Generalization of neural networks: A Hessian view (Some partial progress)

Hongyang Ryan Zhang

Assistant Professor, Khoury College of Computer Sciences
177 Huntington Ave (22nd floor), Northeastern University, Boston

December 4, 2024

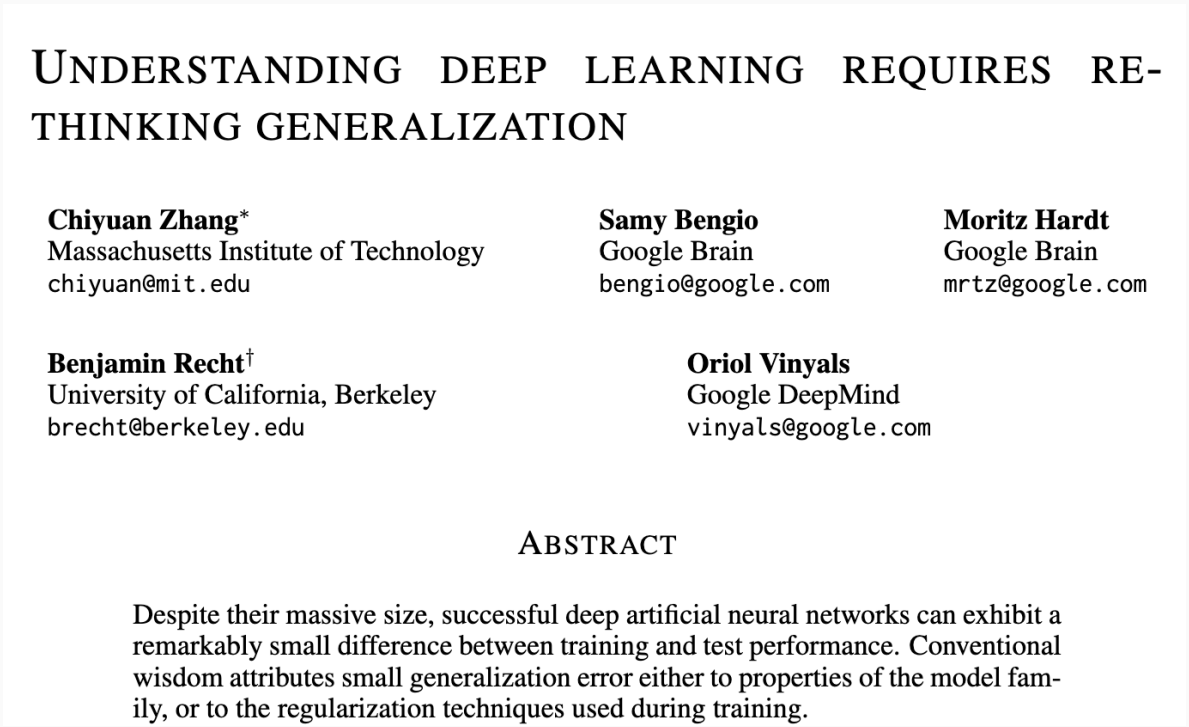
Motivating questions

Some intriguing questions related to the working of GPT-3: How does a large model pretrained on a large corpus of unlabeled data learn, and why is it adaptable to multiple downstream tasks?

Motivating questions

Some intriguing questions related to the working of GPT-3: How does a large model pretrained on a large corpus of unlabeled data learn, and why is it adaptable to multiple downstream tasks?

- Related to research on the generalization of over-parameterized models (2017): Modern deep networks (e.g., ResNet, BERT) have more parameters than dataset labels to memorize training data [ZBH+21]



UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION

Chiyuan Zhang*
Massachusetts Institute of Technology
chiyuan@mit.edu

Samy Bengio
Google Brain
bengio@google.com

Moritz Hardt
Google Brain
mrtz@google.com

Benjamin Recht†
University of California, Berkeley
brecht@berkeley.edu

Oriol Vinyals
Google DeepMind
vinyals@google.com

ABSTRACT

Despite their massive size, successful deep artificial neural networks can exhibit a remarkably small difference between training and test performance. Conventional wisdom attributes small generalization error either to properties of the model family, or to the regularization techniques used during training.

Generalization of language models

Recently, there has been growing interest in the generalization capability of LLMs (broadly de ned)

ACL 2025 Theme Track: Generalization of NLP Models

Following the success of the ACL 2020-2024 Theme tracks, we are happy to announce that ACL 2025 will have a new theme with the goal of reflecting and stimulating discussion about the current state of development of the field of NLP.

