetilde{\mathbf{X}} \mathbf{W}^\natural) = \frac{1}{N} \widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} \Delta.$$

## Our results: Alignment on $B_t$

□ one-step full gradient:  $G \in \mathbb{R}^{d \times k}$  and  $\text{rank}(G) = r^*$

$$G := -\nabla_{\mathbf{W}} L(\mathbf{W}^\natural) = \frac{1}{N} \widetilde{\mathbf{X}}^\top (\widetilde{\mathbf{Y}} - \widetilde{\mathbf{X}} \mathbf{W}^\natural) = \frac{1}{N} \widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} \Delta.$$



### Theorem (Alignment between $G$ and $B_t$ )

For the linear setting, LoRA trained by gradient descent admits

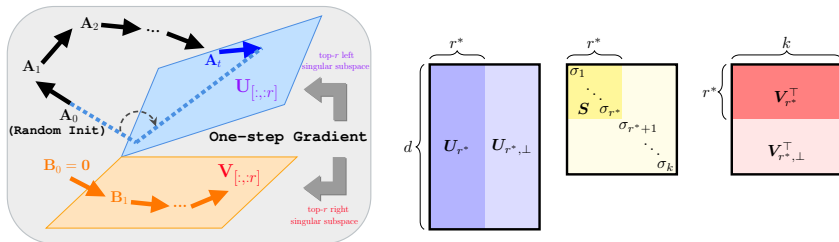
$$\angle(V_{r^*}(B_t), V_{r^*}(G)) = 0, \quad \forall t \in \mathbb{N}_+.$$



## Our results: Alignment on $B_t$

□ one-step full gradient:  $G \in \mathbb{R}^{d \times k}$  and  $\text{rank}(G) = r^*$

$$G := -\nabla_{\mathbf{W}} L(\mathbf{W}^\natural) = \frac{1}{N} \widetilde{\mathbf{X}}^\top (\widetilde{\mathbf{Y}} - \widetilde{\mathbf{X}} \mathbf{W}^\natural) = \frac{1}{N} \widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} \Delta.$$



### Theorem (Alignment between $G$ and $B_t$ )

For the linear setting, LoRA trained by gradient descent admits

$$\angle(V_{r^*}(B_t), V_{r^*}(G)) = 0, \quad \forall t \in \mathbb{N}_+.$$

**Remark:**  $B_1 = \eta_1 A_0^\top G$  with  $\text{Rank}(B_1) \leq r^*$

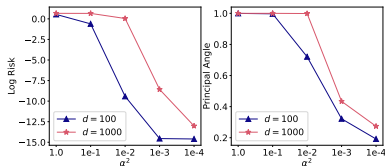


# Our results: Alignment on $A_t$

## Theorem (Informal, LoRA initialization)

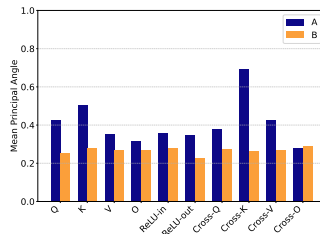
For  $r \geq r^*$ ,  $[A_0]_{ij} \sim \mathcal{N}(0, \alpha^2)$ , for any  $\epsilon \in (0, 1)$ , choosing  $\alpha = \mathcal{O}(\epsilon d^{-\frac{3}{4}} \kappa^{\frac{1}{4}} - \frac{1}{2})$ , running GD with  $t^* = \Theta(\ln d)$  steps, then we have

$$\angle(U_{r^*}(A_{t^*}), U_{r^*}(G)) \lesssim \epsilon, w.h.p.$$



**Figure 9:** Left: the risk  $\frac{1}{2} \|A_t B_t - \Delta\|_F^2$ .

Right: the principal angle is  $\min_t \|U_{r^*, \perp}^\top(G) U_{r^*}(A_t)\|_{op}$ .



**Figure 10:** Principal angle of fine-tuning T5 on MRPC.

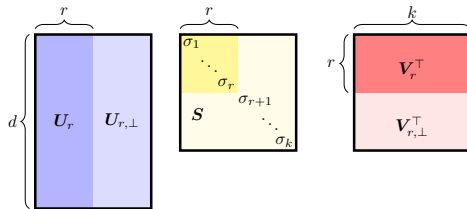
# Algorithm design principle

□ SVD:  $G = USV^\top$

$$A_0 = U_{[:,1:r]} S_{[1:r]}^{\frac{1}{2}} \cdot$$

$$B_0 = S_{[1:r]}^{\frac{1}{2}} V_{[:,1:r]}^\top \cdot$$

(Spec-init.)



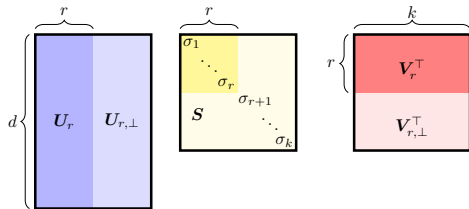
# Algorithm design principle

□ SVD:  $G = USV^\top$

$$A_0 = U_{[:,1:r]} S_{[1:r]}^{\frac{1}{2}}$$

$$B_0 = S_{[1:r]}^{\frac{1}{2}} V_{[:,1:r]}^\top$$

(Spec-init.)



**Key Message:** we can “escape” the alignment stage

Under (Spec-init.), for both linear/nonlinear models, we can directly achieve the alignment at initialization.

$$\|A_0 B_0 - \Delta\|_F \text{ is small, w.h.p.}$$

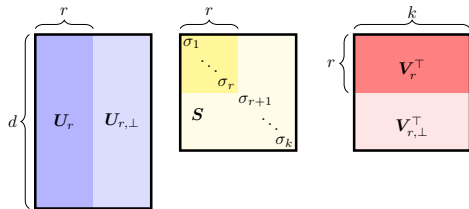
# Algorithm design principle

□ SVD:  $G = USV^\top$

$$A_0 = U_{[:,1:r]} S_{[1:r]}^{\frac{1}{2}}$$

$$B_0 = S_{[1:r]}^{\frac{1}{2}} V_{[:,1:r]}^\top$$

(Spec-init.)



**Key Message:** we can “escape” the alignment stage

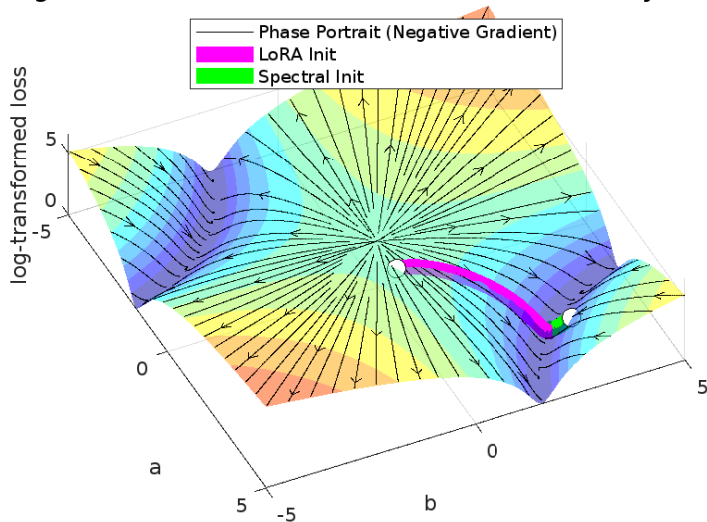
Under (Spec-init.), for both linear/nonlinear models, we can directly achieve the alignment at initialization.

$$\|A_0 B_0 - \Delta\|_F \text{ is small, w.h.p.}$$

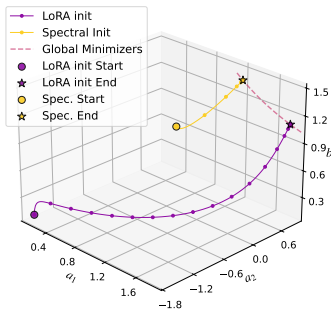
The “best” initialization strategy!

## “Best” initialization: phase portrait

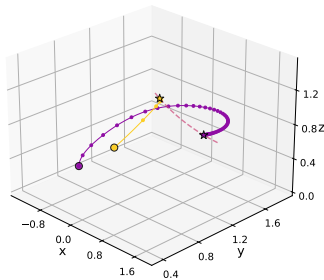
Log-Transformed Surface with Phase Portrait and Trajectories



## Toy example (I)



(a) Trajectory 1

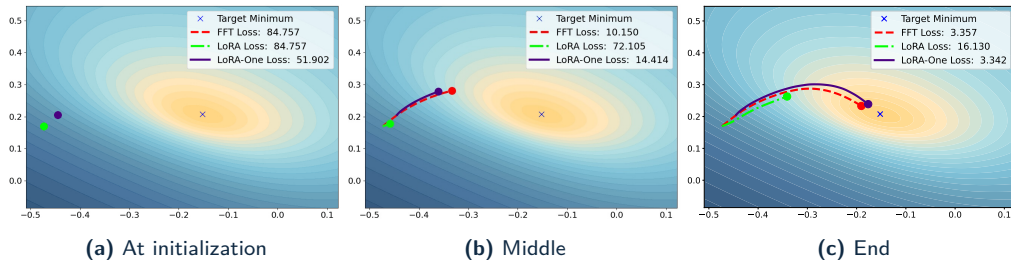


(b) Trajectory 2

**Figure 11:** Comparison of the GD trajectories between LoRA and ours. (a) and (b):  $A \in \mathbb{R}^2$  and  $B \in \mathbb{R}$  with different initializations. The set of global minimizers is  $\{a_1^* = 2/t, a_2^* = 1/t, b^* = t \mid t \in \mathbb{R}\}$ .



## Toy example (II)



**Figure 12:** Comparison of the GD trajectories between LoRA and ours. We use two-layer neural networks pre-trained on odd-labeled data and fine-tuned on even-labeled data on MNIST, see [GIF illustration](#).

Muon (no momentum, using NewtonSchulz5 instead of SVD)

$$\begin{aligned}[U, S, V] &= \text{SVD}(\nabla_{\mathbf{W}} L(\mathbf{W}_t)) \\ \mathbf{W}_{t+1} &= \mathbf{W}_t - \eta \mathbf{U} \mathbf{V}^\top\end{aligned}$$

### Muon (no momentum, using NewtonSchulz5 instead of SVD)

$$\begin{aligned}[U, S, V] &= \text{SVD}(\nabla_{\mathbf{W}} L(\mathbf{W}_t)) \\ \mathbf{W}_{t+1} &= \mathbf{W}_t - \eta \mathbf{U} \mathbf{V}^\top\end{aligned}$$

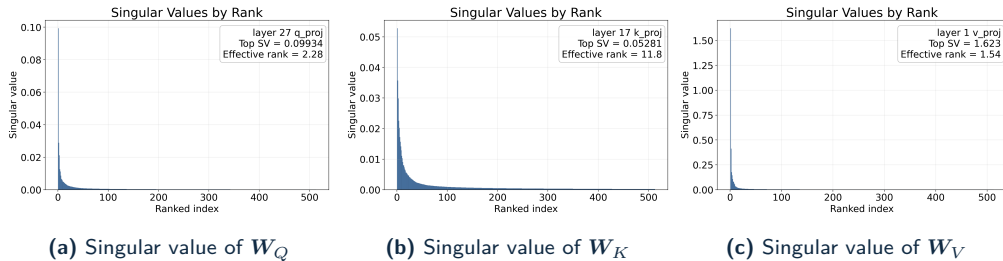
### One-step Muon for fine-tuning...

$$[U, S, V] = \text{SVD}(\underbrace{\nabla_{\mathbf{W}} L(\mathbf{W}^{\text{pre}})}_{:=G})$$

$$\text{Muon:} \quad \mathbf{W}_1 = \mathbf{W}^{\text{pre}} - \eta \mathbf{U} \mathbf{V}^\top$$

$$\text{LoRA-One:} \quad \mathbf{W}_1 = \mathbf{W}^{\text{pre}} - \eta \mathbf{U}_{[:,1:r]} \mathbf{S}_{[:,1:r]} \mathbf{V}_{[:,1:r]}^\top$$

# Spectral distribution



**Figure 13:** Spectral distribution of  $Q, K, V$ .

## One-step gradient can suffice on small-scale datasets!

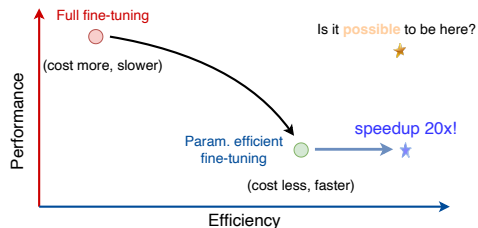
| Dataset<br>Size   | MNLI<br>393k     | SST-2<br>67k     | CoLA<br>8.5k     | QNLI<br>105k     | MRPC<br>3.7k     |
|-------------------|------------------|------------------|------------------|------------------|------------------|
| Pre-trained       | -                | 89.79            | 59.03            | 49.28            | 63.48            |
| One-step full     | -                | 90.94            | 69.13            | 70.35            | 68.38            |
| Muon-style        | -                | 90.48            | <b>73.00</b>     | 76.64            | <b>68.38</b>     |
| Spectral init.    | -                | 89.91            | <b>69.22</b>     | 76.31            | <b>68.38</b>     |
| LoRA <sub>8</sub> | 85.30 $\pm$ 0.04 | 94.04 $\pm$ 0.09 | 72.84 $\pm$ 1.25 | 93.02 $\pm$ 0.07 | 68.38 $\pm$ 0.01 |

# One-step gradient can suffice on small-scale datasets!

| Dataset Size      | MNLI<br>393k     | SST-2<br>67k     | CoLA<br>8.5k     | QNLI<br>105k     | MRPC<br>3.7k     |
|-------------------|------------------|------------------|------------------|------------------|------------------|
| Pre-trained       | -                | 89.79            | 59.03            | 49.28            | 63.48            |
| One-step full     | -                | 90.94            | 69.13            | 70.35            | 68.38            |
| Muon-style        | -                | 90.48            | <b>73.00</b>     | 76.64            | <b>68.38</b>     |
| Spectral init.    | -                | 89.91            | <b>69.22</b>     | 76.31            | <b>68.38</b>     |
| LoRA <sub>8</sub> | 85.30 $\pm$ 0.04 | 94.04 $\pm$ 0.09 | 72.84 $\pm$ 1.25 | 93.02 $\pm$ 0.07 | 68.38 $\pm$ 0.01 |

| Time cost (sec.) | LoRA | Spectral init. |
|------------------|------|----------------|
| CoLA             | 47s  | <1s            |
| MRPC             | 25s  | <1s            |

