

A Hessian View of Fine-tuning, Task Attribution, and Reinforcement Learning

Three Vignettes in Modern Machine Learning

Hongyang R. Zhang
Assistant Professor of Computer Science
Northeastern University, Boston

February 14, 2026



Overarching goal: an information-theoretic measure of modern AI models

AI models have found applications across assistive driving, multi-modal generation, speech recognition, coding, math, predicted for massive automation



 Gemini

 Claude

GPT-3 and scaling laws [BMR+20]: One of the earliest findings is that large transformer models trained to predict next tokens on massive corpus show impressive adaptivity to downstream tasks

Overarching goal: an information-theoretic measure of modern AI models

AI models have found applications across assistive driving, multi-modal generation, speech recognition, coding, math, predicted for massive automation



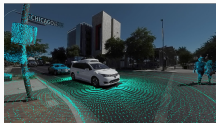
◆ Gemini ✨ Claude

GPT-3 and scaling laws [BMR+20]: One of the earliest findings is that large transformer models trained to predict next tokens on massive corpus show impressive adaptivity to downstream tasks

- **Generalization in over-parameterized models** [ZBH+21]: From a theoretical perspective, deep nets such as ResNet, BERT have more parameters than labels to memorize training dataset
- The notion of entropy for measuring information in a language dates back to classical work by Claude Shannon [Sha48]. Applying entropy to N -gram statistics provides one of the first information measure in text

Overarching goal: an information-theoretic measure of modern AI models

AI models have found applications across assistive driving, multi-modal generation, speech recognition, coding, math, predicted for massive automation



Gemini



Claude

GPT-3 and scaling laws [BMR+20]: One of the earliest findings is that large transformer models trained to predict next tokens on massive corpus show impressive adaptivity to downstream tasks

- **Generalization in over-parameterized models** [ZBH+21]: From a theoretical perspective, deep nets such as ResNet, BERT have more parameters than labels to memorize training dataset
- **The notion of entropy for measuring information in a language dates back to classical work by Claude Shannon** [Sha48]. Applying entropy to N -gram statistics provides one of the first information measure in text

Overarching goal

AI models have
generation, spe



GPT-3 and sc
transformer mc
impressive adap

A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist¹ and Hartley² on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one *selected from a set* of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design.

If the number of messages in the set is finite then this number or any monotonic function of this number can be regarded as a measure of the information produced when one message is chosen from the set, all choices being equally likely. As was pointed out by Hartley the most natural choice is the logarithmic function. Although this definition must be generalized considerably when we consider the influence of the statistics of the message and when we have a continuous range of messages, we will in all cases use an essentially logarithmic measure.

The logarithmic measure is more convenient for various reasons:

1. It is practically more useful. Parameters of engineering importance such as time, bandwidth, number of relays, etc., tend to vary linearly with the logarithm of the number of possibilities. For example, adding one relay to a group doubles the number of possible states of the relays. It adds 1 to the base 2 logarithm of this number. Doubling the time roughly squares the number of possible messages, or doubles the logarithm, etc.
2. It is nearer to our intuitive feeling as to the proper measure. This is closely related to (1) since we intuitively measures entities by linear comparison with common standards. One feels, for example, that two punched cards should have twice the capacity of one for information storage, and two identical channels twice the capacity of one for transmitting information.
3. It is mathematically more suitable. Many of the limiting operations are simple in terms of the logarithm but would require clumsy restatement in terms of the number of possibilities.

The choice of a logarithmic base corresponds to the choice of a unit for measuring information. If the base 2 is used the resulting units may be called binary digits, or more briefly *bits*, a word suggested by J. W. Tukey. A device with two stable positions, such as a relay or a flip-flop circuit, can store one bit of information. N such devices can store N bits, since the total number of possible states is 2^N and $\log_2 2^N = N$. If the base 10 is used the units may be called decimal digits. Since

$$\begin{aligned}\log_2 M &= \log_{10} M / \log_{10} 2 \\ &= 3.32 \log_{10} M,\end{aligned}$$

¹Nyquist, H., "Certain Factors Affecting Telegraph Speed," *Bell System Technical Journal*, April 1924, p. 324; "Certain Topics in Telegraph Transmission Theory," *A.I.E.E. Trans.*, v. 47, April 1928, p. 617.

²Hartley, R. V. L., "Transmission of Information," *Bell System Technical Journal*, July 1928, p. 535.

ern AI models

modal
ive automation



s is that large
pus show

Part I: A Hessian view of generalization in supervised fine-tuning (SFT)

Problem setup: given a hypothesis space of neural nets \mathcal{F}_W and a fine-tuning range, how do we quantify generalization gap $L(f_W) - \hat{L}(f_W)$?

- **Training loss:** given n samples from distribution D , $(x_1; y_1); \dots; (x_n; y_n)$,
$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^n \ell(f_W(x_i); y_i)$$
- **Test loss:** expected risk over D , $L(f_W) = \mathbb{E}_{(x;y) \sim D} [\ell(f_W(x); y)]$

Related literature

- **Rademacher complexity and uniform convergence** [BFT17]
- **Neural tangent kernels:** Characterize neural net features with a kernel map [ADH+19]
- **Implicit regularization:** Implicit bias of the learning algorithm [Var23] (e.g., driven by a special initialization [LMZ18])
- **Benign overfitting** and double-descent [BHM+19]: Random-matrix characterization of min-norm interpolators [BLL+20]
- See textbook by Tong Zhang [Zha23] (UIUC) for references

This talk: A Hessian view of generalization in SFT [JLZ22; JLS+23; ZLJ24]

Part I: A Hessian view of generalization in supervised fine-tuning (SFT)

Problem setup: given a hypothesis space of neural nets \mathcal{F}_W and a fine-tuning range, how do we quantify generalization gap $L(f_W) - \hat{L}(f_W)$?

- **Training loss:** given n samples from distribution D , $(x_1; y_1); \dots; (x_n; y_n)$,
$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^n \ell(f_W(x_i); y_i)$$
- **Test loss:** expected risk over D , $L(f_W) = \mathbb{E}_{(x;y) \sim D} [\ell(f_W(x); y)]$

Related literature

- **Rademacher complexity and uniform convergence** [BFT17]
- **Neural tangent kernels:** Characterize neural net features with a kernel map [ADH+19]
- **Implicit regularization:** Implicit bias of the learning algorithm [Var23] (e.g., driven by a special initialization [LMZ18])
- **Benign overfitting** and double-descent [BHM+19]: Random-matrix characterization of min-norm interpolators [BLL+20]
- See textbook by Tong Zhang [Zha23] (UIUC) for references

This talk: A Hessian view of generalization in SFT [JLZ22; JLS+23; ZLJ24]

Part I: A Hessian view of generalization in supervised fine-tuning (SFT)

Problem setup: given a hypothesis space of neural nets \mathcal{F}_W and a fine-tuning range, how do we quantify generalization gap $L(f_W) - \hat{L}(f_W)$?

- **Training loss:** given n samples from distribution D , $(x_1; y_1); \dots; (x_n; y_n)$,
$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^n \ell(f_W(x_i); y_i)$$
- **Test loss:** expected risk over D , $L(f_W) = \mathbb{E}_{(x;y) \sim D} [\ell(f_W(x); y)]$

Related literature

- **Rademacher complexity and uniform convergence** [BFT17]
- **Neural tangent kernels:** Characterize neural net features with a kernel map [ADH+19]
- **Implicit regularization:** Implicit bias of the learning algorithm [Var23] (e.g., driven by a special initialization [LMZ18])
- **Benign overfitting** and double-descent [BHM+19]: Random-matrix characterization of min-norm interpolators [BLL+20]
- See textbook by Tong Zhang [Zha23] (UIUC) for references

This talk: A Hessian view of generalization in SFT [JLZ22; JLS+23; ZLJ24]

Part I: A Hessian view of generalization in supervised fine-tuning (SFT)

Problem setup: given a hypothesis space of neural nets \mathcal{F}_W and a fine-tuning range, how do we quantify generalization gap $L(f_W) - \hat{L}(f_W)$?