Generalization is crucial for ensuring that models behave robustly, reliably, and fairly when making predictions on data different from their training data. Achieving good generalization is critically important for models used in real-world applications, as they should emulate human-like behavior. Humans are known for their ability to generalize well, and models should aspire to this standard.

The theme track invites empirical and theoretical research and position and survey papers reflecting on the Generalization of NLP Models. The possible topics of discussion include (but are not limited to) the following:

- How can we enhance the generalization of NLP models across various dimensions—compositional, structural, cross-task, cross-lingual, cross-domain, and robustness?
- What factors affect the generalization of NLP models?
- What are the most effective methods for evaluating the generalization capabilities of NLP models?
- While Large Language Models (LLMs) significantly enhance the generalization of NLP models, what are the key limitations of LLMs in this regard?

Generalization of language models

Understanding what information is presented in a natural language and how we may quantify content is certainly not a new research quest*:::*

- The notion of entropy for measuring information in a language dates back to very classical work [Sha48]
- Applying entropy to N-gram statistics gives a measure of information (more recent instantiations include next-word prediction in GPT-3, and next-sentence prediction in BERT)

Generalization of learning with IID samples

Training: given n samples from D, $(x_1; y_1); (x_2; y_2); : : : ; (x_n; y_n)$,

$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_W(x_i); y_i)$$

Test: expected risk over a random sample of D

$$L(f_W) = \mathop{\mathbb{E}}_{(x;y) \quad D} [\ell(f_W(x); y)]$$

Generalization analysis of ML algorithms: given a hypothesis space of neural nets, how do we quantify generalization gap $L(f_W) \quad \hat{L}(f_W)$?

- **Rademacher complexity** [BFT17]
- **PAC-Bayes** [AGN+18; JLZ22]: data-dependent bounds
- **Neural tangent kernels** [ADH+19]: high width, xed random matrix
- **Implicit regularization** [LMZ18]: starting from a small, random initialization, SGD searches inside a low-rank space
- **Benign overfitting** [BLL+20]

See a recent textbook by Tong Zhang [Zha23] (UIUC) for further references

Generalization of learning with IID samples

Training: given n samples from D, $(x_1; y_1); (x_2; y_2); : : : ; (x_n; y_n)$,

$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_W(x_i); y_i)$$

Test: expected risk over a random sample of D

$$L(f_W) = \mathop{\mathbb{E}}_{(x;y) \quad D} [\ell(f_W(x); y)]$$

Generalization analysis of ML algorithms: given a hypothesis space of neural nets, how do we quantify generalization gap $L(f_W) \quad \hat{L}(f_W)$?

- **Rademacher complexity** [BFT17]
- **PAC-Bayes** [AGN+18; JLZ22]: data-dependent bounds
- **Neural tangent kernels** [ADH+19]: high width, fixed random matrix
- **Implicit regularization** [LMZ18]: starting from a small, random initialization, SGD searches inside a low-rank space
- **Benign overfitting** [BLL+20]

See a recent textbook by Tong Zhang [Zha23] (UIUC) for further references

Generalization of learning with IID samples

Training: given n samples from D, $(x_1; y_1); (x_2; y_2); : : : ; (x_n; y_n)$,

$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_W(x_i); y_i)$$

Test: expected risk over a random sample of D

$$L(f_W) = \mathop{\mathbb{E}}_{(x;y) \quad D} [\ell(f_W(x); y)]$$

Generalization analysis of ML algorithms: given a hypothesis space of neural nets, how do we quantify generalization gap $L(f_W) \quad \hat{L}(f_W)$?

- **Rademacher complexity** [BFT17]
- **PAC-Bayes** [AGN+18; JLZ22]: data-dependent bounds
- **Neural tangent kernels** [ADH+19]: high width, xed random matrix
- **Implicit regularization** [LMZ18]: starting from a small, random initialization, SGD searches inside a low-rank space
- **Benign overfitting** [BLL+20]

See a recent textbook by Tong Zhang [Zha23] (UIUC) for further references

Generalization of learning with IID samples

Training: given n samples from D, $(x_1; y_1); (x_2; y_2); : : : ; (x_n; y_n)$,

$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_W(x_i); y_i)$$

Test: expected risk over a random sample of D

$$L(f_W) = \mathop{\mathbb{E}}_{(x;y) \quad D} [\ell(f_W(x); y)]$$

Generalization analysis of ML algorithms: given a hypothesis space of neural nets, how do we quantify generalization gap $L(f_W) \quad \hat{L}(f_W)$?