❑ memory-efficient (Lv et al., 2024) + rand. SVD



## Results on LLaMA 2-7B (for one epoch)

|             | GSM8K                   |                         | MMLU                    | HumanEval               |
|-------------|-------------------------|-------------------------|-------------------------|-------------------------|
| ( $r = 8$ ) | Direct                  | 8s-CoT                  | Avg.                    | PASS@1                  |
| LoRA        | 59.26 $\pm$ 0.76        | 53.36 $\pm$ 0.77        | 45.73 $\pm$ 0.30        | 25.85 $\pm$ 1.75        |
| LoRA-GA     | 56.44 $\pm$ 1.37        | 46.07 $\pm$ 1.01        | 45.70 $\pm$ 0.77        | 26.95 $\pm$ 1.30        |
| LoRA-One    | <b>60.44</b> $\pm$ 0.17 | <b>55.88</b> $\pm$ 0.44 | <b>47.12</b> $\pm$ 0.12 | <b>28.66</b> $\pm$ 0.39 |

- One epoch, rank 8, three runs
- Hyperparameter optimized over learning rate, batch size
- Train: 100k subset from MetaMathQA
- Test: GSM8K, Accuracy (%)

## Results on LLaMA 2-7B (for one epoch)

|             | GSM8K                   |                         | MMLU                    | HumanEval               |
|-------------|-------------------------|-------------------------|-------------------------|-------------------------|
| ( $r = 8$ ) | Direct                  | 8s-CoT                  | Avg.                    | PASS@1                  |
| LoRA        | 59.26 $\pm$ 0.76        | 53.36 $\pm$ 0.77        | 45.73 $\pm$ 0.30        | 25.85 $\pm$ 1.75        |
| LoRA-GA     | 56.44 $\pm$ 1.37        | 46.07 $\pm$ 1.01        | 45.70 $\pm$ 0.77        | 26.95 $\pm$ 1.30        |
| LoRA-One    | <b>60.44</b> $\pm$ 0.17 | <b>55.88</b> $\pm$ 0.44 | <b>47.12</b> $\pm$ 0.12 | <b>28.66</b> $\pm$ 0.39 |

- One epoch, rank 8, three runs
- Hyperparameter optimized over learning rate, batch size
- Train: 100k subset from MetaMathQA
- Test: GSM8K, Accuracy (%)

|           |                |         |
|-----------|----------------|---------|
| Time cost | LoRA: 6h 20min | + 3min  |
| Memory    | LoRA: 21.6 GB  | + 0.1GB |



## Results on LLaMA 2-7B (for one epoch)

|             | GSM8K                   |                         | MMLU                    | HumanEval               |
|-------------|-------------------------|-------------------------|-------------------------|-------------------------|
| ( $r = 8$ ) | Direct                  | 8s-CoT                  | Avg.                    | PASS@1                  |
| LoRA        | 59.26 $\pm$ 0.76        | 53.36 $\pm$ 0.77        | 45.73 $\pm$ 0.30        | 25.85 $\pm$ 1.75        |
| LoRA-GA     | 56.44 $\pm$ 1.37        | 46.07 $\pm$ 1.01        | 45.70 $\pm$ 0.77        | 26.95 $\pm$ 1.30        |
| LoRA-One    | <b>60.44</b> $\pm$ 0.17 | <b>55.88</b> $\pm$ 0.44 | <b>47.12</b> $\pm$ 0.12 | <b>28.66</b> $\pm$ 0.39 |

- One epoch, rank 8, three runs
- Hyperparameter optimized over learning rate, batch size
- Train: 100k subset from Code-Feedback
- Test: Humaneval, Pass@1

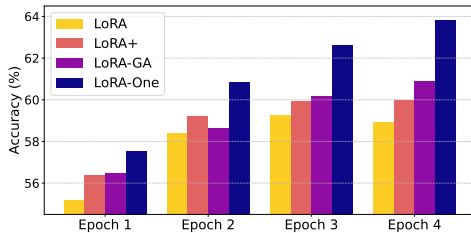
## Results on LLaMA 2-7B (for one epoch)

|             | GSM8K                   |                         | MMLU                    | HumanEval               |
|-------------|-------------------------|-------------------------|-------------------------|-------------------------|
| ( $r = 8$ ) | Direct                  | 8s-CoT                  | Avg.                    | PASS@1                  |
| LoRA        | 59.26 $\pm$ 0.76        | 53.36 $\pm$ 0.77        | 45.73 $\pm$ 0.30        | 25.85 $\pm$ 1.75        |
| LoRA-GA     | 56.44 $\pm$ 1.37        | 46.07 $\pm$ 1.01        | 45.70 $\pm$ 0.77        | 26.95 $\pm$ 1.30        |
| LoRA-One    | <b>60.44</b> $\pm$ 0.17 | <b>55.88</b> $\pm$ 0.44 | <b>47.12</b> $\pm$ 0.12 | <b>28.66</b> $\pm$ 0.39 |

- One epoch, rank 8, three runs
- Hyperparameter optimized over learning rate, batch size
- Train: 100k subset from Code-Feedback
- Test: Humaneval, Pass@1

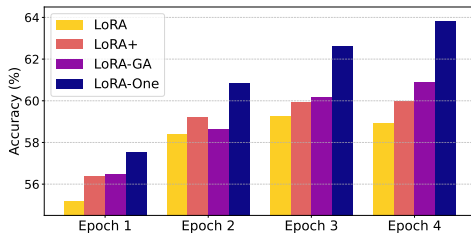
|           |                |         |
|-----------|----------------|---------|
| Time cost | LoRA: 6h 24min | + 2min  |
| Memory    | LoRA: 22.6 GB  | - 1.1GB |

## Results on LLaMA 2-7B (for more epochs)

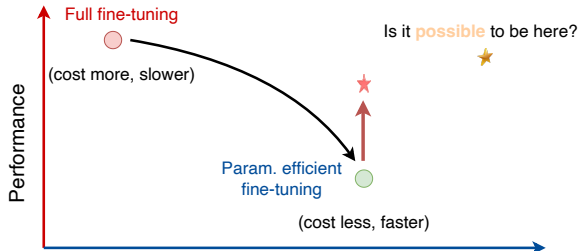


**Figure 14:** Accuracy comparison across different methods over epochs on GSM8K.

## Results on LLaMA 2-7B (for more epochs)



**Figure 14:** Accuracy comparison across different methods over epochs on GSM8K.



## Clarification on gradient alignment based work

---

LoRA-GA ([Wang et al., 2024](#)): make LoRA's gradients align to full fine-tuning!

## Clarification on gradient alignment based work

---

LoRA-GA (Wang et al., 2024): make LoRA's gradients align to full fine-tuning!

□ best  $2r$  approximation

$$\begin{cases} \text{rank}(\nabla_{\mathbf{A}} L(\mathbf{A}_t, \mathbf{B}_t)) \leq r \\ \text{rank}(\nabla_{\mathbf{B}} L(\mathbf{A}_t, \mathbf{B}_t)) \leq r \end{cases}$$

## Clarification on gradient alignment based work

LoRA-GA (Wang et al., 2024): make LoRA's gradients align to full fine-tuning!

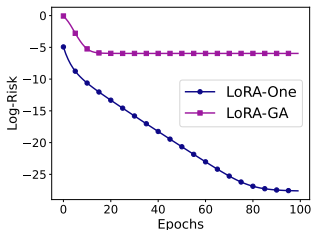
□ best  $2r$  approximation

$$\begin{cases} \text{rank}(\nabla_A L(\mathbf{A}_t, \mathbf{B}_t)) \leq r \\ \text{rank}(\nabla_B L(\mathbf{A}_t, \mathbf{B}_t)) \leq r \end{cases}$$

| Method   | Init. on $A$                  | Init. on $B$                       | Calibration                |
|----------|-------------------------------|------------------------------------|----------------------------|
| LoRA     | $\mathcal{N}(0, \alpha^2)$    | $\mathbf{0}$                       | -                          |
| LoRA-GA  | $U_{[:,1:r]}$                 | $V_{[:,r+1:2r]}^\top$              | $W^{\text{pre}} - A_0 B_0$ |
| LoRA-One | $U_{[:,1:r]} S_{[1:r]}^{1/2}$ | $S_{[1:r]}^{1/2} V_{[:,1:r]}^\top$ | -                          |

# Clarification on gradient alignment based work

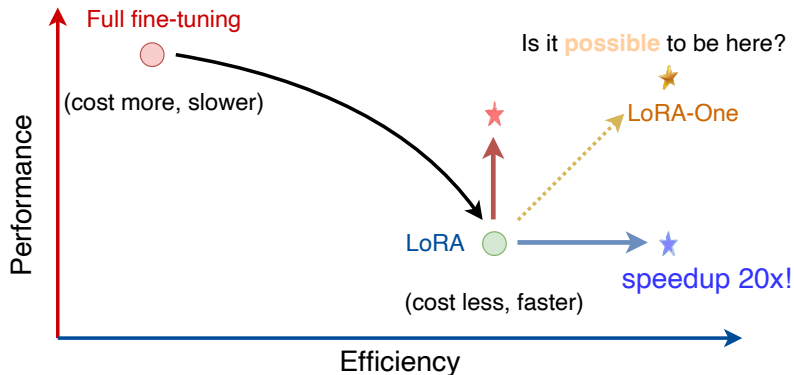
LoRA-GA (Wang et al., 2024): make LoRA's gradients align to full fine-tuning!



| Method   | Init. on $A$                  | Init. on $B$                       | Calibration                |
|----------|-------------------------------|------------------------------------|----------------------------|
| LoRA     | $\mathcal{N}(0, \alpha^2)$    | $\mathbf{0}$                       | -                          |
| LoRA-GA  | $U_{[:,1:r]}$                 | $V_{[:,r+1:2r]}^\top$              | $W^{\text{pre}} - A_0 B_0$ |
| LoRA-One | $U_{[:,1:r]} S_{[1:r]}^{1/2}$ | $S_{[1:r]}^{1/2} V_{[:,1:r]}^\top$ | -                          |



## Takeaway messages: speedup via spectral initialization



□ Yuanhe Zhang, Fanghui Liu, Yudong Chen. *LoRA-One: One-step full gradient could suffice for fine-tuning large language models, provably and efficiently.* ICML'25 Oral

- **subspace alignment:**  $G$  and  $(A_t, B_t) \Rightarrow$  theory-grounded algorithm design
- “optimal” non-zero initialization strategy

# Go beyond...

---

## Target

- How to handle **nonlinearity** at a theoretical level (e.g., training dynamics)
- How to precisely and efficiently approximate **nonlinearity** at a practical level under theoretical guidelines

# Go beyond...

## Target

- How to handle **nonlinearity** at a theoretical level (e.g., training dynamics)
- How to precisely and efficiently approximate **nonlinearity** at a practical level under theoretical guidelines

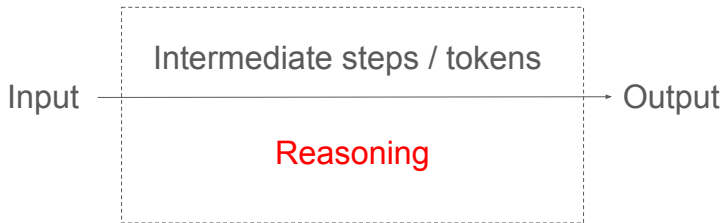


(Generated by GPT5.5)

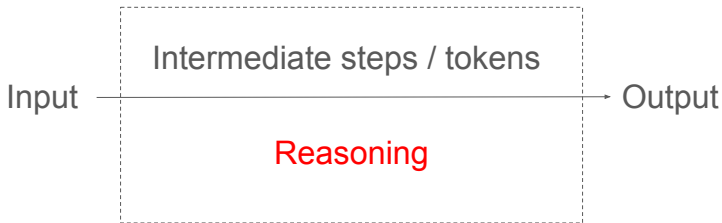
# DAG-MATH

---

# What is LLM reasoning/Chain-of-Thought (CoT) (Wei et al., 2022)?



# What is LLM reasoning/Chain-of-Thought (CoT) (Wei et al., 2022)?



What is the output when concatenating the last letter of each word in “artificial intelligence”?

## No reasoning

The answer is “le”.

## Reasoning

The last letter of “artificial” is “l”. The last letter of “intelligence” is “e”. Concatenating “l” and “e” leads to “le”. So the answer is “le”.



## Motivation: Is CoT real reasoning

---

LLMs achieve high scores on math benchmarks using Chain-of-Thought (CoT). **But is this success from:**

- Genuine logical inference?
- Search / exploration?
- Memorization?

# Motivation: Is CoT real reasoning

LLMs achieve high scores on math benchmarks using Chain-of-Thought (CoT). **But is this success from:**

- Genuine logical inference?
- Search / exploration?
- Memorization?

## Existing Gaps

- **Framework:** Prior work uses fixed graphs (Dziri et al., 2023), reversible chains (Kim et al., 2025), or k-parity trees (Yao et al., 2023) — missing generality.
- **Evaluation:** PASS@k only checks final answers, but not reasoning fidelity.



# Motivation: Is CoT real reasoning

LLMs achieve high scores on math benchmarks using Chain-of-Thought (CoT). **But is this success from:**

- Genuine logical inference?
- Search / exploration?
- Memorization?

## Existing Gaps

- **Framework:** Prior work uses fixed graphs (Dziri et al., 2023), reversible chains (Kim et al., 2025), or k-parity trees (Yao et al., 2023) — missing generality.
- **Evaluation:** PASS@k only checks final answers, but not reasoning fidelity.

Intuitively, LLM reasoning requires

- identify all required premises (e.g., facts, constraints)
- correct **logical inference** from premises to the conclusion
- Accurate calculation and symbolic execution

# Mathematical reasoning of LLMs (Natural language)

## Logarithmic Count Problem from MATH-500

For how many integers  $k \in [-300, 300]$  does the equation  $2\log(x - 1) = \log k$  have exactly one real solution  $x$ ?

# Mathematical reasoning of LLMs (Natural language)

## Logarithmic Count Problem from MATH-500

For how many integers  $k \in [-300, 300]$  does the equation  $2\log(x-1) = \log k$  have exactly one real solution  $x$ ?

ChatGPT 5.2 ▾

↑ Share

For real logs we need

- $x - 1 > 0 \Rightarrow x > 1$ ,
- $k > 0$  (since  $\log k$  must be defined).

Given

$$2\log(x-1) = \log k \implies \log((x-1)^2) = \log k$$

and since  $\log$  is injective on  $(0, \infty)$ ,

$$(x-1)^2 = k.$$

Thus  $x = 1 \pm \sqrt{k}$ . The constraint  $x > 1$  eliminates  $1 - \sqrt{k}$ , leaving exactly one real solution  $x = 1 + \sqrt{k}$  for every  $k > 0$ .