- **Training loss:** given n samples from distribution D , $(x_1; y_1); \dots; (x_n; y_n)$,
$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^n \ell(f_W(x_i); y_i)$$
- **Test loss:** expected risk over D , $L(f_W) = \mathbb{E}_{(x;y) \sim D} [\ell(f_W(x); y)]$

Related literature

- **Rademacher complexity and uniform convergence** [BFT17]
- **Neural tangent kernels:** Characterize neural net features with a kernel map [ADH+19]
- **Implicit regularization:** Implicit bias of the learning algorithm [Var23] (e.g., driven by a special initialization [LMZ18])
- **Benign overfitting** and double-descent [BHM+19]: Random-matrix characterization of min-norm interpolators [BLL+20]
- See textbook by Tong Zhang [Zha23] (UIUC) for references

This talk: A Hessian view of generalization in SFT [JLZ22; JLS+23; ZLJ24]

Part I: A Hessian view of generalization in supervised fine-tuning (SFT)

Problem setup: given a hypothesis space of neural nets \mathcal{F}_W and a fine-tuning range, how do we quantify generalization gap $L(f_W) - \hat{L}(f_W)$?

- **Training loss:** given n samples from distribution D , $(x_1; y_1); \dots; (x_n; y_n)$,
$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^n \ell(f_W(x_i); y_i)$$
- **Test loss:** expected risk over D , $L(f_W) = \mathbb{E}_{(x;y) \sim D} [\ell(f_W(x); y)]$

Related literature

- **Rademacher complexity and uniform convergence** [BFT17]
- **Neural tangent kernels:** Characterize neural net features with a kernel map [ADH+19]
- **Implicit regularization:** Implicit bias of the learning algorithm [Var23] (e.g., driven by a special initialization [LMZ18])
- **Benign overfitting** and double-descent [BHM+19]: Random-matrix characterization of min-norm interpolators [BLL+20]
- See textbook by Tong Zhang [Zha23] (UIUC) for references

This talk: A Hessian view of generalization in SFT [JLZ22; JLS+23; ZLJ24]

Part I: A Hessian view of generalization in supervised fine-tuning (SFT)

Problem setup: given a hypothesis space of neural nets \mathcal{F}_W and a fine-tuning range, how do we quantify generalization gap $L(f_W) - \hat{L}(f_W)$?

- **Training loss:** given n samples from distribution D , $(x_1; y_1); \dots; (x_n; y_n)$,
$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^n \ell(f_W(x_i); y_i)$$
- **Test loss:** expected risk over D , $L(f_W) = \mathbb{E}_{(x;y) \sim D} [\ell(f_W(x); y)]$

Related literature

- **Rademacher complexity and uniform convergence** [BFT17]
- **Neural tangent kernels:** Characterize neural net features with a kernel map [ADH+19]
- **Implicit regularization:** Implicit bias of the learning algorithm [Var23] (e.g., driven by a special initialization [LMZ18])
- **Benign overfitting** and double-descent [BHM+19]: Random-matrix characterization of min-norm interpolators [BLL+20]
- See textbook by Tong Zhang [Zha23] (UIUC) for references

This talk: A Hessian view of generalization in SFT [JLZ22; JLS+23; ZLJ24]

Part I: A Hessian view of generalization in supervised fine-tuning (SFT)

Problem setup: given a hypothesis space of neural nets \mathcal{F}_W and a fine-tuning range, how do we quantify generalization gap $L(f_W) - \hat{L}(f_W)$?

- **Training loss:** given n samples from distribution D , $(x_1; y_1); \dots; (x_n; y_n)$,
$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^n \ell(f_W(x_i); y_i)$$
- **Test loss:** expected risk over D , $L(f_W) = \mathbb{E}_{(x;y) \sim D} [\ell(f_W(x); y)]$

Related literature

- **Rademacher complexity and uniform convergence** [BFT17]
- **Neural tangent kernels:** Characterize neural net features with a kernel map [ADH+19]
- **Implicit regularization:** Implicit bias of the learning algorithm [Var23] (e.g., driven by a special initialization [LMZ18])
- **Benign overfitting** and double-descent [BHM+19]: Random-matrix characterization of min-norm interpolators [BLL+20]
- See textbook by Tong Zhang [Zha23] (UIUC) for references

This talk: A Hessian view of generalization in SFT [JLZ22; JLS+23; ZLJ24]

Parts II-III: Applying this Hessian view to model interpretability, LLMs, RL, and beyond

RL (reasoning) vs. SFT: DeepSeek-R1 team [GYZ+25] argues that RL allows models to discover potential reasoning paths beyond SFT memorization (see also a theoretical perspective by [WSS+25])

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirog Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z.F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J.L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuan Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaoqin Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R.J. Chen, R.L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S.S. Li et al. (100 additional authors not shown)

General reasoning represents a long-standing and formidable challenge in artificial intelligence. Recent breakthroughs, exemplified by large language models (LLMs) and chain-of-thought prompting, have achieved considerable success on foundational reasoning tasks. However, this success is heavily contingent upon extensive human-annotated demonstrations, and models' capabilities are still insufficient for more complex problems. Here we show that the reasoning abilities of LLMs can be incentivized through pure reinforcement learning (RL), obviating the need for human-labeled reasoning trajectories. The proposed RL framework facilitates the emergent development of advanced reasoning patterns, such as self-reflection, verification, and dynamic strategy adaptation. Consequently, the trained model achieves superior performance on verifiable tasks such as mathematics, coding competitions, and STEM fields, surpassing its counterparts trained via conventional supervised learning on human demonstrations. Moreover, the emergent reasoning patterns exhibited by these large-scale models can be systematically harnessed to guide and enhance the reasoning capabilities of smaller models.

Subjects: **Computation and Language (cs.CL)**; Artificial Intelligence (cs.AI); Machine Learning (cs.LG)

Cite as: [arXiv:2501.12948 \[cs.CL\]](https://arxiv.org/abs/2501.12948)

(or [arXiv:2501.12948v2 \[cs.CL\]](https://arxiv.org/abs/2501.12948v2) for this version)

<https://doi.org/10.48550/arXiv.2501.12948> 

Journal reference: Nature volume 645, pages 633–638 (2025)

Related DOI: <https://doi.org/10.1038/s41586-025-09422-z> 

Submission history

From: Wenfeng Liang [[view email](mailto:wliang@deepseek.com)]

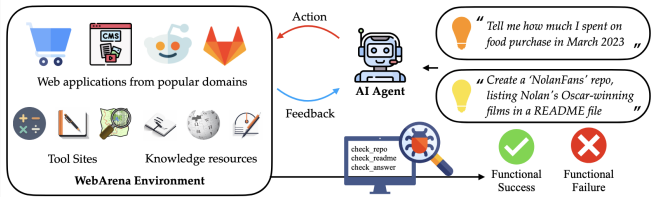
[v1] Wed, 22 Jan 2025 15:19:35 UTC (928 KB)

[v2] Sun, 4 Jan 2026 03:57:36 UTC (1,562 KB)

Parts II-III: Applying this Hessian view to model interpretability, LLMs, RL, and beyond