In  $[-300, 300]$ , the positive integers are  $1, 2, \dots, 300$ : **300 values**.

# Lean Language (Interact with Claude)

```
12 Analysis:
13 - For  $\log(x-1)$  to be defined, we need  $x > 1$ 
14 - For  $\log k$  to be defined (as a real), we need  $k > 0$ 
15 - The equation  $2\log(x-1) = \log k$  becomes  $\log((x-1)^2) = \log k$ 
16 - This gives  $(x-1)^2 = k$ , so  $x - 1 = \sqrt{k}$ 
17 - Since  $x > 1$ , we need  $x - 1 > 0$ , so  $x - 1 = \sqrt{k}$  (the positive root)
18 - Thus  $x = 1 + \sqrt{k}$  is the unique solution for each  $k > 0$ 
19
20 Answer: 300 (the positive integers from 1 to 300)
21 -/
22
23 /-- The set of real solutions  $x$  to the equation  $2 * \log(x - 1) = \log k$  -/
24 def solutions (k : ℝ) : Set ℝ :=
25   {x : ℝ | x > 1 ∧ k > 0 ∧ 2 * Real.log (x - 1) = Real.log k}
26
27 /-- Helper lemma:  $2 * \log(\sqrt{k}) = \log k$  for  $k > 0$  -/
28 lemma two_log_sqrt_eq (k : ℝ) (hk : k > 0) : 2 * Real.log (Real.sqrt k) = Real.log k := by
29   rw [Real.log_sqrt (le_of_lt hk)]
30   ring
31
32 /-- For  $k > 0$ , the equation  $2 * \log(x - 1) = \log k$  has exactly one solution:  $x = 1 + \sqrt{k}$  -/
33 theorem solutions_eq_singleton (k : ℝ) (hk : k > 0) :
34   solutions k = {1 + Real.sqrt k} := by
35   ext x
36   simp only [solutions, mem_setOf_eq, mem_singleton_iff]
37   constructor
38   · intro (hwl, _, heq)
39   · have hwm_pos : x - 1 > 0 := by linarith
40   · have h1 : Real.log ((x - 1) ^ 2) = Real.log k := by
41     rw [Real.log_pow, ← heq]; ring
42   · have hsq_pos : (x - 1) ^ 2 > 0 := sq_pos_of_pos hwm_pos
43   · have h2 : (x - 1) ^ 2 = k := Real.log_injOn_pos (mem_Ioi.mpr hsq_pos) (mem_Ioi.mpr hk) h1
44   · have h3 : x - 1 = Real.sqrt k := by
45     have hsqrt : Real.sqrt ((x - 1) ^ 2) = Real.sqrt k := congrArg Real.sqrt h2
46     rw [Real.sqrt_sq (le_of_lt hwm_pos)] at hsqrt
47   · linarith
48   · intro hx
49   · refine (?, hk, ?_)
50   · rw [hx]
51   · have : Real.sqrt k > 0 := Real.sqrt_pos.mpr hk
52   · linarith
53   · rw [hx]
54   · simp only [add_sub_cancel_left]
55   · exact two_log_sqrt_eq k hk
56
57 /-- For  $k \leq 0$ , the equation has no solutions -/
58 theorem solutions_empty_of_nonpos (k : ℝ) (hk : k ≤ 0) :
59   solutions k = ∅ := by
60   ext x
61   simp only [solutions, mem_setOf_eq, mem_empty_iff_false, iff_false, not_and]
62   intro _ hk_pos
63   linarith
```

```
65 /-- The equation has exactly one solution if and only if  $k > 0$  -/
66 theorem has_unique_solution_iff (k : ℝ) :
67   (∃! x, x ∈ solutions k) = k > 0 := by
68   constructor
69   · intro (x, hx, _)
70   · exact hx.2.1
71   · intro hk
72   · rw [solutions_eq_singleton k hk]
73   · use 1 + Real.sqrt k
74   · simp only [mem_singleton_iff, true_and]
75   · exact fun _ => id
76
77 /-- The set of integers  $k$  in  $[-300, 300]$  for which the equation has exactly one solution -/
78 def validIntegers : Finset ℤ :=
79   Finset.filter (fun k => (0 : ℤ) < k) (Finset.Icc (-300) 300)
80
81 /-- The valid integers are exactly  $\{1, 2, \dots, 300\}$  -/
82 theorem validIntegers_eq : validIntegers = Finset.Icc 1 300 := by
83   ext k
84   simp only [validIntegers, Finset.mem_filter, Finset.mem_Icc]
85   constructor
86   · intro (⟦_, hhi⟧, hpos)
87   · exact (hpos, hhi)
88   · intro (hpos, hhi)
89   · exact ((by linarith, hhi), hpos)
90
91 /-- The cardinality of  $\text{Icc } 1 \text{ } 300$  is 300 -/
92 theorem card_Icc_one_300 : (Finset.Icc (1 : ℤ) 300).card = 300 := by
93   have h : (Finset.Icc (1 : ℤ) 300).card = (300 - 1 + 1 : ℤ).toNat := Int.card_Icc 1 300
94   simp only [h]
95   native_decide
96
97 /-- The answer to the problem is 300 -/
98 theorem answer : validIntegers.card = 300 := by
99   rw [validIntegers_eq, card_Icc_one_300]
100
101 /-- Characterization:  $k \in \text{validIntegers}$  iff  $k$  is in range and the equation has unique solution -/
102 theorem validIntegers_iff (k : ℤ) :
103   k ∈ validIntegers = k ∈ Finset.Icc (-300 : ℤ) 300 ∧ (∃! x : ℝ, x ∈ solutions k) := by
104   simp only [validIntegers, Finset.mem_filter, Finset.mem_Icc]
105   constructor
106   · intro (hlo, hhi), hpos
107   · refine (hlo, hhi, ?_)
108   · rw [has_unique_solution_iff]
109   · exact Int.cast_pos.mpr hpos
110   · intro (hlo, hhi), huniq
111   · refine (hlo, hhi, ?_)
112   · rw [has_unique_solution_iff] at huniq
113   · exact Int.cast_pos.mpr huniq
114
```

## Phase 1: Task-specific Direct Acyclic Graph (DAG) for Step-Level CoT

One CoT Step in the Logarithmic Count Problem

If two logarithmic expressions are equal, then their arguments is equal.

Hence, from  $2 \log(x - 1) = \log k$ , we can conclude  $(x - 1)^2 = k$ .

## Phase 1: Task-specific Direct Acyclic Graph (DAG) for Step-Level CoT

### One CoT Step in the Logarithmic Count Problem

If two logarithmic expressions are equal, then their arguments is equal.

Hence, from  $2 \log(x - 1) = \log k$ , we can conclude  $(x - 1)^2 = k$ .

- Edge (Justification): encodes the logical dependency, making the reasoning chain transparent.
- Node (Conclusion): the step's conclusion (state or value)

## Phase 1: Task-specific Direct Acyclic Graph (DAG) for Step-Level CoT

### One CoT Step in the Logarithmic Count Problem

If two logarithmic expressions are equal, then their arguments is equal.

Hence, from  $2 \log(x - 1) = \log k$ , we can conclude  $(x - 1)^2 = k$ .

- Edge (Justification): encodes the logical dependency, making the reasoning chain transparent.
- Node (Conclusion): the step's conclusion (state or value)

Decomposition is *non-unique* due to semantic variation.

## Phase 1: Task-specific Direct Acyclic Graph (DAG) for Step-Level CoT

### One CoT Step in the Logarithmic Count Problem

If two logarithmic expressions are equal, then their arguments is equal.

Hence, from  $2 \log(x - 1) = \log k$ , we can conclude  $(x - 1)^2 = k$ .

- Edge (Justification): encodes the logical dependency, making the reasoning chain transparent.
- Node (Conclusion): the step's conclusion (state or value)

Given a prompt  $x_{\text{in}}$ , we define the directed graph as

$$\mathcal{G}(x_{\text{in}}) := (\mathcal{V}(x_{\text{in}}), \mathcal{E}(x_{\text{in}})), \quad \text{where } \mathcal{E}(x_{\text{in}}) \subseteq \mathcal{V}(x_{\text{in}}) \times \mathcal{V}(x_{\text{in}}).$$



# Phase 1: Task-specific Direct Acyclic Graph (DAG) for Step-Level CoT

## One CoT Step in the Logarithmic Count Problem

If two logarithmic expressions are equal, then their arguments is equal.

Hence, from  $2 \log(x - 1) = \log k$ , we can conclude  $(x - 1)^2 = k$ .

- Edge (Justification): encodes the logical dependency, making the reasoning chain transparent.
- Node (Conclusion): the step's conclusion (state or value)

Given a prompt  $x_{\text{in}}$ , we define the directed graph as

$$\mathcal{G}(x_{\text{in}}) := (\mathcal{V}(x_{\text{in}}), \mathcal{E}(x_{\text{in}})), \quad \text{where } \mathcal{E}(x_{\text{in}}) \subseteq \mathcal{V}(x_{\text{in}}) \times \mathcal{V}(x_{\text{in}}).$$

**Empirical evidence (Ye et al., 2025):** A latent directed dependency graph: once a question/prompt is posted, before any output is generated.

# Why acyclic?

## Assumption

*For any input prompt  $x_{\text{in}}$ , the task-specific directed graph  $\mathcal{G}(x_{\text{in}})$  is acyclic.*

# Why acyclic?

## Assumption

*For any input prompt  $x_{\text{in}}$ , the task-specific directed graph  $\mathcal{G}(x_{\text{in}})$  is acyclic.*

- Thinking models encourage backtracking and self-correction during the thinking process

# Why acyclic?

## Assumption

For any input prompt  $x_{\text{in}}$ , the task-specific directed graph  $\mathcal{G}(x_{\text{in}})$  is acyclic.

- Thinking models encourage backtracking and self-correction during the thinking process

ChatGPT 5.2 Thinking ▾

↑ Share

👤 Add people

...

Activity · 1m 49s

×

On  $\triangle ABC$ , points  $A, D, E$ , and  $B$  lie in that order on side  $AB$  with  $AD = 4$ ,  $DE = 16$ ,  $EB = 8$ . Points  $A, F, G$  and  $C$  lie in that order on side  $AC$  with  $AF = 13$ ,  $FG = 32$ , and  $GC = 26$ . Let  $M$  be the reflection of  $D$  through  $F$ , and let  $N$  be the reflection of  $G$  through  $E$ . Quadrilateral  $DEGF$  has area 288. Find the area of heptagon  $AFNBCEM$ .

sovlve this problem

Thought for 1m 49s ▸

Let  $\vec{u} = \overrightarrow{AB}$  and  $\vec{v} = \overrightarrow{AC}$ . Use the affine coordinate system

$$(x, y) \mapsto x\vec{u} + y\vec{v},$$

so that

$$A = (0, 0), \quad B = (1, 0), \quad C = (0, 1).$$

- Checking shape of heptagon  $AFNBCEM$  for self-intersection

Seems like the area I calculated for the heptagon is 588, which matches the triangle's area. But I need to check if  $AFNBCEM$  is a simple polygon (without self-intersections). I'm looking at the points carefully to see if they cross each other. I suspect it's not self-crossing, but I'll double-check for intersections between segments like  $F-N$  and  $C-E$ .

- Checking intersections between edges of the polygon

I'm looking into possible intersections between the polygon's edges. So far, I've checked the slopes and equations for several edge pairs. For example, edge  $F-N$  doesn't intersect with  $C-E$ . Edge  $M-A$  also doesn't cross with  $F-N$ . But I need to check more pairs like  $N-B$  and  $E-M$  to be sure.

# Why acyclic?

## Assumption

For any input prompt  $x_{\text{in}}$ , the task-specific directed graph  $\mathcal{G}(x_{\text{in}})$  is acyclic.

LLMs output only the finalized, perfect, correct reasoning results to the user

ChatGPT 5.2 Thinking ▾

↑ Share

👤 Add people

...

Activity · 1m 49s

×

On  $\triangle ABC$ , points  $A, D, E$ , and  $B$  lie in that order on side  $AB$  with  $AD = 4$ ,  $DE = 16$ ,  $EB = 8$ . Points  $A, F, G$  and  $C$  lie in that order on side  $AC$  with  $AF = 13$ ,  $FG = 52$ , and  $GC = 26$ . Let  $M$  be the reflection of  $D$  through  $F$ , and let  $N$  be the reflection of  $G$  through  $E$ . Quadrilateral  $DEGF$  has area 288. Find the area of heptagon  $AFNBCEM$ .

solve this problem

Thought for 1m 49s >

Let  $\vec{u} = \overrightarrow{AB}$  and  $\vec{v} = \overrightarrow{AC}$ . Use the affine coordinate system

$$(x, y) \mapsto x\vec{u} + y\vec{v},$$

so that

$$A = (0, 0), \quad B = (1, 0), \quad C = (0, 1).$$

- Checking shape of heptagon  $AFNBCEM$  for self-intersection

Seems like the area I calculated for the heptagon is 588, which matches the triangle's area. But I need to check if  $AFNBCEM$  is a simple polygon (without self-intersections). I'm looking at the points carefully to see if they cross each other. I suspect it's not self-crossing, but I'll double-check for intersections between segments like  $F-N$  and  $C-E$ .

- Checking intersections between edges of the polygon

I'm looking into possible intersections between the polygon's edges. So far, I've checked the slopes and equations for several edge pairs. For example, edge  $F-N$  doesn't intersect with  $C-E$ . Edge  $M-A$  also doesn't cross with  $F-N$ . But I need to check more pairs like  $N-B$  and  $E-M$  to be sure.

# Why acyclic?

## Assumption

For any input prompt  $x_{\text{in}}$ , the task-specific directed graph  $\mathcal{G}(x_{\text{in}})$  is acyclic.

LLMs output only the finalized, perfect, correct reasoning results to the user

answer-based math problem: constructed by backtracking through its ancestor

ChatGPT 5.2 Thinking ▾

↑ Share

👤 Add people

...