RL (reasoning) vs. SFT: DeepSeek-R1 team [GYZ+25] argues that RL allows models to discover potential reasoning paths beyond SFT memorization (see also a theoretical perspective by [WSS+25])

- **Connection to RL: Performing agentic tasks** (cf. WebArena [ZXZ+23])

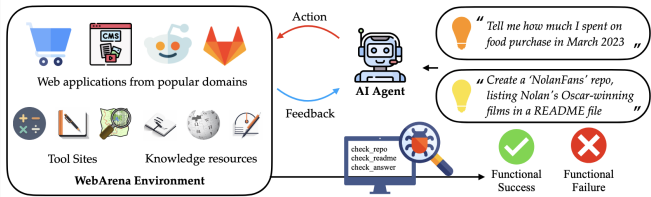


- **Strategyproof RLHF:** Ensure safety in online platforms and data vendors

Parts II-III: Applying this Hessian view to model interpretability, LLMs, RL, and beyond

RL (reasoning) vs. SFT: DeepSeek-R1 team [GYZ+25] argues that RL allows models to discover potential reasoning paths beyond SFT memorization (see also a theoretical perspective by [WSS+25])

- **Connection to RL: Performing agentic tasks** (cf. WebArena [ZXZ+23])

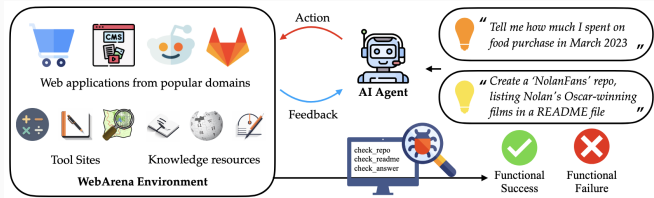


- **Strategyproof RLHF:** Ensure safety in online platforms and data vendors

Parts II-III: Applying this Hessian view to model interpretability, LLMs, RL, and beyond

RL (reasoning) vs. SFT: DeepSeek-R1 team [GYZ+25] argues that RL allows models to discover potential reasoning paths beyond SFT memorization (see also a theoretical perspective by [WSS+25])

- **Connection to RL: Performing agentic tasks** (cf. WebArena [ZXZ+23])

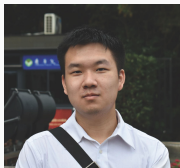


- **Strategyproof RLHF:** Ensure safety in online platforms and data vendors

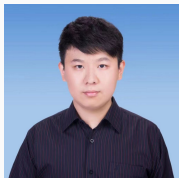
Present a Hessian-guided approximation approach to study task attribution [ZDZ26], LLMs [ZZL+25b], RL optimization [ZDY+26], and strategyproofness

Robust fine-tuning by measuring the Hessian spectrum

ICML 2022 [[JLZ22](#)]; TMLR 2024 [[ZLJ24](#)]; NeurIPS Workshop 2025 [[ZZL+25a](#)]



Zhenshuo Zhang



Haotian Ju



Dongyue Li

Supervised prediction

Problem setup: Given n training samples, an l -layer deep net outputs a representation such as

$$f_w(x) = \sigma_l \circ \sigma_{l-1} \circ \dots \circ \sigma_3 \circ W_3 \circ \sigma_2 \circ W_2 \circ \sigma_1(W_1 x)$$

SFT via a PAC-Bayes analysis [JLZ22]

$P = \mathcal{N}(W^{(0)}; \sigma^2 \text{Id})$: prior distribution centered at initialization weights
Pretrained model (e.g., Qwen3-8B, Llama-3.2-3B, DeepSeek-R1)

• $Q = \mathcal{N}(W^{(T)}; \sigma^2 \text{Id})$: posterior distribution centered at trained weights /
Fine-tuned model (e.g., via chain-of-thought, DPO alignment)

• PAC-Bayes bound: With probability at least $1 - \delta$ for any $\delta \in (0, 1)$

$$\mathbb{E}_w \mathbb{E}_O [L(f_w)] \leq \frac{1}{n} \mathbb{E}_w \mathbb{E}_O [\hat{L}(f_w)] + \frac{KL(Q||P) + \log(\frac{1}{\delta})}{2(1-\delta)n} \quad (1)$$

Proof via moment-generating functions and variational inequalities

Interpretation: If the fine-tuned model does not move the weights too far from their initialization, it is less prone to overfit, and this is captured by the KL

Supervised prediction

Problem setup: Given n training samples, an l -layer deep net outputs a representation such as

$$f_W(x) = \dots \circ W_3 \circ W_2 \circ W_1(x)$$

SFT via a PAC-Bayes analysis [JLZ22]

- $P = N(W^{(0)}; \sigma^2 \text{Id})$: *prior* distribution centered at initialization weights / **Pretrained model** (e.g., Qwen3-8B, Llama-3.2-3B, DeepSeek-R1)
- $Q = N(W^{(T)}; \sigma^2 \text{Id})$: *posterior* distribution centered at trained weights / **Fine-tuned model** (e.g., via chain-of-thought, DPO alignment)
- **PAC-Bayes bound:** With probability at least $1 - \delta$ for any $\beta \in (0, 1)$

$$\mathbb{E}_W [L(f_W)] \leq \frac{1}{n} \mathbb{E}_Q [\hat{L}(f_W)] + \frac{KL(Q||P) + \log(\frac{1}{\delta})}{2\beta(1-\beta)n} \quad (1)$$

Proof via moment-generating functions and variational inequalities

Interpretation: If the fine-tuned model does not move the weights too far from their initialization, it is less prone to overfit, and this is captured by the KL

Supervised prediction

Problem setup: Given n training samples, an l -layer deep net outputs a representation such as

$$f_W(x) = \sigma_3 \circ W_3 \circ \sigma_2 \circ W_2 \circ \sigma_1 \circ W_1 x$$

SFT via a PAC-Bayes analysis [JLZ22]

- $P = N(W^{(0)}; \sigma^2 \text{Id})$: *prior* distribution centered at initialization weights $W^{(0)}$ / **Pretrained model** (e.g., Qwen3-8B, Llama-3.2-3B, DeepSeek-R1)
- $Q = N(W^{(T)}; \sigma^2 \text{Id})$: *posterior* distribution centered at trained weights $W^{(T)}$ / **Fine-tuned model** (e.g., via chain-of-thought, DPO alignment)
- **PAC-Bayes bound:** With probability at least $1 - \delta$ for any $\delta \in (0, 1)$

$$\mathbb{E}_W [L(f_W)] \leq \frac{1}{n} \mathbb{E}_W [L(f_W)] + \frac{KL(Q||P) + \log(1/\delta)}{2(1-\delta)n} \quad (1)$$

Proof via moment-generating functions and variational inequalities

Interpretation: If the fine-tuned model does not move the weights too far from their initialization, it is less prone to overfit, and this is captured by the KL

Supervised prediction

Problem setup: Given n training samples, an l -layer deep net outputs a representation such as

$$f_w(x) = \sigma_l \circ \sigma_{l-1} \circ \dots \circ \sigma_3 \circ W_3 \circ \sigma_2 \circ W_2 \circ \sigma_1(W_1 x)$$

SFT via a PAC-Bayes analysis [JLZ22]

- $P = N(W^{(0)}; \sigma^2 \text{Id})$: *prior* distribution centered at initialization weights $W^{(0)}$ / **Pretrained model** (e.g., Qwen3-8B, Llama-3.2-3B, DeepSeek-R1)
- $Q = N(W^{(T)}; \sigma^2 \text{Id})$: *posterior* distribution centered at trained weights $W^{(T)}$ / **Fine-tuned model** (e.g., via chain-of-thought, DPO alignment)
- **PAC-Bayes bound:** With probability at least $1 - \delta$ for any $\delta \in (0, 1)$

$$\mathbb{E}_w \mathbb{E}_O [L(f_w)] \leq \frac{1}{n} \mathbb{E}_w \mathbb{E}_O [\hat{L}(f_w)] + \frac{KL(Q||P) + \log(1/\delta)}{2(1-\delta)n} \quad (1)$$

Proof via moment-generating functions and variational inequalities

Interpretation: If the fine-tuned model does not move the weights too far from their initialization, it is less prone to overfit, and this is captured by the KL

Supervised prediction

Problem setup: Given n training samples, an l -layer deep net outputs a representation such as

$$f_w(x) = \sigma_l \circ \sigma_{l-1} \circ \dots \circ \sigma_3 \circ W_3 \circ \sigma_2 \circ W_2 \circ \sigma_1 \circ W_1 x$$

SFT via a PAC-Bayes analysis [JLZ22]

- $P = N(W^{(0)}; \sigma^2 \text{Id})$: *prior* distribution centered at initialization weights $W^{(0)}$ / **Pretrained model** (e.g., Qwen3-8B, Llama-3.2-3B, DeepSeek-R1)
- $Q = N(W^{(T)}; \sigma^2 \text{Id})$: *posterior* distribution centered at trained weights $W^{(T)}$ / **Fine-tuned model** (e.g., via chain-of-thought, DPO alignment)
- **PAC-Bayes bound:** With probability at least $1 - \delta$ for any $\delta \in (0, 1)$

$$\mathbb{E}_w \mathbb{E}_o [L(f_w)] \leq \frac{1}{n} \mathbb{E}_w \mathbb{E}_o [\hat{L}(f_w)] + \frac{KL(Q||P) + \log(1/\delta)}{2(1-\delta)n} \quad (1)$$

Proof via moment-generating functions and variational inequalities

Interpretation: If the fine-tuned model does not move the weights too far from their initialization, it is less prone to overfit, and this is captured by the KL

Illustrating the Hessian approximation

Taylor's expansion of the perturbed loss

Let

$$\ell_Q(f_W) = \mathbb{E}_W \ell_Q[f_W] = \mathbb{E}_U \mathbb{E}_{N(0; \sigma^2 \text{Id})} [\ell(f_{W+U})]$$

We have

$$\ell_Q = \mathbb{E}_U [\ell(f_{W+U})] = \ell(f_W) + \frac{\sigma^2}{2} \text{Tr} \left\{ \underbrace{r^2}_{\text{Hessian regularization}} \underbrace{\ell''(f_W)}_i \right\} + O(\sigma^3) \quad (2)$$

Illustrating the Hessian approximation

Taylor's expansion of the perturbed loss

Let

$$\mathbb{E}_W [\mathbb{E}_U [\ell(f_{W+U})]] = \mathbb{E}_U [\mathbb{E}_W [\ell(f_{W+U})]]$$

We have

$$\mathbb{E}_U [\mathbb{E}_W [\ell(f_{W+U})]] = \ell(f_W) + \frac{\sigma^2}{2} \text{Tr} \{ \underbrace{H}_{\text{Hessian regularization}} \ell(f_W) \} + O(\sigma^3) \quad (2)$$

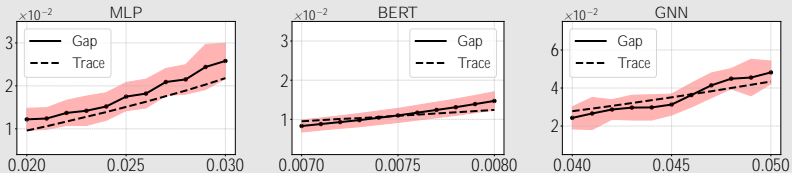


Figure 1: Illustration of the Hessian approximation in equation (2), measured on various fine-tuned networks. σ : standard deviation of Gaussian noise injected

Non-vacuous generalization bounds by measuring the Hessian trace

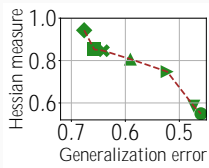
Uniform convergence of the Hessian operator

Assuming the Hessian operator $r^{-2} \cdot (f_W(\cdot); \cdot)$ is C -Lipschitz-continuous on W for some constant $C > 0$, the trace operator satisfies uniform convergence within a bounded hypothesis space

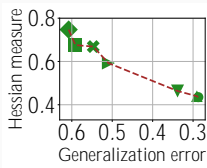
Non-vacuous generalization bounds by measuring the Hessian trace

Uniform convergence of the Hessian operator

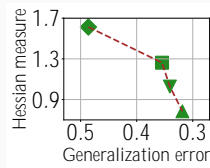
Assuming the Hessian operator $r^{-2}(f_W(\cdot); \cdot)$ is C -Lipschitz-continuous on \mathcal{W} for some constant $C > 0$, the trace operator satisfies uniform convergence within a bounded hypothesis space



(a) ResNet-50



(b) ResNet-50



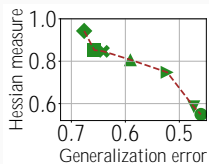
(c) BERT-Base

Figure 2: The Hessian-trace measure accurately correlates with empirical generalization errors for seven fine-tuning methods, including early stopping, ℓ_2 weight decay, label smoothing, mixup, sharpness-aware minimization

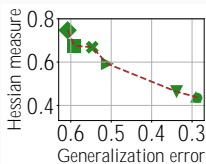
Non-vacuous generalization bounds by measuring the Hessian trace

Uniform convergence of the Hessian operator

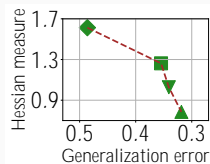
Assuming the Hessian operator $r^{-2} \cdot (f_W(\cdot); \cdot)$ is C -Lipschitz-continuous on W for some constant $C > 0$, the trace operator satisfies uniform convergence within a bounded hypothesis space



(a) ResNet-50



(b) ResNet-50



(c) BERT-Base

Figure 2: The Hessian-trace measure accurately correlates with empirical generalization errors for seven fine-tuning methods, including early stopping, ℓ_2 weight decay, label smoothing, mixup, sharpness-aware minimization

Similar results and observations also hold for chain-of-thought fine-tuning!

Applying Hessian regularization to mitigate grokking

Grokking is a delayed generalization phenomenon in modular arithmetic tasks, such as $a + b \pmod{p}$

- A model (e.g., a transformer) first achieves 100% training accuracy, while test accuracy remains low
- After many more training epochs, test accuracy increases

Hessian regularization via noise injection: Instead of minimizing the loss \mathcal{L} , minimize the perturbed loss

$$\tilde{\mathcal{L}} = \mathbb{E}_U \mathcal{L}(f_{W+U});$$

leading to the following SGD update

$$W \leftarrow W - \eta \nabla_W \tilde{\mathcal{L}}(f_{W+U}(x); y)$$

Applying Hessian regularization to mitigate grokking

Grokking is a delayed generalization phenomenon in modular arithmetic tasks, such as $a + b \pmod p$

- A model (e.g., a transformer) first achieves 100% training accuracy, while test accuracy remains low
- After many more training epochs, test accuracy increases