Activity · 1m 49s

×

On  $\triangle ABC$ , points  $A, D, E$ , and  $B$  lie in that order on side  $AB$  with  $AD = 4$ ,  $DE = 16$ ,  $EB = 8$ . Points  $A, F, G$  and  $C$  lie in that order on side  $AC$  with  $AF = 13$ ,  $FG = 32$ , and  $GC = 20$ . Let  $M$  be the reflection of  $D$  through  $F$ , and let  $N$  be the reflection of  $G$  through  $E$ . Quadrilateral  $DEGF$  has area 288. Find the area of heptagon  $AFNBCEM$ .

solve this problem

Thought for 1m 49s ▸

Let  $\vec{u} = \overrightarrow{AB}$  and  $\vec{v} = \overrightarrow{AC}$ . Use the affine coordinate system

$$(x, y) \mapsto x\vec{u} + y\vec{v},$$

so that

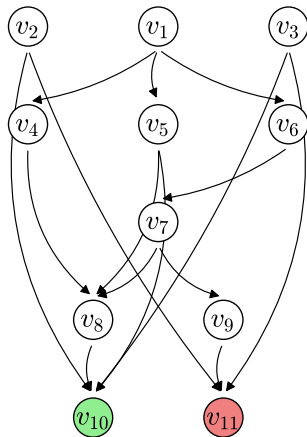
- Checking shape of heptagon  $AFNBCEM$  for self-intersection

Seems like the area I calculated for the heptagon is 588, which matches the triangle's area. But I need to check if  $AFNBCEM$  is a simple polygon (without self-intersections). I'm looking at the points carefully to see if they cross each other. I suspect it's not self-crossing, but I'll double-check for intersections between segments like  $F-N$  and  $C-E$ .

- Checking intersections between edges of the polygon

I'm looking into possible intersections between the polygon's edges. So far, I've checked the slopes and equations for several edge pairs. For

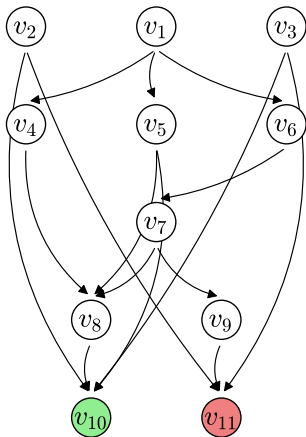
## Example: Task specific DAG for the logarithmic count problem



- $v_1$  : “ $2 \log(x - 1) = \log k$ ” (target equation);
- $v_2$  : “ $k \in [-300, 300]$ ” (range constraint);
- $v_3$  : “exactly one solution” (task requirement);
- $v_4$  : “ $\log(x - 1)$  requires  $x > 1$ ” (constraint inferred from the equation);
- $v_5$  : “ $\log k$  requires  $k > 0$ ” (constraint inferred from the equation);
- $v_6$  : “ $2 \log(x - 1) = \log k \Rightarrow (x - 1)^2 = k$ ” (re-arranged equation);
- $v_7$  : “ $x = 1 \pm \sqrt{k}$ ” (solve the quadratic equation);
- $v_8$  : “ $1 + \sqrt{k}$  is the only solution” (correct check);
- $v_9$  : “For any  $k$ , there are two solutions” (incorrect check);
- $v_{10}$  : “There are 300 valid values for  $k$ ” (the **correct** answer);
- $v_{11}$  : “There are 0 valid value for  $k$ ” (the **incorrect** answer).

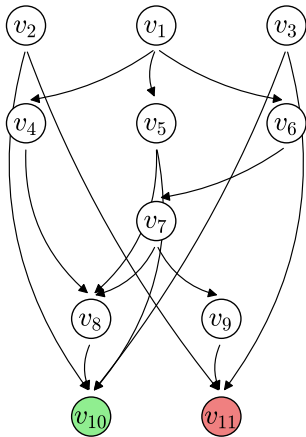
**Figure 15:** Task-specific DAG via LLaMA-3.1-8B-Instruct.

## Phase 2: Stochastic Process on Logic Dependence



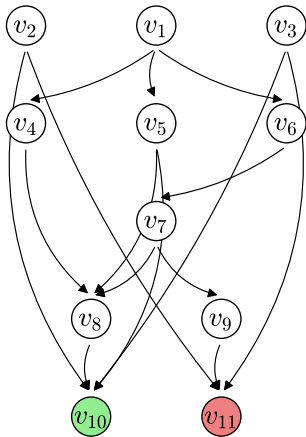


## Phase 2: Stochastic Process on Logic Dependence



- Node-level autoregressive distribution  $\mathbb{P}$
- The CoT trajectory  $\{V_i\}_{i=1}^L$  follows:
  - $V_1 \sim \mathbb{P}(\cdot \mid \mathbf{X}_{\text{in}})$
  - $V_t \sim \mathbb{P}(\cdot \mid V_{t-1}, \dots, V_1, \mathbf{X}_{\text{in}})$
  - $V_{\text{out}} \sim \mathbb{P}(\cdot \mid V_{L-1}, \dots, V_1, \mathbf{X}_{\text{in}})$

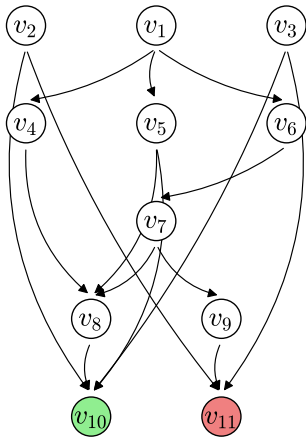
## Phase 2: Stochastic Process on Logic Dependence



- Node-level autoregressive distribution  $\mathbb{P}$
- The CoT trajectory  $\{V_i\}_{i=1}^L$  follows:
  - $V_1 \sim \mathbb{P}(\cdot \mid \mathbf{X}_{\text{in}})$
  - $V_t \sim \mathbb{P}(\cdot \mid V_{t-1}, \dots, V_1, \mathbf{X}_{\text{in}})$
  - $V_{\text{out}} \sim \mathbb{P}(\cdot \mid V_{L-1}, \dots, V_1, \mathbf{X}_{\text{in}})$
- (Valid) transition probability  
 $\mathbb{P}(V_t = v \mid V_{1:t-1} = v_{1:t-1}, \mathbf{X}_{\text{in}} = \mathbf{x}_{\text{in}}),$   
 $\forall v \in \mathcal{V}(v_{1:t-1} \mid \mathbf{x}_{\text{in}})$

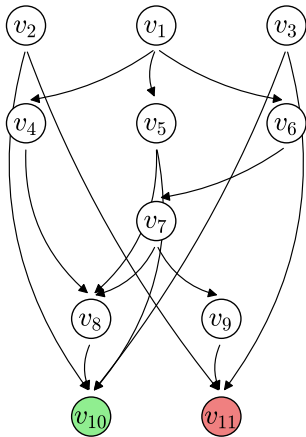
$$\mathcal{V}(v_{1:t-1} \mid \mathbf{x}_{\text{in}}) := \left\{ v \in \mathcal{V}(\mathbf{x}_{\text{in}}) : \text{pa}(v) \subseteq \{v_{1:t-1}\}, v \notin \{v_{1:t-1}\} \right\}.$$

## Example: Task specific DAG for the logarithmic count problem



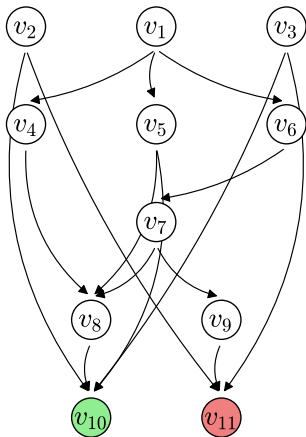
- **Wrong reasoning**, e.g.,  $(v_1, v_2, v_3, v_6, v_7, v_9, v_{11})$ : The final answer is incorrect.

## Example: Task specific DAG for the logarithmic count problem



- **Wrong reasoning**, e.g.,  $(v_1, v_2, v_3, v_6, v_7, v_9, v_{11})$ : The final answer is incorrect.
- **Perfect reasoning**  $(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_{10})$ : The trajectory only includes the correct sink node and its ancestors.

## Example: Task specific DAG for the logarithmic count problem



- **Wrong reasoning**, e.g.,  $(v_1, v_2, v_3, v_6, v_7, v_9, v_{11})$ : The final answer is incorrect.
- **Perfect reasoning**  $(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_{10})$ : The trajectory only includes the correct sink node and its ancestors.
- **Imperfect reasoning** e.g.,  $(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_9, v_8, v_{10})$ : The trajectory still reaches the correct answer but also includes the irrelevant node  $v_9$ , which is not an ancestor of the correct node.

### Key Insight

**PASS@k treats imperfect and perfect reasoning identically.** New metric needed to distinguish them: search vs. genuine inference.

## Logical Closeness & Perfect Reasoning

A generated DAG is **logically closed** if every non-sink node has at least one outgoing edge, meaning every intermediate result is actually used in subsequent reasoning.

**Perfect reasoning** = logically closed + correct final answer.

# Evaluation Metrics

## Logical Closeness & Perfect Reasoning

A generated DAG is **logically closed** if every non-sink node has at least one outgoing edge, meaning every intermediate result is actually used in subsequent reasoning.

**Perfect reasoning** = logically closed + correct final answer.

## Perfect Reasoning Rate (PRR)

Fraction of trajectories that are both logically closed AND reach the correct answer.

Captures genuine reasoning ability.

## AUC Score

Relaxes the closeness threshold from 0% to 100%, plotting accuracy at each level.

Provides a comprehensive ranking of reasoning performance.

## Formulate mathematical reasoning ability

Given an input prompt  $x_{\text{in}}$ , generate one CoT trajectory, we construct a trajectory-specific DAG  $\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})$

### Definition (Logical closeness)

For each node  $v \in \mathcal{G}_{\text{gen}}(\mathbf{x}_{\text{in}})$ , define its out-degree as

$$\text{deg}(v \mid \mathcal{G}_{\text{gen}}(\mathbf{x}_{\text{in}})) := \left| \{u \in \mathcal{G}_{\text{gen}}(\mathbf{x}_{\text{in}}) \mid (v \rightarrow u) \in \mathcal{E}_{\text{gen}}(\mathbf{x}_{\text{in}})\} \right|.$$



## Formulate mathematical reasoning ability

Given an input prompt  $x_{\text{in}}$ , generate one CoT trajectory, we construct a trajectory-specific DAG  $\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})$

### Definition (Logical closeness)

For each node  $v \in \mathcal{G}_{\text{gen}}(\mathbf{x}_{\text{in}})$ , define its out-degree as

$$\deg(v \mid \mathcal{G}_{\text{gen}}(\mathbf{x}_{\text{in}})) := \left| \{u \in \mathcal{G}_{\text{gen}}(\mathbf{x}_{\text{in}}) \mid (v \rightarrow u) \in \mathcal{E}_{\text{gen}}(\mathbf{x}_{\text{in}})\} \right|.$$

We say that  $\mathcal{G}_{\text{gen}}(\mathbf{x}_{\text{in}})$  is **logically closed** if

$$\deg(v \mid \mathcal{G}_{\text{gen}}(\mathbf{x}_{\text{in}})) \geq 1, \quad \forall v \in \mathcal{V}_{\text{gen}}(\mathbf{x}_{\text{in}}),$$

i.e., only the final nodes have no outgoing edges.

# Mathematical reasoning ability of LLMs

## Indicator functions

$$\delta_{\text{final}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) := \begin{cases} 1, & \text{if the sink node of } \mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}}) \text{ is correct,} \\ 0, & \text{otherwise.} \end{cases}$$

$$\delta_{\text{close}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) := \begin{cases} 1, & \text{if } \mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}}) \text{ is logically closed,} \\ 0, & \text{otherwise.} \end{cases}$$

# Mathematical reasoning ability of LLMs

## Indicator functions

$$\delta_{\text{final}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) := \begin{cases} 1, & \text{if the sink node of } \mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}}) \text{ is correct,} \\ 0, & \text{otherwise.} \end{cases}$$

$$\delta_{\text{close}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) := \begin{cases} 1, & \text{if } \mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}}) \text{ is logically closed,} \\ 0, & \text{otherwise.} \end{cases}$$

The **Perfect Reasoning Rate (PRR)** is

$$\text{PRR}(\mathbf{X}_{\text{in}}) := \mathbb{E}_{\mathbb{P}} \left[ \delta_{\text{close}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) \times \delta_{\text{final}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) \right].$$

# Mathematical reasoning ability of LLMs

## Indicator functions

$$\delta_{\text{final}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) := \begin{cases} 1, & \text{if the sink node of } \mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}}) \text{ is correct,} \\ 0, & \text{otherwise.} \end{cases}$$

$$\delta_{\text{close}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) := \begin{cases} 1, & \text{if } \mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}}) \text{ is logically closed,} \\ 0, & \text{otherwise.} \end{cases}$$

The **Perfect Reasoning Rate (PRR)** is

$$\text{PRR}(\mathbf{X}_{\text{in}}) := \mathbb{E}_{\mathbb{P}} \left[ \delta_{\text{close}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) \times \delta_{\text{final}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) \right].$$

## Definition (Mathematical reasoning ability)

The overall **mathematical reasoning ability** of an LLM is

$$\mathcal{R} := \mathbb{E}_{\mathbf{X}_{\text{in}} \sim \mathcal{D}} [\text{PRR}(\mathbf{X}_{\text{in}})] = \mathbb{E}_{\mathbb{P}, \mathbf{X}_{\text{in}} \sim \mathcal{D}} \left[ \delta_{\text{close}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) \times \delta_{\text{final}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) \right].$$

# Mathematical reasoning ability of LLMs

## Indicator functions

$$\delta_{\text{final}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) := \begin{cases} 1, & \text{if the sink node of } \mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}}) \text{ is correct,} \\ 0, & \text{otherwise.} \end{cases}$$

$$s_{\text{close}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) = \alpha, \text{ if ratio } \alpha \text{ of nodes in } \mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}}) \text{ is logically closed}$$

The **Perfect Reasoning Rate (PRR)** is

$$\text{PRR}(\mathbf{X}_{\text{in}}) := \mathbb{E}_{\mathbb{P}} \left[ \delta_{\text{close}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) \times \delta_{\text{final}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) \right].$$

## Definition (Mathematical reasoning ability)

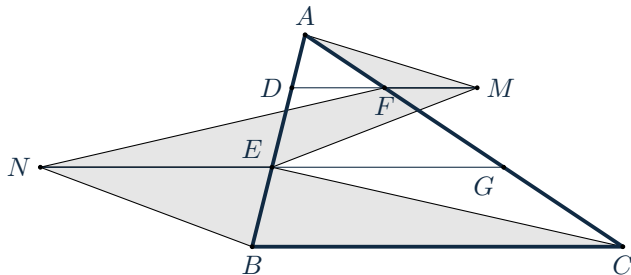
The overall **mathematical reasoning ability** of an LLM is

$$\mathcal{R} := \mathbb{E}_{\mathbf{X}_{\text{in}} \sim \mathcal{D}} [\text{PRR}(\mathbf{X}_{\text{in}})] = \mathbb{E}_{\mathbb{P}, \mathbf{X}_{\text{in}} \sim \mathcal{D}} \left[ \delta_{\text{close}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) \times \delta_{\text{final}}(\mathcal{G}_{\text{gen}}(\mathbf{X}_{\text{in}})) \right].$$