Hessian regularization via noise injection: Instead of minimizing the loss \mathcal{L} , minimize the perturbed loss

$$\mathcal{L}_\sigma = \mathbb{E}_{U \sim \mathcal{N}(0; \sigma^2 \text{Id})} [\mathcal{L}(f_{W+U})];$$

leading to the following SGD update

$$W \leftarrow W - \eta \nabla_W \mathcal{L}(f_{W+U}(x); y)$$

Illustration of grokking in modular arithmetic

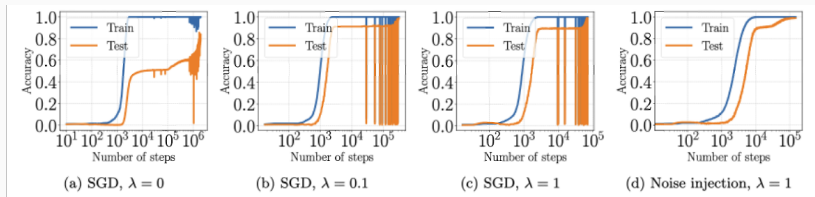


Figure 3: Comparing the training curves of transformer models for modular arithmetic tasks using weight decay (with varying levels indicated by λ), and Hessian regularization (via noise injection)

Finding 1: Hessian regularization stabilizes the training curves and reduces the number of grokking steps

Practical implication: Capturing invariance in math reasoning

Finding II: The loss surface becomes exceptionally flat (low Hessian values) specifically along label-invariant directions

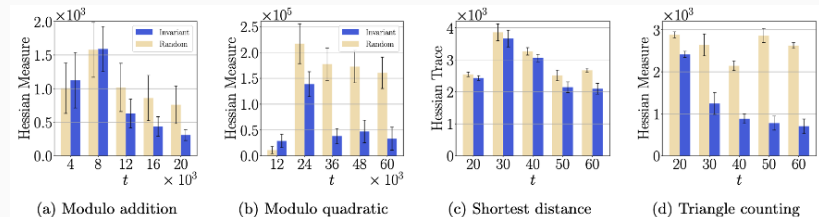
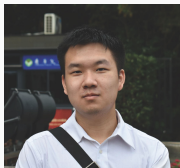


Figure 4: Model generalizes with Hessian regularization. The loss surface becomes flatter along label-invariant directions (blue bars) than random directions (yellow bars)

Invariant direction: for an input $x = (a; b)$ in $a + b \pmod p$, a transformation $g(x) = (a + k) + (b - k) \pmod p$, an invariant direction is given by $g(x) - x$

Kernel methods for LLMs and task attribution

EMNLP 2025 [[ZZL+25b](#)]; ICLR 2026 [[ZDZ26](#)]



Zhenshuo Zhang



Ziniu Zhang

Minxuan Duan

Problem setup

Designing kernel models to scale these insights to LLMs and task attribution

Task attribution: quantifying the influence of individual training tasks on model performance

In-context learning

Given k examples and a query, $((x_1, y_1), \dots, (x_k, y_k), x_{\text{Query}})$, output y_{Query}

$$\left\{ \underbrace{(1 + 2; 3); (2 + 4; 6); \dots; (1 + 4; 5); (3 + 4; ?)}_A \right\} / \left\{ \underbrace{Z}_B \right\}$$

- Query set: $S^{\text{Query}} = \prod_{i=1}^n (x_i^{\text{Query}}; y_i^{\text{Query}})^{O_n^{\text{Query}}}$
- Demonstration set: $S^{\text{Demo}} = \prod_{i=1}^{n^{\text{Demo}}} (x_i^{\text{Demo}}; y_i^{\text{Demo}})^{n^{\text{Demo}}}$

How does adding or removing one example in the prompt affect the LLM's answer?

Problem setup

Designing kernel models to scale these insights to LLMs and task attribution

Task attribution: quantifying the influence of individual training tasks on model performance

In-context learning

Given k examples and a query, $((x_1; y_1); \dots; (x_k; y_k); x_{\text{query}})$, output y_{query}

$$\left\{ \underbrace{(1 + 2; 3); (2 + 4; 6); \dots; (1 + 4; 5); (3 + 4; ?)}_A \right\} \quad ! \quad \left\{ \underbrace{Z}_B \right\}$$

- Query set: $S^{\text{Query}} = \prod_{i=1}^n (x_i^{\text{Query}}; y_i^{\text{Query}}) \quad O_n^{\text{Query}}$
- Demonstration set: $S^{\text{Demo}} = \prod_{i=1}^n (x_i^{\text{Demo}}; y_i^{\text{Demo}}) \quad n^{\text{Demo}}$

How does adding or removing one example in the prompt affect the LLM's answer?

Problem setup

Designing kernel models to scale these insights to LLMs and task attribution

Task attribution: quantifying the influence of individual training tasks on model performance

In-context learning

Given k examples and a query, $((x_1; y_1); \dots; (x_k; y_k); x_{\text{query}})$, output y_{query}

$$\left\{ \underbrace{(1 + 2; 3); (2 + 4; 6); \dots; (1 + 4; 5); (3 + 4; ?)}_A \right\} \rightarrow \left\{ \underbrace{7}_B \right\}$$

- Query set: $S^{\text{Query}} = \prod_{i=1}^n (x_i^{\text{Query}}; y_i^{\text{Query}}) \quad O_n^{\text{Query}}$
- Demonstration set: $S^{\text{Demo}} = \prod_{i=1}^n (x_i^{\text{Demo}}; y_i^{\text{Demo}}) \quad n^{\text{Demo}}$

How does adding or removing one example in the prompt affect the LLM's answer?

Problem setup

Designing kernel models to scale these insights to LLMs and task attribution

Task attribution: quantifying the influence of individual training tasks on model performance

In-context learning

Given k examples and a query, $((x_1; y_1); \dots; (x_k; y_k); x_{\text{query}})$, output y_{query}

$$\left\{ \underbrace{(1 + 2; 3); (2 + 4; 6); \dots; (1 + 4; 5); (3 + 4; ?)}_A \right\} ! \quad \left| \underbrace{Z}_B \right\}$$

- Query set: $S^{\text{Query}} = \prod_{i=1}^n (x_i^{\text{Query}}; y_i^{\text{Query}})^{O_n^{\text{Query}}}$
- Demonstration set: $S^{\text{Demo}} = \prod_{i=1}^{n^{\text{Demo}}} (x_i^{\text{Demo}}; y_i^{\text{Demo}})$

How does adding or removing one example in the prompt affect the LLM's answer?

Methods

Leave-one-out (LOO): To compute the LOO scores for K samples (or tasks), we need to perform inference (or train models) for $K + 1$ times

Influence functions (IFs) [KL17]: Alternatively, we could train models once, and then compute the IFs

- Let $\ell_i(f_W)$ denote the loss for the i -th sample for a model f_W , the influence function measures how adding/removing one sample affects ℓ as

$$\ell_i = \frac{\partial \ell(f_W)}{\partial W} \Big|_{W^*} \cdot \frac{\partial W^*}{\partial \ell_i} \approx \frac{\partial \ell(f_W)}{\partial W} \Big|_{W^*} \cdot \frac{\partial W^*}{\partial \ell_i} \Big|_{W^*}$$

- Computing ℓ_1, \dots, ℓ_K again requires repeated Hessian-inverse approximations

Data attribution [IPE+22; PGI+23]: Instead, estimate a surrogate model or function from every subset of tasks to their joint training outcomes

Methods

Leave-one-out (LOO): To compute the LOO scores for K samples (or tasks), we need to perform inference (or train models) for $K + 1$ times

Influence functions (IFs) [KL17]: Alternatively, we could train models once, and then compute the IFs

- Let $\ell_i(f_W)$ denote the loss for the i -th sample for a model f_W , the influence function measures how adding/removing one sample affects ℓ as

$$I_i = \frac{\partial \ell_i(f_W)}{\partial W} \Big|_{W^*} = \frac{\partial \ell_i(f_W)}{\partial W} \Big|_{W^*} - \frac{\partial \ell(f_W)}{\partial W} \Big|_{W^*}; \quad \forall i = 1, \dots, K$$

- Computing I_1, \dots, I_K again requires repeated Hessian-inverse approximations

Data attribution [IPE+22; PGI+23]: Instead, estimate a surrogate model or function from every subset of tasks to their joint training outcomes

Methods

Leave-one-out (LOO): To compute the LOO scores for K samples (or tasks), we need to perform inference (or train models) for $K + 1$ times

Influence functions (IFs) [KL17]: Alternatively, we could train models once, and then compute the IFs

- Let $\ell_i(f_W)$ denote the loss for the i -th sample for a model f_W , the influence function measures how adding/removing one sample affects ℓ as

$$\ell_i = \frac{\partial \ell(f_W)}{\partial W} \Big|_{W^*} \cdot \frac{\partial W}{\partial \ell_i(f_W)} \Big|_{W^*}$$

- Computing ℓ_1, \dots, ℓ_K again requires repeated Hessian-inverse approximations

Data attribution [IPE+22; PGI+23]: Instead, estimate a surrogate model or function from every subset of tasks to their joint training outcomes

Leave-one-out (LOO): To compute the LOO scores for K samples (or tasks), we need to perform inference (or train models) for $K + 1$ times

Influence functions (IFs) [KL17]: Alternatively, we could train models once, and then compute the IFs

- Let $\ell_i(f_W)$ denote the loss for the i -th sample for a model f_W , the influence function measures how adding/removing one sample affects ℓ as

$$\ell_i = \frac{\partial \ell(f_W)}{\partial W} \Big|_{W^*} \cdot \frac{\partial W}{\partial \ell_i(f_W)} \Big|_{W^*}$$

- Computing ℓ_1, \dots, ℓ_K again requires repeated Hessian-inverse approximations

Data attribution [IPE+22; PGI+23]: Instead, estimate a surrogate model or function from every subset of tasks to their joint training outcomes

Linear surrogate models [LNZ23]

1. Let D denote a distribution of subsets supported on $\{0, 1\}^K$ such as the Binomial distribution with sampling probability p (e.g., $p = 1/K$). Sample n random subsets from D : $S_1; S_2; \dots; S_n$, evaluate their subset training outcomes as $\hat{y}(S_1); \hat{y}(S_2); \dots; \hat{y}(S_n)$

2. Let $\beta = (\beta_1; \dots; \beta_K) \in \mathbb{R}^K$ and define the linear surrogate function as

$$g(S) = \sum_{s \in S} \beta_s$$

3. Estimate $\hat{\beta}$ by minimizing the mean-squared reconstruction error

$$\frac{1}{n} \sum_{i=1}^n (g(S_i) - \hat{y}(S_i))^2 \quad (3)$$

4. $\hat{\beta}_i$ captures relevance between i (for every $1 \leq i \leq K$) and the test function

Linear surrogate models have shown superior performance in practice for measuring data/task influence, but their workings are not well-understood

Linear surrogate models [LNZ23]

1. Let D denote a distribution of subsets supported on $\{0, 1\}^K$ such as the Binomial distribution with sampling probability p (e.g., $p = 1/K$). Sample n random subsets from D : $S_1; S_2; \dots; S_n$, evaluate their subset training outcomes as $\hat{y}(S_1); \hat{y}(S_2); \dots; \hat{y}(S_n)$

2. Let $\beta = (\beta_1; \dots; \beta_K) \in \mathbb{R}^K$ and define the linear surrogate function as

$$g(S) = \sum_{s \in S} \beta_s$$

3. Estimate $\hat{\beta}$ by minimizing the mean-squared reconstruction error

$$\frac{1}{n} \sum_{i=1}^n (g(S_i) - \hat{y}(S_i))^2 \quad (3)$$

4. $\hat{\beta}_i$ captures relevance between i (for every $1 \leq i \leq K$) and the test function

Linear surrogate models have shown superior performance in practice for measuring data/task importance, but their workings are not well-understood

A unified weighting setup [ZDZ26]

$\hat{\cdot}$ Let $\mathbf{s} = [s_1; \dots; s_K]^T \in \mathbb{R}^{K \times 1}$, and $\hat{L}(f_W; \mathbf{s}) = \sum_{k=1}^K s_k \ell_k(f_W)$

$\hat{\cdot}$ First minimize over W , then evaluate the model on a test set, leading to a test metric $F(\mathbf{s})$

A uni ed weighting setup [ZDZ26]

^ Let $\mathbf{s} = [s_1; \dots; s_K]^T \in \mathbb{R}^{K \times 1}$, and $\hat{L}(f_W; \mathbf{s}) = \sum_{k=1}^K s_k \cdot k(f_W)$

^ First minimize over W , then evaluate the model on a test set, leading to a test metric $F(\mathbf{s})$

Linear surrogate models are approximately equal to in uence functions

^ $\hat{L}([l_1(s^?); l_2(s^?); \dots; l_K(s^?)]) = r_s F(s^?)$, with $s^? = [1]_K$

^ ^ Linear surrogate model coe cients (cf. equation (3))

W.h.p. over the randomness of $\{S_2; \dots; S_n\}$,

$$\hat{L} \approx O(r_s^2 F(s^?) \frac{1}{2} \frac{K}{n} + O(\frac{1}{n}))$$

which is near-zero when second-order interaction effects $\frac{\partial^2 F(s^?)}{\partial s^2} = 0$

Idea: apply delta method | a technique for approximating the distribution of a smooth function of an estimator using Taylor expansion

A unified weighting setup [ZDZ26]

$\hat{\cdot}$ Let $\mathbf{s} = [s_1; \dots; s_K]^T \in \mathbb{R}^{K \times 1}$, and $\hat{L}(f_W; \mathbf{s}) = \prod_{k=1}^K s_k \cdot k(f_W)$

$\hat{\cdot}$ First minimize over W , then evaluate the model on a test set, leading to a test metric $F(\mathbf{s})$

Linear surrogate models are approximately equal to inference functions

$\hat{\cdot}$ $\tau [l_1(s^?); l_2(s^?); \dots; l_K(s^?)]^T = r_s F(s^?)$, with $s^? = [1]_K$

$\hat{\cdot}$ Linear surrogate model coefficients (cf. equation (3))

W.h.p. over the randomness of $\{S_2; \dots; S_n\}$,

$$\hat{\cdot} \tau = O\left(r_s^2 F(s^?) \frac{1}{2} \frac{P}{K} + O\left(\frac{1}{n}\right)\right);$$

which is near-zero when second-order interaction effects $\frac{\partial^2 F(s^?)}{\partial s^2} = 0$

Idea: apply delta method | a technique for approximating the distribution of a smooth function of an estimator using Taylor expansion

Linear surrogate models do not capture nonlinear task interactions such as synergy, antagonism, XOR, or quadratic-type relationships

Kernel surrogate models: Regression coefficients $(\beta_i)_{i=1}^n$, kernel function $k : X \times X \rightarrow \mathbb{R}$ (e.g., radial basis function (RBF) kernel)

$$g(s) = \sum_{i=1}^n \beta_i k(s^{(i)}; s)$$

Linear surrogate models do not capture nonlinear task interactions such as synergy, antagonism, XOR, or quadratic-type relationships

Kernel surrogate models: Regression coefficients $(\beta_i)_{i=1}^n$, kernel function $k : X \times X \rightarrow \mathbb{R}$ (e.g., radial basis function (RBF) kernel)

$$g(\mathbf{s}) = \sum_{i=1}^n \beta_i k(\mathbf{s}^{(i)}; \mathbf{s})$$

Linear surrogate models do not capture nonlinear task interactions such as synergy, antagonism, XOR, or quadratic-type relationships