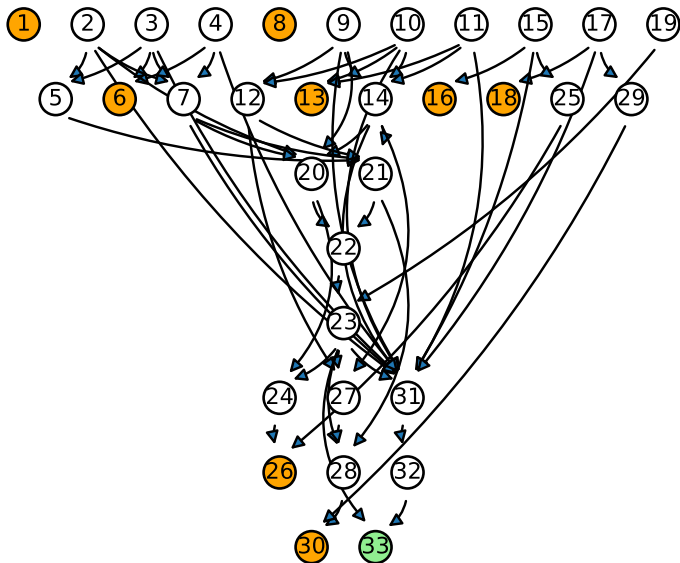
## Why logic closeness useful? - Diagnosis

### Area of Heptagon Problem

On  $\triangle ABC$ , points  $D, E$ , and  $B$  lie in that order on side  $\overline{AB}$  with  $AD = 4$ ,  $DE = 16$ ,  $EB = 8$ . Points  $F, G$  and  $C$  lie in that order on side  $\overline{AC}$  with  $AF = 13$ ,  $FG = 52$ , and  $GC = 26$ . Let  $M$  be the reflection of  $D$  through  $F$ , and let  $N$  be the reflection of  $G$  through  $E$ . Quadrilateral  $DEGF$  has area 288. Find the area of heptagon  $AFNBCEM$ .



## Why logic closeness useful? - Diagnosis



# DAG-MATH Formatted CoT and Benchmark construction

*DAG-MATH* explicitly specifies each reasoning step in forward generation order:  $Edge \rightarrow Parent(s) \rightarrow Node$ .

## Example DAG-MATH Formatted CoT for the logarithmic count problem

### Step 1

**Edge:** Restate the target equation from the problem statement,  $2 \log(x - 1) = \log k$ , which is the central equation to solve for  $x$  in terms of  $k$ .

**Parents:** Null

**Node:**  $2 \log(x - 1) = \log k$ .

:

### Step 4

**Edge:** Since the left-hand side of the equation in Step 1 contains  $\log(x - 1)$ , the domain restriction for a logarithm requires  $x - 1 > 0$ , i.e.,  $x > 1$ .

**Parents:** Step 1

**Node:**  $\log(x - 1)$  requires  $x > 1$ .

:



# DAG-MATH Formatted CoT and Benchmark construction

*DAG-MATH* explicitly specifies each reasoning step in forward generation order:  $Edge \rightarrow Parent(s) \rightarrow Node$ .

## Example DAG-MATH Formatted CoT for the logarithmic count problem

### Step 1

**Edge:** Restate the target equation from the problem statement,  $2 \log(x - 1) = \log k$ , which is the central equation to solve for  $x$  in terms of  $k$ .

**Parents:** Null

**Node:**  $2 \log(x - 1) = \log k$ .

:

### Step 4

**Edge:** Since the left-hand side of the equation in Step 1 contains  $\log(x - 1)$ , the domain restriction for a logarithm requires  $x - 1 > 0$ , i.e.,  $x > 1$ .

**Parents:** Step 1

**Node:**  $\log(x - 1)$  requires  $x > 1$ .

:

## Gold-standard CoT trajectory

- it adheres to the DAG-MATH format;
- its corresponding DAG is logically closed;
- the final answer is correct.

# DAG-MATH Formatted CoT and Benchmark construction

DAG-MATH explicitly specifies each reasoning step in forward generation order:  $Edge \rightarrow Parent(s) \rightarrow Node$ .

## Example DAG-MATH Formatted CoT for the logarithmic count problem

### Step 1

**Edge:** Restate the target equation from the problem statement,  $2 \log(x - 1) = \log k$ , which is the central equation to solve for  $x$  in terms of  $k$ .

**Parents:** Null

**Node:**  $2 \log(x - 1) = \log k$ .

:

### Step 4

**Edge:** Since the left-hand side of the equation in Step 1 contains  $\log(x - 1)$ , the domain restriction for a logarithm requires  $x - 1 > 0$ , i.e.,  $x > 1$ .

**Parents:** Step 1

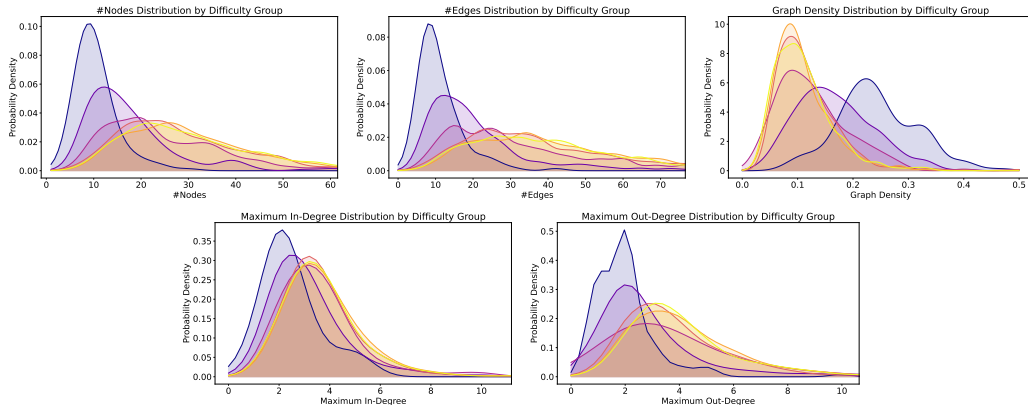
**Node:**  $\log(x - 1)$  requires  $x > 1$ .

:

DAG-MATH Benchmark with 2,894 gold-standard DAGs

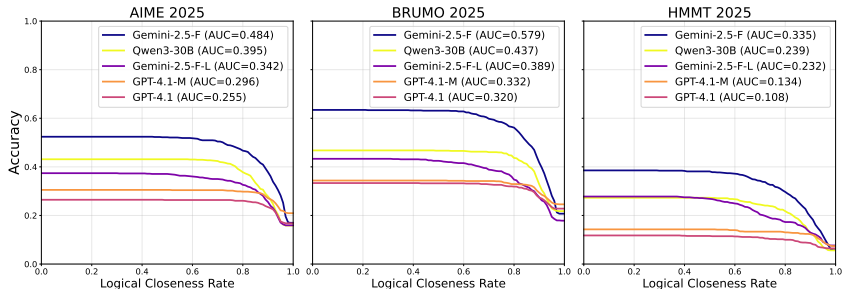
- Omni-MATH (difficulty level from 1 to 6)
- a three-stage prompting strategy via GPT-o4-mini and Qwen3-235B-A22B-Thinking-2507

# DAG Statistics Across Difficulty Levels



- Harder problems produce larger graphs.
- Harder problems produce sparser graphs.
- Max out-degree rises sharply; max in-degree grows slowly.

# Evaluation: AUC Curves Across Models



1. Search inflates PASS@1 but PRR remains similar across models, i.e., genuine reasoning ability is comparable.
2. Perfect trajectories = smallest, densest DAGs. Incorrect ones = large, sparse, heavy branching.
3. ~36% gap between PASS@1 and PRR (e.g. Gemini-2.5-Flash), statistically significant with 128 samples.

## Takeaway messages: balance natural and formal languages

---

□ Yuanhe Zhang, Ilja Kuzborskij, Jason D. Lee, Chenlei Leng, Fanghui Liu. *DAG-Math: Graph-Guided Mathematical Reasoning in LLMs* ICLR'26, [code]

- A DAG-MATH framework for reasoning evaluation
- Perfect reasoning metric (logic closeness): all models are similar
- Whether it can maintain compact, low-branching DAGs

## Takeaway messages: balance natural and formal languages

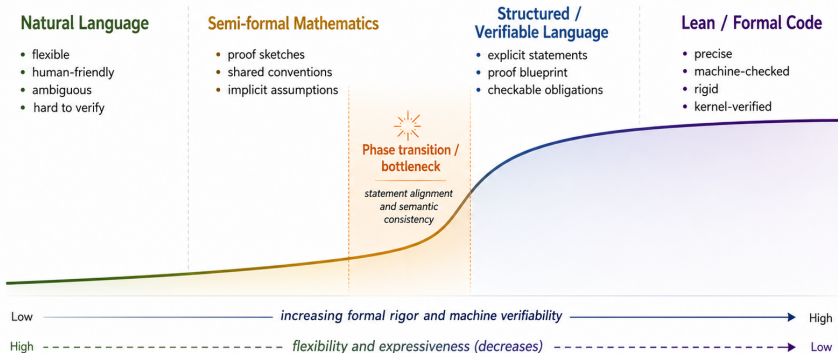
❑ Yuanhe Zhang, Ilja Kuzborskij, Jason D. Lee, Chenlei Leng, Fanghui Liu. *DAG-Math: Graph-Guided Mathematical Reasoning in LLMs* ICLR'26, [code]

- A DAG-MATH framework for reasoning evaluation
- Perfect reasoning metric (logic closeness): all models are similar
- Whether it can maintain compact, low-branching DAGs

### Future directions

- mixture of natural and formal language (e.g., Seed-Prover 1.5)
- Using DAG-MATH for reasoning performance improvement/Lean formalization
- Mathematical definition of reasoning

# From Natural Language to Lean: a Verification Phase Transition



(Generated by GPT5.5)



**Key challenge:** the hardest step is often not proof checking, but aligning mathematical intent with formal statements.

**AI4MATH...**

---



# Brief introduction of Lean 4

## Lean as a programming language

- We write a mathematical statement and construct a proof as **coding**
- Lean's trusted **kernel** checks that the proof is valid.

❑ State it, but not prove it

### Adding zero changes nothing

```
theorem add_zero
  (n : Nat) :
  n + 0 = n := by
  sorry
```

**sorry** as a placeholder:

*"I have not provided the proof yet."*

❑ State it and prove it

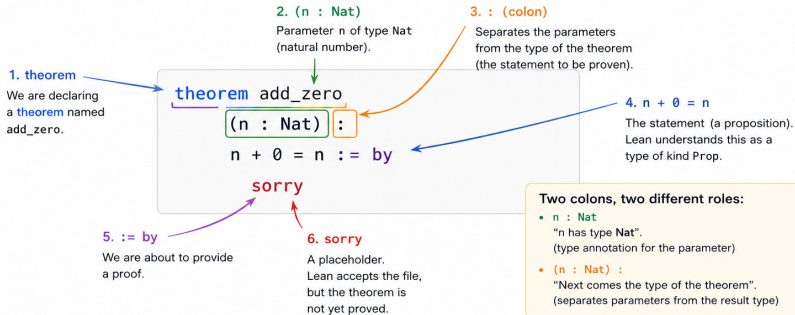
### Verified by Lean

```
theorem add_zero
  (n : Nat) :
  n + 0 = n := by
  simp
```

The **tactic** `simp` as a **strategy**:

constructs a proof via Lean's simplification rules.

## Understanding a Lean Theorem Declaration

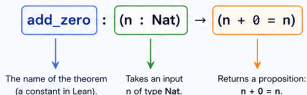


### What does this mean?

This declaration says:

"For any natural number `n`, we have `n + 0 = n`."

### As a function type in Lean:



### After proving

Replace `sorry` with a real proof, for example:

```
by
  simp
```

Lean's kernel will check the proof and mark the theorem as `verified`.

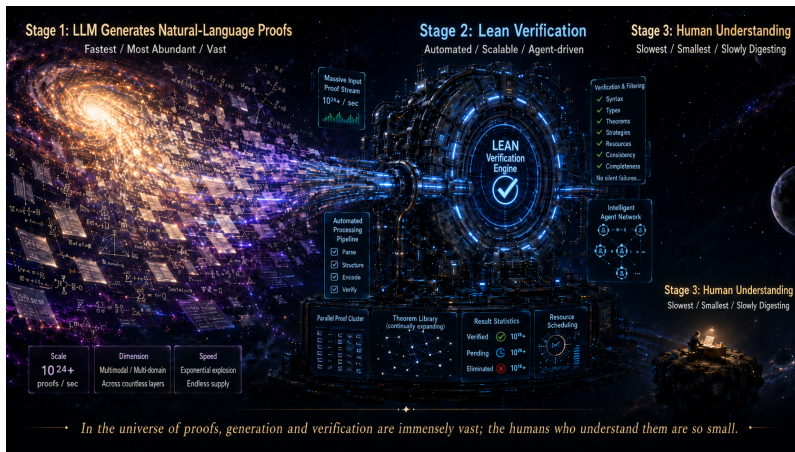
(Generated by GPT5.5)

# (Industrial) Revolution in Mathematics via LLMs

**Proof generation (LLMs)**  
Natural-language ideas,  
lemmas, candidate arguments

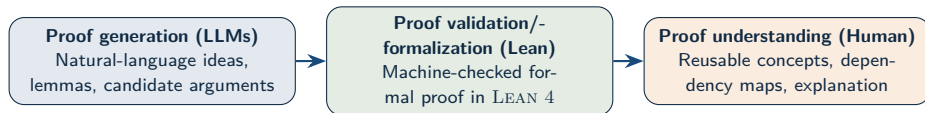
**Proof validation/-  
formalization (Lean)**  
Machine-checked formal  
proof in LEAN 4

**Proof understanding (Human)**  
Reusable concepts, dependency maps, explanation



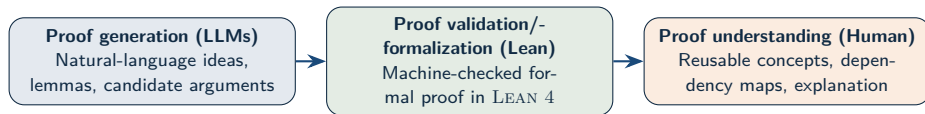
(Generated by GPT5.5)

# (Industrial) Revolution in Mathematics via LLMs



- **Validation is bottleneck**
  - plausibility is not correctness;
  - small gaps can invalidate a proof;
  - informal checking does not scale to libraries.
- **Understanding is the final goal**
  - Math aims to advance human understanding
  - not just a machine for generating formal proofs

# (Industrial) Revolution in Mathematics via LLMs

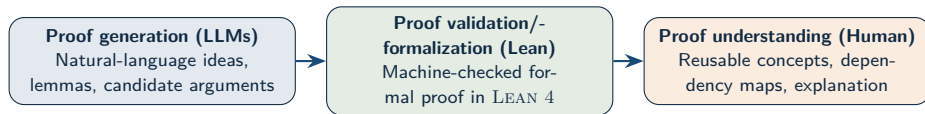


- **Validation is bottleneck**
  - plausibility is not correctness;
  - small gaps can invalidate a proof;
  - informal checking does not scale to libraries.
- **Understanding is the final goal**
  - Math aims to advance human understanding
  - not just a machine for generating formal proofs

Lean (verification) is not a silver bullet but helps **reduce** and **transfer** risk.

- **consistency** vs. **complete** [Gödel's incompleteness theorems]
- statement consistency

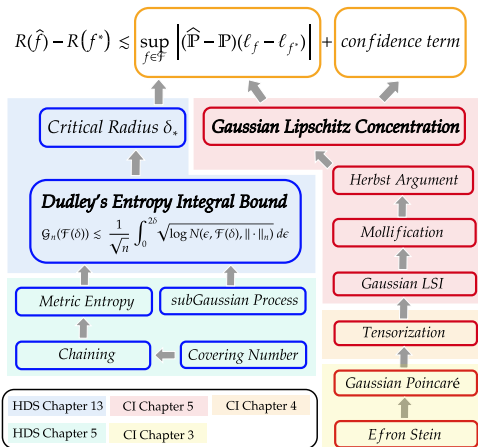
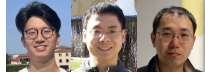
# (Industrial) Revolution in Mathematics via LLMs



- **Validation is bottleneck**
  - plausibility is not correctness;
  - small gaps can invalidate a proof;
  - informal checking does not scale to libraries.
- **Understanding is the final goal**
  - Math aims to advance human understanding
  - not just a machine for generating formal proofs

## Question shift

Can AI suggest a proof?  $\implies$  Can **WE** validate, reuse, and help understanding?



Wainwright (2019) (High-dimensional Statistics, HDS)

Boucheron et al. (2013) (Concentration Inequality, CI)

## Concentration

High-probability control of the empirical process fluctuation.

$$\mathbb{P}\{|f(X) - \mathbb{E}f(X)| \geq t\} \leq 2 \exp\left(-\frac{t^2}{2L^2}\right).$$

## Capacity control

Localize around  $f^*$ :

$$\mathcal{F}(\delta) = \{f \in \mathcal{F} : d(f, f^*) \leq \delta\},$$

then bound complexity by metric entropy:

$$G_n(\mathcal{F}(\delta)) \lesssim \frac{1}{\sqrt{n}} \int_0^{2\delta} \sqrt{\log N(\epsilon, \mathcal{F}(\delta), d)} d\epsilon.$$

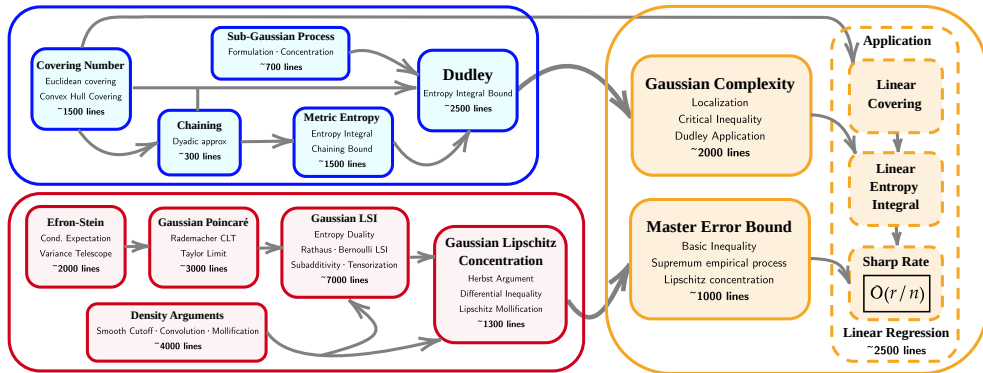
# Natural-language SLT vs. Lean 4 obligations

---

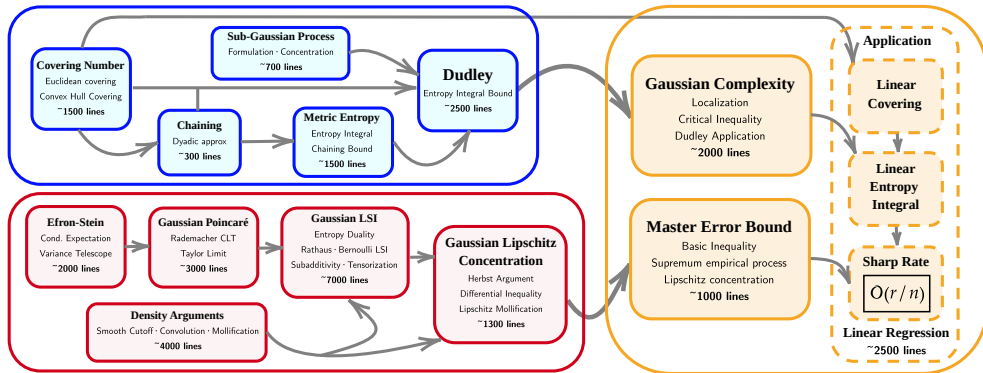
| Mathematical phrase            | What Lean asks for                                                        |
|--------------------------------|---------------------------------------------------------------------------|
| “Take a supremum over a class” | Topology, measurability, boundedness or extended-real formulation         |
| “Apply concentration”          | Exact Lipschitz structure, distributional assumptions, integrability      |
| “Use covering numbers”         | Metric-space structure, finite nets, cardinal coercions, monotonicity     |
| “Integrate entropy”            | $\mathbb{R}$ vs. $\mathbb{R}_{\geq 0} \cup \{\infty\}$ integral interface |
| “By density”                   | Approximation sequence, convergence mode, preservation of constants       |



# Dependency graph and deliverables



# Dependency graph and deliverables



## Main implementation

- Gaussian concentration analysis toolbox.
- Dudley's entropy integral.
- Human-AI workflow.

## Scale

|                 |                |
|-----------------|----------------|
| Lean 4 code     | ~ 30,000 lines |
| Theorems/lemmas | 1000+          |

## Structured TASK.md protocol

Each task specification contains:

1. also need NL target statement;
2. tactic-level proof plan (for agents);
3. relevant file paths and expected crossref;
4. hard boundaries for what the agent should not rewrite.

## Collaborative level

|                           |       |
|---------------------------|-------|
| Autonomous                | ~ 15% |
| Sketch-guided             | ~ 30% |
| Specified under NL proofs | ~ 40% |
| Human-critical            | ~ 15% |

# Human-AI collaborative formalization

## Structured TASK.md protocol

Each task specification contains:

1. also need NL target statement;
2. tactic-level proof plan (for agents);
3. relevant file paths and expected crossref;
4. hard boundaries for what the agent should not rewrite.

## Collaborative level

|                           |       |
|---------------------------|-------|
| Autonomous                | ~ 15% |
| Sketch-guided             | ~ 30% |
| Specified under NL proofs | ~ 40% |
| Human-critical            | ~ 15% |

## Takeaway

The dominant bottleneck is not tactic writing; it is converting compressed mathematical prose into precise, decomposed proof architecture.

## Practical lessons for formalization

---

- ❑ **Decompose aggressively.** Target small lemmas that fit within one agent context window.

## Practical lessons for formalization

---

- ❑ **Decompose aggressively.** Target small lemmas that fit within one agent context window.
- ❑ **Specify more than the theorem.** Include local APIs, intended proof route.

## Practical lessons for formalization

---

- ❑ **Decompose aggressively.** Target small lemmas that fit within one agent context window.
- ❑ **Specify more than the theorem.** Include local APIs, intended proof route.
- ❑ **Treat persistent failure as mathematical signal.** Recheck the statement and look for missing assumptions or counterexamples.

## Practical lessons for formalization

---

- ❑ **Decompose aggressively.** Target small lemmas that fit within one agent context window.
- ❑ **Specify more than the theorem.** Include local APIs, intended proof route.
- ❑ **Treat persistent failure as mathematical signal.** Recheck the statement and look for missing assumptions or counterexamples.
- ❑ **Make hidden regularity reusable.** Integrability, continuity, nonemptiness, and boundedness hypotheses become reusable theorem contracts.



## Practical lessons for formalization

---

- ❑ **Decompose aggressively.** Target small lemmas that fit within one agent context window.
- ❑ **Specify more than the theorem.** Include local APIs, intended proof route.
- ❑ **Treat persistent failure as mathematical signal.** Recheck the statement and look for missing assumptions or counterexamples.
- ❑ **Make hidden regularity reusable.** Integrability, continuity, nonemptiness, and boundedness hypotheses become reusable theorem contracts.
- ❑ **To agent: don't frequently change the proof!**

## Practical lessons for formalization

- ❑ **Decompose aggressively.** Target small lemmas that fit within one agent context window.
- ❑ **Specify more than the theorem.** Include local APIs, intended proof route.
- ❑ **Treat persistent failure as mathematical signal.** Recheck the statement and look for missing assumptions or counterexamples.
- ❑ **Make hidden regularity reusable.** Integrability, continuity, nonemptiness, and boundedness hypotheses become reusable theorem contracts.
- ❑ **To agent: don't frequently change the proof!**
  - Know more details (e.g., implicit assumptions)
  - Know which key infrastructure is reused (structure view, multi-scale understanding)
  - proof connections (e.g., Gaussian Poincaré by Efron-Stein, Hermite decomposition, OU semigroup)

If Mathlib is “complete”, can **we** autoformalize research-level problems even with proof gap?

If Mathlib is “complete”, can **we** autoformalize research-level problems even with proof gap?



## LeanMarathon: Toward Reliable AI Co-Mathematicians through Long-Horizon Lean Autoformalization

Yuanhe Zhang\* Yuekai Sun† Taiji Suzuki‡ Jason D. Lee§ Fanghui Liu¶



Erdős problems (#1051, #1196, #164, #1217) ([Barreto et al., 2026](#); [Alexeev et al., 2026](#))

# Why research-level autoformalization is hard

---

## Not isolated theorem proving

- A paper may contain hundreds of definitions, lemmas, and theorems.
- Formal statements, prose proof sketches, and dependencies must evolve together.
- A wrong early statement can contaminate substantial downstream work.

## Not merely prompt engineering

- Context windows compact or lose state.
- Agents over-trust their own blueprints.
- A local repair can silently break distant completed proofs.

# Why research-level autoformalization is hard

## Not isolated theorem proving

- A paper may contain hundreds of definitions, lemmas, and theorems.
- Formal statements, prose proof sketches, and dependencies must evolve together.
- A wrong early statement can contaminate substantial downstream work.

## Not merely prompt engineering

- Context windows compact or lose state.
- Agents over-trust their own blueprints.
- A local repair can silently break distant completed proofs.

## Systems perspective

We need a **fault-containment harness**: assume agents will drift, overclaim, and occasionally formalize the wrong target; make each error local, observable, and recoverable.

# Why we need the harness?



Two major failure modes:

- **Goal drift** – agent's working trace drifts from contributing to the target.
- **Lost-in-the-middle** – agent lost in the uncertain middle place of formalization.

Three causes:

- **Coherence loss; Self-evaluation bias; Irreversibility.**

## Design principle

Narrow input, single output, bounded scope, deterministic gate, fault containment

## Three long-horizon failure modes

---

| Failure mode                | What it looks like                                                                                             | System consequence                            |
|-----------------------------|----------------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| <b>Coherence loss</b>       | The agent forgets the global proof plan and patches one region by damaging another.                            | The run drifts away from the target.          |
| <b>Self-evaluation bias</b> | The agent declares its blueprint or proof attempt correct despite missing hypotheses or mismatched statements. | Bad specifications become downstream tasks.   |
| <b>Irreversibility</b>      | A root-level statement error makes later lemmas formally valid but mathematically irrelevant.                  | Prover compute is spent on the wrong problem. |



## What the harness does

---



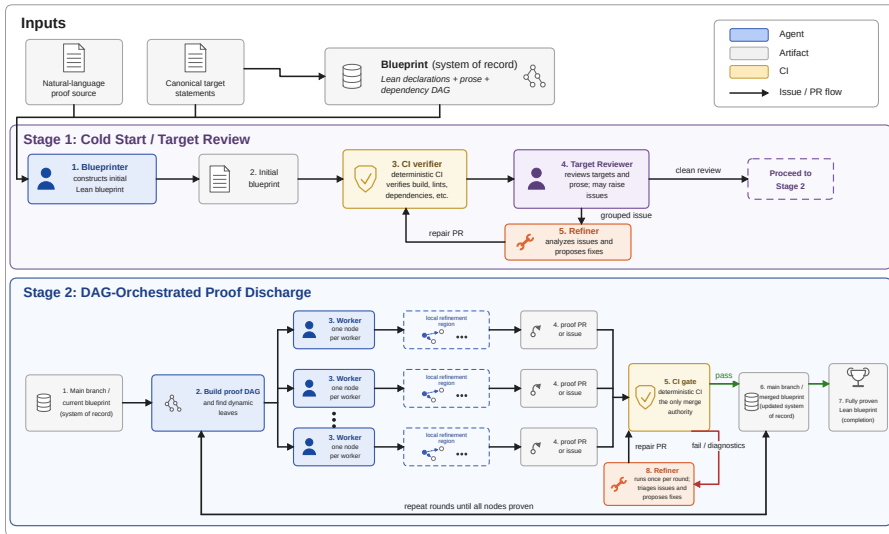
# What the harness does



## Core ideas

- **Blueprint as substrate.** A single Lean file in *blueprint* format maintained across the whole run.
- **Specialist agents.** Each role has well-separated task, under skilled workflow, within boundaries.
- **System orchestration.** Lean file and GitHub PR/issue stream coordinate agents.
- **Verifiable gates.** *Deterministic* gate & wrapper to calibrate system's trace.