Kernel surrogate models: Regression coefficients $(\beta_i)_{i=1}^n$, kernel function $k : X \times X \rightarrow \mathbb{R}$ (e.g., radial basis function (RBF) kernel)

$$g(s) = \sum_{i=1}^n \beta_i k(s^{(i)}; s)$$

Residual error	Linear	RBF
CIFAR-10	4:4 0:9	1:0 0:0
Modular arithmetic	4:6 1:3	1:5 0:4
In-context learning	0:8 0:2	0:4 0:1
Multi-objective RL	0:2 0:1	0:1 0:1

Figure 5: Illustrate linear vs. kernel models on a decision boundary. Using the RBF kernel gives more accurate fit than linear surrogate models

Estimating kernel ridge coefficients is sample-expensive; Instead, apply a gradient-projection trick [ZYL+25b; ZDZ26]

- Pre-compute gradients and functional values at a pretrained model \hat{W}
- Only use pre-computed gradient features to estimate \hat{W}

Gradient estimation for binary classification

Minimize the log-loss of an input $(x; y)$ pair on a neural network f

$$(f_W(x); y) = \log(1 + \exp(-yf_W(x)))$$

2. Approximate $f_W(x)$ with first-order Taylor's expansion

$$\hat{f}_W = \log \left(1 + \exp \left(y \left(f_{\hat{W}}(x) + h \left[\frac{\partial f_{\hat{W}}(x)}{\partial W} \right] \right) \right) \right); W = \hat{W} + \eta \Delta W$$

func: value
precomputed grad:

3. Estimate W on the combined data $D(S)$ from all the tasks of S

$$\min_W \sum_{(x,y) \in D(S)} (f_W(x); y)$$

Estimating kernel ridge coefficients is sample-expensive; Instead, apply a gradient-projection trick [ZLL+25b; ZDZ26]

- Pre-compute gradients and functional values at a pretrained model \hat{W}
- Only use pre-computed gradient features to estimate \hat{W}_i

Gradient estimation for binary classification

- Minimize the log-loss of an input $(x; y)$ pair on a neural network f_W

$$\ell(f_W(x); y) = \log(1 + \exp(-yf_W(x)))$$

- Approximate $f_W(x)$ with first-order Taylor's expansion

$$\hat{\ell}(f_W) = \log \left(1 + \exp \left(y \left(f_{\hat{W}}(x) + h \left[\frac{\partial f_{\hat{W}}}{\partial z} \right]_{z=\hat{z}} \right) \right) \right); \quad W = \hat{W} + \Delta W$$

func: value
precomputed grad:

- Estimate W on the combined data $D(S)$ from all the tasks of S

$$\min_W \sum_{(x,y) \in D(S_i)} \ell(f_W(x); y)$$

Estimating kernel ridge coefficients is sample-expensive; Instead, apply a gradient-projection trick [ZLL+25b; ZDZ26]

- ^ Pre-compute gradients and functional values at a pretrained model \hat{W}
- ^ Only use pre-computed gradient features to estimate \hat{W}

Gradient estimation for binary classification

1. Minimize the log-loss of an input $(x; y)$ pair on a neural network f_W

$$\ell(f_W(x); y) = \log(1 + \exp(-yf_W(x)))$$

2. Approximate $f_W(x)$ with first-order Taylor's expansion

$$\hat{\ell}(f_W) = \log \left(1 + \exp \left(y \left(f_{\hat{W}}(x) + h \left[\frac{\partial f_{\hat{W}}(x)}{\partial W} \right] ; W = \hat{W} \right) \right) \right)$$

func: value
precomputed grad:

3. Estimate \hat{W} on the combined data $D(S)$ from all the tasks of S

$$\min_W \sum_{(x,y) \in D(S)} \ell(f_W(x); y)$$

Estimating kernel ridge coefficients is sample-expensive; Instead, apply a gradient-projection trick [ZZL+25b; ZDZ26]

- Pre-compute gradients and functional values at a pretrained model \hat{W}
- Only use pre-computed gradient features to estimate \hat{W}

Gradient estimation for binary classification

- Minimize the log-loss of an input $(x; y)$ pair on a neural network f_W

$$\ell(f_W(x); y) = \log(1 + \exp(-yf_W(x)))$$

- Approximate $f_W(x)$ with first-order Taylor's expansion

$$\hat{\ell}(f_W) = \log \left(1 + \exp \left(y \left(f_{\hat{W}}(x) + h \left[\frac{\partial f_{\hat{W}}(x)}{\partial W} \right] ; W = \hat{W} \right) \right) \right)$$

func: value
precomputed grad:

- Estimate W on the combined data $D(S)$ from all the tasks of S

$$\min_W \sum_{(x,y) \in D(S)} \ell(f_W(x); y)$$

Multi-objective, strategyproof reinforcement learning from human feedback

AAAI 2026 [[ZDY+26](#)] (+working papers)

Zhenshuo Zhang

Minxuan Duan

Problem setup

Trajectory-level data arise from a variety of applications in robotics, control, RL reasoning, RLHF, LLM agents, digital platforms

Problem setup: Multi-objective Markov decision process (MOMDP) $T = (\mathcal{S}; \mathcal{A}; P; P_0; r; \gamma)$. There are m objectives given by $r \in \mathbb{R}^m$, who share the same state space \mathcal{S} and action space \mathcal{A} but different reward feedback. A trajectory consists of

$$\tau^{(t)} = (s_0^{(t)}; a_0^{(t)}; r_0^{(t)}; \dots; s_H^{(t)}; a_H^{(t)}; r_H^{(t)})$$

Problem setup

Trajectory-level data arise from a variety of applications in robotics, control, RL reasoning, RLHF, LLM agents, digital platforms

Problem setup: Multi-objective Markov decision process (MOMDP)

$T = (\mathcal{S}; \mathcal{A}; P; P_0; r; \gamma)$. There are m objectives given by $r \in \mathbb{R}^m$, who share the same state space \mathcal{S} and action space \mathcal{A} but different reward feedback. A trajectory consists of

$$\tau^{(t)} = (s_0^{(t)}; a_0^{(t)}; r_0^{(t)}; \dots; s_H^{(t)}; a_H^{(t)}; r_H^{(t)})$$

Figure 6: Visualize the objective space returns $(r_1; r_2)$ for policies optimized under varying preference weights $w \in [0; 1]$ in three canonical regimes: Pareto frontier of clustered vs. conflicting objectives.

Problem setup

Trajectory-level data arise from a variety of applications in robotics, control, RL reasoning, RLHF, LLM agents, digital platforms

Problem setup: Multi-objective Markov decision process (MOMDP)

$T = (\mathcal{S}; \mathcal{A}; P; P_0; r; \gamma)$. There are m objectives given by $r \in \mathbb{R}^m$, who share the same state space \mathcal{S} and action space \mathcal{A} but different reward feedback. A trajectory consists of

$$\tau^{(t)} = (s_0^{(t)}; a_0^{(t)}; r_0^{(t)}; \dots; s_H^{(t)}; a_H^{(t)}; r_H^{(t)})$$

Figure 6: Visualize the objective space returns $(r_1; r_2)$ for policies optimized under varying preference weights $w \in [0; 1]$ in three canonical regimes: Pareto frontier of clustered vs. conflicting objectives.