# LeanMarathon in one picture



# Blueprint: the system of record

## Definition (Zhu et al., 2026)

The **blueprint** is a single Lean file that simultaneously acts as:

```
@[blueprint "lem:weighted-tail-bound"  
  (statement := /-- LaTeX statement text -/  
  (proof     := /-- LaTeX proof prose with \cref{...} citations -/  
  (title     := /-- one-line LaTeX title -/  
  (latexEnv  := "lemma")]  
lemma weighted_tail_bound ... : ... := by  
  sorry_using [aux_lemma_one, aux_lemma_two]
```

- Proof bodies are either **by sorry**, **by sorry\_using [...]**, or **complete Lean proofs**.
- Definitions are global context; lemma/theorem declarations form the proof DAG.

## Principle

Agents do not rely on hidden chat history or a shared scratchpad. Durable mathematical state lives in the blueprint and Git history.

# Dependency–prose parity: Avoid drift

## Prose side

- LaTeX proof text exposes intended dependencies via `\cref{...}`.
- Placeholder proofs expose intended parents via `sorry_using [...]`.

## Lean side

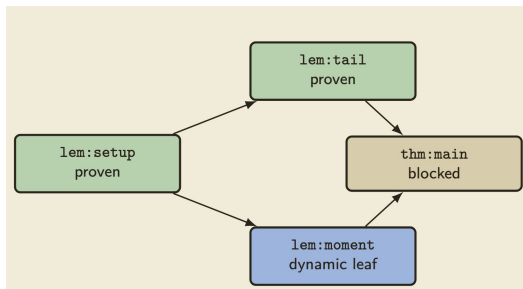
- The verifier extracts actual dependencies from elaborator metadata.
- These dependencies reflect what the proof really uses.

```
@[blueprint "lem:C"  
  (statement := /-- LaTeX statement text -/)  
  (proof     := /-- By \cref{lem:A} and \cref{lem:B}, we obtain the claim. -/)  
  (title     := /-- one-line LaTeX title -/)  
  (latexEnv  := "lemma")]  
lemma C ... : ... := by  
  exact proof_using_A_and_B
```

## Why this matters

Without parity, a long-running agent can make the Lean proof and the human-readable proof graph silently diverge.

# Blueprint as a Proof DAG



## Interpretation

- Nodes are declarations of lemma/theorem.
- Edges are cited and Lean-elaborated dependencies.
- Dynamic leaves are unproven nodes whose all ancestors are proven.

## Why a DAG matters

It turns one long formalization into a graph-induced task plan with mechanically computable scheduling and repair scope.

## The four contract-scoped agents

| Agent                  | Primary role                                                                    | Failure it avoids                                |
|------------------------|---------------------------------------------------------------------------------|--------------------------------------------------|
| <b>Blueprinter</b>     | Builds the initial typed blueprint from the source proof and canonical targets. | No executable proof graph for later scheduling.  |
| <b>Target-Reviewer</b> | Audits target fidelity before large-scale proving begins.                       | Proving the wrong theorem.                       |
| <b>Worker</b>          | Proves one proof node; may add local helper lemmas.                             | Local proof failure corrupting global state.     |
| <b>Refiner</b>         | Repairs connected blueprint-level defects reported as issues.                   | Inconsistent local fixes to a shared root cause. |

### Key idea

Each agent has a narrow interface, bounded edit scope, and an externally checkable output.

## Agent contracts at a glance

| Agent           | Main input          | Main output                 | Direct edit main branch? |
|-----------------|---------------------|-----------------------------|--------------------------|
| Blueprinter     | source + targets    | initial blueprint PR        | ✗                        |
| Target-Reviewer | blueprint + targets | clean exit or grouped issue | ✗                        |
| Worker          | one proof node      | proof PR or issue           | ✗                        |
| Refiner         | issue set + DAG     | repair PR                   | ✗                        |
| CI verifier     | PR + base branch    | pass/fail diagnostics       | ✓ only authority         |

### Slogan

Agents propose. The deterministic verifier judges.



## Two-stage orchestration

### Stage 1: Cold Start / Target Review

- Generate the initial blueprint.
- Run structural CI.
- Audit target fidelity.
- Refine until the review is clean.

### Stage 2: DAG-Orchestrated Proof Discharge

- Extract the current proof DAG.
- Find dynamic leaves.
- Dispatch Workers in parallel.
- Merge proof PRs through CI; repair issues by Refiner.

### Termination condition

- Every proof node has a complete Lean proof;
- no sorry or sorry\_using remains;
- all CI checks pass.

## Experimental results

We pick three recent Erdős problems solved by AI in natural language:

- **Erdős 1051:** Barreto, Kevin, Jiwon Kang, Sang-hyun Kim, Vjekoslav Kovač, and Shengtong Zhang. "Irrationality of rapidly converging series: a problem of Erdős and Graham." arXiv preprint arXiv:2601.21442 (2026).
- **Erdős 1196 & 164 & 1217:** Alexeev, Boris, Kevin Barreto, Yanyang Li, Jared Duker Lichtman, Liam Price, Jibran Iqbal Shah, Quanyu Tang, and Terence Tao. "Primitive sets and von Mangoldt chains: Erdős Problem #1196 and beyond." arXiv preprint arXiv:2605.00301 (2026).

|           | Erdős 1051 | Erdős 1196 | Erdős 164 | Erdős 1217 |
|-----------|------------|------------|-----------|------------|
| Aristotle | 7          | 7          | 7         | —          |
| Ours      | 3          | 3          | 3         | 3          |

(GPT-5.5-xhigh, 258K, no web access)

## Experimental results

| Metric                        | Erdős–Graham | ESS #1196 | #164 & #1217 |
|-------------------------------|--------------|-----------|--------------|
| Rounds                        | 19           | 17        | 40           |
| WORKERS launched              | 58           | 33        | 111          |
| REFINERS                      | 7            | 6         | 25           |
| Merged PRs                    | 53           | 32        | 93           |
| Critical path (excl. CI wait) | 11:38:23     | 11:32:40  | 40:43:21     |
| Aggregate agent active time   | 21:29:41     | 16:26:32  | 71:16:52     |
| Stop-hook / CI wait time      | 07:15:10     | 02:15:33  | 07:05:26     |
| Tool-call parsed wall time    | 10:16:25     | 06:16:40  | 49:08:03     |
| Total tool calls              | 5,273        | 4,067     | 12,204       |
| Total tokens                  | 308M         | 245M      | 796M         |
| GPT-5.5 API-equiv. cost       | \$257.17     | \$189.43  | \$623.54     |

## Experimental results - Baseline

|                      | Erdős–Graham |                 | ESS #1196 |                 |
|----------------------|--------------|-----------------|-----------|-----------------|
|                      | Aristotle    | Ours            | Aristotle | Ours            |
| Targets proven       | 0 / 3        | 3 / 3           | 0 / 1     | 1 / 1           |
| Lean lines delivered | 751          | 8,513           | 24        | 3,988           |
| Remaining sorry      | 2            | 0               | 1         | 0               |
| Outcome              | failed       | <b>complete</b> | failed    | <b>complete</b> |

The run caught a genuine gap in the published proof (Case C of Proposition 9 in the paper).

- **Issue:** the claim is stronger than what hypotheses actually give.
- **Fix:** makes the statements weaker/more constrained, but in exchange they become actually provable from the available hypotheses.

# Takeaway

---

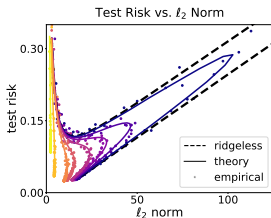
Research-level autoformalization should be treated as a **fault-contained systems problem**, not merely as a larger proof-generation prompt.

## Takeaway

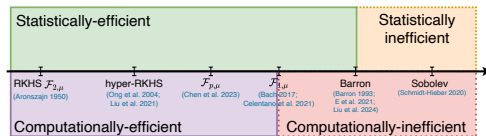
- not make one agent think longer
- make multiple fallible agents collaborate

# Takeaway: From SLT4AI to AI4SLT

- ❑ The  $\varphi$  curve: be aware of model capacity!



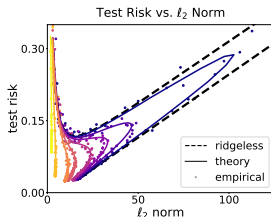
- ❑ Trade-off between sample complexity and dimension dependence



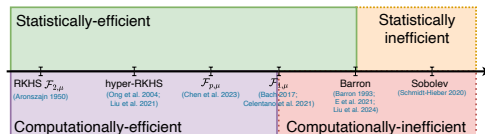
- ❑ One-step full gradient suffices to LLM low-rank fine-tuning!

# Takeaway: From SLT4AI to AI4SLT

❑ The  $\varphi$  curve: be aware of model capacity!



❑ Trade-off between sample complexity and dimension dependence



❑ One-step full gradient suffices to LLM low-rank fine-tuning!

## AI4MATH

- modelling mathematical reasoning as DAG;
- Infrastructure in AI4SLT;
- Lean-Marathon: harness engineering for auto-formalization

## Lean 4: why it matters

- explicit assumptions and scale up;
- Machine-checked used for the review system and insights for AI4AI/ML;
- **Multi-scale understanding.**



## Q & A

[fanghui.liu@sjtu.edu.cn](mailto:fanghui.liu@sjtu.edu.cn)

[www.lfhsgre.org](http://www.lfhsgre.org)

## References

---

- Emmanuel Abbe, Enric Boix Adsera, and Theodor Misiakiewicz. The merged-staircase property: a necessary and nearly sufficient condition for sgd learning of sparse functions on two-layer neural networks. In *Conference on Learning Theory*, pages 4782–4887, 2022.
- Boris Alexeev, Kevin Barreto, Yanyang Li, Jared Duker Lichtman, Liam Price, Jibran Iqbal Shah, Quanyu Tang, and Terence Tao. Primitive sets and von Mangoldt chains: Erdős Problem #1196 and beyond, 2026. URL <https://arxiv.org/abs/2605.00301>.
- Gerard Ben Arous, Reza Gheissari, and Aukosh Jagannath. Online stochastic gradient descent on non-convex losses from high-dimensional inference. *Journal of Machine Learning Research*, 22(106):1–51, 2021.

## References ii

- Francis Bach. High-dimensional analysis of double descent for linear regression with random projections. *SIAM Journal on Mathematics of Data Science*, 6(1):26–50, 2024.
- Kevin Barreto, Jiwon Kang, Sang-hyun Kim, Vjekoslav Kovač, and Shengtong Zhang. Irrationality of rapidly converging series: a problem of Erdős and Graham, 2026. URL <https://arxiv.org/abs/2601.21442>.
- Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.
- Peter Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.
- Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *the National Academy of Sciences*, 116(32):15849–15854, 2019.

## References iii

- Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 02 2013. URL <https://doi.org/10.1093/acprof:oso/9780199535255.001.0001>.
- Sitan Chen and Shyam Narayanan. A faster and simpler algorithm for learning shallow networks. *arXiv preprint arXiv:2307.12496*, 2023.
- Chen Cheng and Andrea Montanari. Dimension free ridge regression. *The Annals of Statistics*, 52(6):2879–2912, 2024.
- Alexandru Damian, Jason Lee, and Mahdi Soltanolkotabi. Neural networks can learn representations with gradient descent. In *Conference on Learning Theory*, pages 5413–5452. PMLR, 2022.

## References iv

- Yatin Dandi, Emanuele Troiani, Luca Arnaboldi, Luca Pesce, Lenka Zdeborova, and Florent Krzakala. The benefits of reusing batches for gradient descent in two-layer networks: Breaking the curse of information and leap exponents. In *Forty-first International Conference on Machine Learning*, 2024.
- Leonardo Defilippis, Bruno Loureiro, and Theodor Misiakiewicz. Dimension-free deterministic equivalents for random feature regression. In *Advances in Neural Information Processing Systems*, 2024.
- Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems*, volume 36, pages 70293–70332, 2023.
- Weinan E, Chao Ma, and Lei Wu. The barron space and the flow-induced function spaces for neural network models. *Constructive Approximation*, pages 1–38, 2021.

## References v

- Per Christian Hansen. Analysis of discrete ill-posed problems by means of the l-curve. *SIAM Review*, 34(4):561–580, 1992.
- Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.
- Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *Annals of Statistics*, 50(2):949–986, 2022.
- Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

## References vi

- Juno Kim, Denny Wu, Jason D Lee, and Taiji Suzuki. Metastable dynamics of chain-of-thought reasoning: Provable benefits of search, rl and distillation. In *International Conference on Machine Learning*, 2025.
- Jason D Lee, Kazusato Oko, Taiji Suzuki, and Denny Wu. Neural network learns low-dimensional polynomials with sgd near the information-theoretic limit. *arXiv preprint arXiv:2406.01581*, 2024.
- Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan AK Suykens. Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7128–7148, 2021.
- Fanghui Liu, Leello Dadi, and Volkan Cevher. Learning with norm constrained, over-parameterized, two-layer neural networks. *Journal of Machine Learning Research*, 25(138):1–42, 2024.

## References vii

- Kai Lv, Yuqing Yang, Tengxiao Liu, Qipeng Guo, and Xipeng Qiu. Full Parameter Fine-tuning for Large Language Models with Limited Resources. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8187–8198, 2024.
- Theodor Misiakiewicz and Basil Saeed. A non-asymptotic theory of kernel ridge regression: deterministic equivalents, test error, and gcv estimator. *arXiv preprint arXiv:2403.08938*, 2024.
- Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2019.
- Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401. PMLR, 2015.



## References viii

Andrew Ng and Tengyu Ma. CS229 lecture notes. 2023. URL

[https://cs229.stanford.edu/main\\_notes.pdf](https://cs229.stanford.edu/main_notes.pdf).

Rahul Parhi and Robert D Nowak. Near-minimax optimal estimation with shallow ReLU neural networks. *IEEE Transactions on Information Theory*, 2022.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2007.

Pedro Savarese, Itay Evron, Daniel Soudry, and Nathan Srebro. How do infinite width bounded norm networks look in function space? In *Conference on Learning Theory*, pages 2667–2690. PMLR, 2019.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

## References ix

- Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-rank adaptation with gradient approximation. In *Advances in Neural Information Processing Systems*, 2024.
- Yichen Wang, Yudong Chen, Lorenzo Rosasco, and Fanghui Liu. The  $\varphi$  curve: The shape of generalization through the lens of norm-based capacity control. In *Advances in Neural Information Processing Systems*, pages 114595–114670, 2025.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, pages 24824–24837, 2022.
- Denny Wu and Ji Xu. On the optimal weighted  $\ell_2$  regularization in overparameterized linear regression. In *Advances in Neural Information Processing Systems*, pages 10112–10123, 2020.