Applications: Multi-objective optimization, adversarially-robust planning

1. The optimal (ensemble) policy follows a routing/MoE mechanism

1. **The optimal (ensemble) policy follows a routing/MoE mechanism**
2. Strategic preference reporting: Can agents bias feedback to steer the policy? Solution: Tukey Pessimistic median mechanism

Figure 7: Relative sub-optimality for the Tukey pessimistic median mechanism

1. **The optimal (ensemble) policy follows a routing/MoE mechanism**
2. Strategic preference reporting: Can agents bias feedback to steer the policy? Solution: Tukey Pessimistic median mechanism

Figure 7: Relative sub-optimality for the Tukey pessimistic median mechanism

3. **Agentic work ows:** Better explainability for black-box model behaviors
4. **Preference optimization:** Reconcile multi-modal preferences for better alignment?

Desiderata: an information-theoretic measure of modern AI models

- ^ By measuring spectral statistics of the loss landscape Hessian, can explain numerous empirical phenomena related to SFT [JLZ22; ZLJ24], grokking [ZZL+25a], and more

Desiderata: an information-theoretic measure of modern AI models

- ^ By measuring spectral statistics of the loss landscape Hessian, can explain numerous empirical phenomena related to SFT [JLZ22; ZLJ24], grokking [ZZL+25a], and more
- ^ New insights and connections on LLM inference [ZZL+25b], model interpretability [ZDZ26], RL [ZDY+26], through the Hessian lens

Desiderata: an information-theoretic measure of modern AI models

- ^ By measuring spectral statistics of the loss landscape Hessian, can explain numerous empirical phenomena related to SFT [JLZ22; ZLJ24], grokking [ZZL+25a], and more
- ^ New insights and connections on LLM inference [ZZL+25b], model interpretability [ZDZ26], RL [ZDY+26], through the Hessian lens
- ^ **A new perspective to understand AI models through exciting connections to random matrix theory, spectral analysis, tail behaviors!**

Desiderata: an information-theoretic measure of modern AI models

- ^ By measuring spectral statistics of the loss landscape Hessian, can explain numerous empirical phenomena related to SFT [JLZ22; ZLJ24], grokking [ZZL+25a], and more
- ^ New insights and connections on LLM inference [ZZL+25b], model interpretability [ZDZ26], RL [ZDY+26], through the Hessian lens
- ^ **A new perspective to understand AI models through exciting connections to random matrix theory, spectral analysis, tail behaviors!**

Lab page: <https://virtuosoresearch.github.io/>. We run an ML Foundations seminar; Drop me a line if you are in the Northeast region!

Open-source libraries and codes: <https://github.com/VirtuosoResearch>

- ^ <https://github.com/VirtuosoResearch/NNHessian>

Acknowledgment

Ziniu Zhang

Zhenshuo Zhang

Dongyue Li

Minxuan Duan

Haotian Ju

+ Many collaborators: Prof. Jennifer Dy (Northeastern), Chris Re (Stanford),
Lu Wang (Michigan)

Funding source: Northeastern University Provost's office and Khoury College,
NSF IIS-2412008, JP Morgan Chase






Arora, S., Du, S., Hu, W., Li, Z., and Wang, R. (2019). "Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks". In: International Conference on Machine Learning. PMLR, pp. 322{332 (6{12}).






Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017). "Spectrally-normalized margin bounds for neural networks". In: Advances in neural information processing systems 30 (6{12}).



Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A. (2020). "Benign overfitting in linear regression". In: Proceedings of the National Academy of Sciences 117.48, pp. 30063{30070 (6{12}).



Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). "Reconciling modern machine-learning practice and the classical bias-variance trade-off". In: Proceedings of the National Academy of Sciences 116.32, pp. 15849{15854 (6{12}).

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). "Language models are few-shot learners". In: Advances in neural information processing systems 33, pp. 1877{1901 (2{5}).

-  Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. (2025). **“Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning”**. In: *arXiv preprint arXiv:2501.12948* (13–16).
-  Ilyas, A., Park, S. M., Engstrom, L., Leclerc, G., and Madry, A. (2022). **“Datamodels: Predicting predictions from training data”**. In: *ICML* (37–40).
-  Ju, H., Li, D., Sharma, A., and Zhang, H. R. (2023). **“Generalization in graph neural networks: Improved pac-bayesian bounds on graph diffusion”**. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 6314–6341 (6–12).
-  Ju, H., Li, D., and Zhang, H. R. (2022). **“Robust fine-tuning of deep neural networks with hessian-based generalization guarantees”**. In: *International Conference on Machine Learning*. PMLR, pp. 10431–10461 (6–12, 17–22, 60–63).
-  Koh, P. W. and Liang, P. (2017). **“Understanding black-box predictions via influence functions”**. In: *International conference on machine learning*. PMLR, pp. 1885–1894 (37–40).

-  Li, D., Nguyen, H. L., and Zhang, H. R. (2023). **“Identification of Negative Transfers in Multitask Learning using Surrogate Models”**. In: *Transactions on Machine Learning Research (Featured Certification)*. URL: <https://openreview.net/forum?id=KgfFAI9f3E> (41, 42).
-  Li, Y., Ma, T., and Zhang, H. (2018). **“Algorithmic Regularization in Over-parameterized Matrix Sensing and Neural Networks with Quadratic Activations”**. In: *Conference On Learning Theory* (6–12).
-  Park, S. M., Georgiev, K., Ilyas, A., Leclerc, G., and Madry, A. (2023). **“TRAK: Attributing Model Behavior at Scale”**. In: *International Conference on Machine Learning*. PMLR, pp. 27074–27113 (37–40).
-  Shannon, C. E. (1948). **“A mathematical theory of communication”**. In: *The Bell system technical journal* 27.3, pp. 379–423 (3, 4).
-  Vardi, G. (2023). **“On the implicit bias in deep-learning algorithms”**. In: *Communications of the ACM* 66.6, pp. 86–93 (6–12).
-  Wang, S., Shen, Y., Sun, H., Feng, S., Teng, S.-H., Dong, L., Hao, Y., and Chen, W. (2025). **“Benefits and pitfalls of reinforcement learning for language model planning: a theoretical perspective”**. In: *arXiv preprint arXiv:2509.22613* (13–16).

-  Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021). **“Understanding deep learning (still) requires rethinking generalization”**. In: *Communications of the ACM* 64.3, pp. 107–115 (3, 4).
-  Zhang, H. R., Li, D., and Ju, H. (2024). **“Noise Stability Optimization for Finding Flat Minima: A Hessian-based Regularization Approach”**. In: *Transactions on Machine Learning Research* (6–12, 17, 60–63).
-  Zhang, H. R., Zhang, Z., Liu, J. W., and Re, C. (2025a). **“Label-Invariant Hessian Regularization Mitigates Grokking in Mathematical Reasoning”**. In: *NeurIPS workshop* (17, 60–63).
-  Zhang, T. (2023). **Mathematical analysis of machine learning algorithms**. Cambridge University Press (6–12).
-  Zhang, Z., Duan, M., Ye, Y., and Zhang, H. R. (2026a). **“Scalable Multi-Objective and Meta Reinforcement Learning via Gradient Estimation”**. In: *AAAI* (16, 53, 61–63).
-  Zhang, Z., Duan, M., and Zhang, H. R. (2026b). **“Efficient Estimation of Kernel Surrogate Models for Task Attribution”**. In: *ICLR* (16, 32, 43–45, 49–52, 61–63).

-  Zhang, Z., Zhang, Z., Li, D., Wang, L., Dy, J., and Zhang, H. R. (2025b). **“Linear-Time Demonstration Selection for In-Context Learning via Gradient Estimation”**. In: *EMNLP* (16, 32, 49–52, 61–63).
-  Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., et al. (2023). **“Webarena: A realistic web environment for building autonomous agents”**. In: *arXiv preprint arXiv:2307.13854* (14–16).