## References x

- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.
- Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=Tn5B6Udq3E>.
- Thomas Zhu, Pietro Monticone, Jeremy Avigad, and Sean Welleck. LeanArchitect: Automating blueprint generation for humans and AI. *arXiv preprint arXiv:2601.22554*, 2026.

## Background: Random features ridge regression

$$f_p(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{i=1}^p \textcolor{red}{a}_i \phi(\boldsymbol{x}, \textcolor{blue}{w}_i), \quad \boldsymbol{\theta} := \{(a_i, \boldsymbol{w}_i)\}_{i=1}^p$$

- $\phi : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}$ , e.g., ReLU:  $\phi(\boldsymbol{x}, \boldsymbol{w}) = \max(\langle \boldsymbol{x}, \boldsymbol{w} \rangle, 0)$

## Background: Random features ridge regression

$$f_p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{i=1}^p \textcolor{red}{a}_i \phi(\mathbf{x}, \textcolor{blue}{w}_i), \quad \boldsymbol{\theta} := \{(a_i, \mathbf{w}_i)\}_{i=1}^p$$

- $\phi : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}$ , e.g., ReLU:  $\phi(\mathbf{x}, \mathbf{w}) = \max(\langle \mathbf{x}, \mathbf{w} \rangle, 0)$
- Random features models (RFMs) (Rahimi and Recht, 2007; Liu et al., 2021):
  - $\{\mathbf{w}_i\}_{i=1}^p \stackrel{iid}{\sim} \mu$  for a given  $\mu \in \mathcal{P}(\mathcal{W})$
  - only train the second layer

## Background: Random features ridge regression

$$f_p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{i=1}^p \mathbf{a}_i \phi(\mathbf{x}, \mathbf{w}_i), \quad \boldsymbol{\theta} := \{(a_i, \mathbf{w}_i)\}_{i=1}^p$$

- $\phi : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}$ , e.g., ReLU:  $\phi(\mathbf{x}, \mathbf{w}) = \max(\langle \mathbf{x}, \mathbf{w} \rangle, 0)$
- Random features models (RFMs) (Rahimi and Recht, 2007; Liu et al., 2021):
  - $\{\mathbf{w}_i\}_{i=1}^p \stackrel{iid}{\sim} \mu$  for a given  $\mu \in \mathcal{P}(\mathcal{W})$
  - only train the second layer

$$\hat{\mathbf{a}} := \operatorname{argmin}_{\mathbf{a} \in \mathbb{R}^p} \left\{ \sum_{i=1}^n (y_i - f(\mathbf{x}_i; \mathbf{a}))^2 + \lambda \|\mathbf{a}\|_2^2 \right\} = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I}_p)^{-1} \mathbf{Z}^\top \mathbf{y}.$$

- $\mathbf{Z} \in \mathbb{R}^{n \times p}$  with  $[\mathbf{Z}]_{ij} = \frac{1}{\sqrt{m}} \phi(\mathbf{x}_i, \mathbf{w}_j)$ .

## Background: Random features ridge regression

$$f_p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{i=1}^p \mathbf{a}_i \phi(\mathbf{x}, \mathbf{w}_i), \quad \boldsymbol{\theta} := \{(\mathbf{a}_i, \mathbf{w}_i)\}_{i=1}^p$$

- $\phi : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}$ , e.g., ReLU:  $\phi(\mathbf{x}, \mathbf{w}) = \max(\langle \mathbf{x}, \mathbf{w} \rangle, 0)$
- Random features models (RFMs) (Rahimi and Recht, 2007; Liu et al., 2021):
  - $\{\mathbf{w}_i\}_{i=1}^p \stackrel{iid}{\sim} \mu$  for a given  $\mu \in \mathcal{P}(\mathcal{W})$
  - only train the second layer

$$\hat{\mathbf{a}} := \operatorname{argmin}_{\mathbf{a} \in \mathbb{R}^p} \left\{ \sum_{i=1}^n (y_i - f(\mathbf{x}_i; \mathbf{a}))^2 + \lambda \|\mathbf{a}\|_2^2 \right\} = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I}_p)^{-1} \mathbf{Z}^\top \mathbf{y}.$$

- $\mathbf{Z} \in \mathbb{R}^{n \times p}$  with  $[\mathbf{Z}]_{ij} = \frac{1}{\sqrt{m}} \phi(\mathbf{x}_i, \mathbf{w}_j)$ .
  - Norm over the first-layer (untrained)  $\|\mathbf{W}\|_F$
  - Norm over the second-layer  $\|\hat{\mathbf{a}}\|_2^2$

## Background: Test risk of random features model

- A **compact** integral operator  $\mathbb{T} : L^2(\rho_X) \rightarrow L^2(\mu_{\mathcal{W}})$  for any  $f \in L^2(\rho_X)$  (Defilippis et al., 2024)

$$(\mathbb{T}f)(\boldsymbol{w}) := \int_{\mathbb{R}^d} \phi(\boldsymbol{x}, \boldsymbol{w}) f(\boldsymbol{x}) d\rho(\boldsymbol{x}), \quad \mathbb{T} = \sum_{k=1}^{\infty} \xi_k \psi_k \varphi_k^*.$$



## Background: Test risk of random features model

- A **compact** integral operator  $\mathbb{T} : L^2(\rho_X) \rightarrow L^2(\mu_{\mathcal{W}})$  for any  $f \in L^2(\rho_X)$  (Defilippis et al., 2024)

$$(\mathbb{T}f)(\boldsymbol{w}) := \int_{\mathbb{R}^d} \phi(\boldsymbol{x}, \boldsymbol{w}) f(\boldsymbol{x}) d\rho(\boldsymbol{x}), \quad \mathbb{T} = \sum_{k=1}^{\infty} \xi_k \psi_k \varphi_k^*.$$

- Covariate feature matrix  $\boldsymbol{G} := [\boldsymbol{g}_1, \dots, \boldsymbol{g}_n]^\top \in \mathbb{R}^{n \times \infty}$  with  $\boldsymbol{g}_i := (\psi_k(\boldsymbol{x}_i))_{k \geq 1}$
- Weight feature matrix  $\boldsymbol{H} := [\boldsymbol{h}_1, \dots, \boldsymbol{h}_p]^\top \in \mathbb{R}^{p \times \infty}$  with  $\boldsymbol{h}_j := (\xi_k \varphi_k(\boldsymbol{w}_j))_{k \geq 1}$

## Background: Test risk of random features model

- A **compact** integral operator  $\mathbb{T} : L^2(\rho_X) \rightarrow L^2(\mu_{\mathcal{W}})$  for any  $f \in L^2(\rho_X)$  (Defilippis et al., 2024)

$$(\mathbb{T}f)(\boldsymbol{w}) := \int_{\mathbb{R}^d} \phi(\boldsymbol{x}, \boldsymbol{w}) f(\boldsymbol{x}) d\rho(\boldsymbol{x}), \quad \mathbb{T} = \sum_{k=1}^{\infty} \xi_k \psi_k \varphi_k^*.$$

- Covariate feature matrix  $\boldsymbol{G} := [\boldsymbol{g}_1, \dots, \boldsymbol{g}_n]^\top \in \mathbb{R}^{n \times \infty}$  with  $\boldsymbol{g}_i := (\psi_k(\boldsymbol{x}_i))_{k \geq 1}$
- Weight feature matrix  $\boldsymbol{H} := [\boldsymbol{h}_1, \dots, \boldsymbol{h}_p]^\top \in \mathbb{R}^{p \times \infty}$  with  $\boldsymbol{h}_j := (\xi_k \varphi_k(\boldsymbol{w}_j))_{k \geq 1}$
- target function:  $f_*(\boldsymbol{x}) = \sum_{k \geq 1} \boldsymbol{\theta}_{*,k} \psi_k(\boldsymbol{x})$

## Background: Test risk of random features model

- A **compact** integral operator  $\mathbb{T} : L^2(\rho_X) \rightarrow L^2(\mu_{\mathcal{W}})$  for any  $f \in L^2(\rho_X)$  (Defilippis et al., 2024)

$$(\mathbb{T}f)(\boldsymbol{w}) := \int_{\mathbb{R}^d} \phi(\boldsymbol{x}, \boldsymbol{w}) f(\boldsymbol{x}) d\rho(\boldsymbol{x}), \quad \mathbb{T} = \sum_{k=1}^{\infty} \xi_k \psi_k \varphi_k^*.$$

- Covariate feature matrix  $\boldsymbol{G} := [\boldsymbol{g}_1, \dots, \boldsymbol{g}_n]^\top \in \mathbb{R}^{n \times \infty}$  with  $\boldsymbol{g}_i := (\psi_k(\boldsymbol{x}_i))_{k \geq 1}$
- Weight feature matrix  $\boldsymbol{H} := [\boldsymbol{h}_1, \dots, \boldsymbol{h}_p]^\top \in \mathbb{R}^{p \times \infty}$  with  $\boldsymbol{h}_j := (\xi_k \varphi_k(\boldsymbol{w}_j))_{k \geq 1}$
- target function:  $f_*(\boldsymbol{x}) = \sum_{k \geq 1} \boldsymbol{\theta}_{*,k} \psi_k(\boldsymbol{x})$

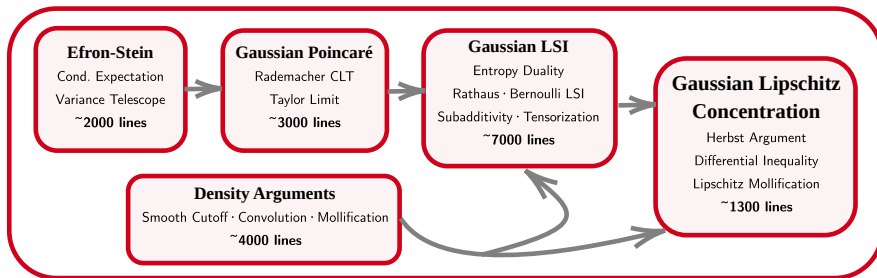
$$\mathcal{R}^{\text{RFM}} := \mathbb{E}_{\varepsilon} \left\| \boldsymbol{\theta}_* - \frac{1}{\sqrt{p}} \boldsymbol{H}^\top \hat{\boldsymbol{a}} \right\|_2^2$$

# Gaussian concentration: target theorem

## Gaussian Lipschitz concentration

Let  $X \sim \mathcal{N}(0, I_n)$  and  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be  $L$ -Lipschitz. Then, for all  $t \geq 0$ ,

$$\mathbb{P}\{f(X) - \mathbb{E}f(X) \geq t\} \leq \exp\left(-\frac{t^2}{2L^2}\right).$$



## Example interface: Efron–Stein inequality

### Mathematical statement

Let  $X = (X_1, \dots, X_n)$  be a vector of  $n$  independent random variables and  $Z = f(X)$  square-integrable. With  $\mathbb{E}^{(i)}$  denoting conditional expectation over all coordinates except  $i$ ,

$$\text{Var}(Z) \leq \sum_{i=1}^n \mathbb{E} \left[ \left( Z - \mathbb{E}^{(i)}[Z] \right)^2 \right].$$

□ Leave-one-coordinate conditional expectation

```
noncomputable def condExpExceptCoord
  (i : Fin n) (f : (Fin n -> Omega) -> R) :
  (Fin n -> Omega) -> R :=
  fun x => integral y, f (Function.update x i y) d(mus i)

theorem efronStein
  (f : (Fin n -> Omega) -> R) (hf : MemLp f 2 mus) :
  variance f mus <=
    sum i : Fin n, integral x, (f x - condExpExceptCoord i f x)^2 dmus := by
```

# Why is Lean formalization complex?

We want to prove

$$\text{Law}(X^{(i)}) = \mu_1 \otimes \cdots \otimes \mu_n$$

## Key Point

- ❑ Natural language: *Resampling one independent coordinate preserves the joint distribution*
- ❑ Lean: *Prove that a pushforward of a product measure equals the original product measure*

# Why is Lean formalization complex?

We want to prove

$$\text{Law}(X^{(i)}) = \mu_1 \otimes \cdots \otimes \mu_n$$

## Key Point

- ❑ **Natural language:** *Resampling one independent coordinate preserves the joint distribution*
- ❑ **Lean:** *Prove that a pushforward of a product measure equals the original product measure*

product measures   measurable maps   pushforwards   Fubini's theorem   measure rectangles.

# Why is Lean formalization complex?

We want to prove

$$\text{Law}(X^{(i)}) = \mu_1 \otimes \cdots \otimes \mu_n$$

## Key Point

- ❑ **Natural language:** *Resampling one independent coordinate preserves the joint distribution*
- ❑ **Lean:** *Prove that a pushforward of a product measure equals the original product measure*

product measures   measurable maps   pushforwards   Fubini's theorem   measure rectangles.

## Density argument (missing in the textbook ([Boucheron et al., 2013](#)))

Gaussian LSI on  $C_c^\infty \implies$  Gaussian LSI on  $W^{1,2}(\gamma)$

Gaussian **Lipschitz** concentration on  $C_c^\infty \implies \dots$



# Dudley's entropy integral

## Mathematical statement

Let  $(A, d)$  be a pseudo-metric space,  $T \subseteq A$  totally bounded,  $\text{diam}(s) \leq D$ , and  $\{X_t\}_{t \in T}$  a normalized sub-Gaussian process with parameter  $\sigma$ . Under integrability and path-continuity hypotheses,

$$\mathbb{E} \left[ \sup_{t \in T} X_t \right] \leq 12\sqrt{2} \sigma \int_0^D \sqrt{\log \mathcal{N}(\epsilon, T, d)} \, d\epsilon.$$

# Dudley's entropy integral

## Mathematical statement

Let  $(A, d)$  be a pseudo-metric space,  $T \subseteq A$  totally bounded,  $\text{diam}(s) \leq D$ , and  $\{X_t\}_{t \in T}$  a normalized sub-Gaussian process with parameter  $\sigma$ . Under integrability and path-continuity hypotheses,

$$\mathbb{E} \left[ \sup_{t \in T} X_t \right] \leq 12\sqrt{2} \sigma \int_0^D \sqrt{\log \mathcal{N}(\epsilon, T, d)} \, d\epsilon.$$

## Why it is complex in Lean

The proof combines finite nets, sub-Gaussian maximum inequalities, dyadic chaining, Fatou-type limits, and real vs. nonnegative integration APIs.

## One proof-engineering issue

The convenient measure-theoretic integral lives in  $\mathbb{R}_{\geq 0} \cup \{\infty\}$ , while downstream SLT inequalities are stated over  $\mathbb{R}$ .