- **Rademacher complexity** [BFT17]
- **PAC-Bayes** [AGN+18; JLZ22]: data-dependent bounds
- **Neural tangent kernels** [ADH+19]: high width, xed random matrix
- **Implicit regularization** [LMZ18]: starting from a small, random initialization, SGD searches inside a low-rank space
- **Benign overfitting** [BLL+20]

See a recent textbook by Tong Zhang [Zha23] (UIUC) for further references

Generalization of learning with IID samples

Training: given n samples from D, $(x_1; y_1); (x_2; y_2); : : : ; (x_n; y_n)$,

$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_W(x_i); y_i)$$

Test: expected risk over a random sample of D

$$L(f_W) = \mathop{\mathbb{E}}_{(x;y) \quad D} [\ell(f_W(x); y)]$$

Generalization analysis of ML algorithms: given a hypothesis space of neural nets, how do we quantify generalization gap $L(f_W) \quad \hat{L}(f_W)$?

- **Rademacher complexity** [BFT17]
- **PAC-Bayes** [AGN+18; JLZ22]: data-dependent bounds
- **Neural tangent kernels** [ADH+19]: high width, xed random matrix
- **Implicit regularization** [LMZ18]: starting from a small, random initialization, SGD searches inside a low-rank space
- **Benign overfitting** [BLL+20]

See a recent textbook by Tong Zhang [Zha23] (UIUC) for further references

Generalization of learning with IID samples

Training: given n samples from D, $(x_1; y_1); (x_2; y_2); : : : ; (x_n; y_n)$,

$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_W(x_i); y_i)$$

Test: expected risk over a random sample of D

$$L(f_W) = \mathop{\mathbb{E}}_{(x;y) \quad D} [\ell(f_W(x); y)]$$

Generalization analysis of ML algorithms: given a hypothesis space of neural nets, how do we quantify generalization gap $L(f_W) \quad \hat{L}(f_W)$?

- **Rademacher complexity** [BFT17]
- **PAC-Bayes** [AGN+18; JLZ22]: data-dependent bounds
- **Neural tangent kernels** [ADH+19]: high width, xed random matrix
- **Implicit regularization** [LMZ18]: starting from a small, random initialization, SGD searches inside a low-rank space
- **Benign overfitting** [BLL+20]

See a recent textbook by Tong Zhang [Zha23] (UIUC) for further references

Generalization of learning with IID samples

Training: given n samples from D, $(x_1; y_1); (x_2; y_2); : : : ; (x_n; y_n)$,

$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_W(x_i); y_i)$$

Test: expected risk over a random sample of D

$$L(f_W) = \mathop{\mathbb{E}}_{(x;y) \quad D} [\ell(f_W(x); y)]$$

Generalization analysis of ML algorithms: given a hypothesis space of neural nets, how do we quantify generalization gap $L(f_W) \quad \hat{L}(f_W)$?

- **Rademacher complexity** [BFT17]
- **PAC-Bayes** [AGN+18; JLZ22]: data-dependent bounds
- **Neural tangent kernels** [ADH+19]: high width, xed random matrix
- **Implicit regularization** [LMZ18]: starting from a small, random initialization, SGD searches inside a low-rank space
- **Benign overfitting** [BLL+20]

See a recent textbook by Tong Zhang [Zha23] (UIUC) for further references

Generalization of learning with IID samples

Training: given n samples from D, $(x_1; y_1); (x_2; y_2); : : : ; (x_n; y_n)$,

$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_W(x_i); y_i)$$

Test: expected risk over a random sample of D

$$L(f_W) = \mathop{\mathbb{E}}_{(x;y) \quad D} [\ell(f_W(x); y)]$$

Generalization analysis of ML algorithms: given a hypothesis space of neural nets, how do we quantify generalization gap $L(f_W) \quad \hat{L}(f_W)$?

- **Rademacher complexity** [BFT17]
- **PAC-Bayes** [AGN+18; JLZ22]: data-dependent bounds
- **Neural tangent kernels** [ADH+19]: high width, xed random matrix
- **Implicit regularization** [LMZ18]: starting from a small, random initialization, SGD searches inside a low-rank space
- **Benign overfitting** [BLL+20]

See a recent textbook by Tong Zhang [Zha23] (UIUC) for further references

Neural network setup

- f_W: A multi-layer neural net with weight matrices $W = [W_1; W_2; \ldots; W_l]$
- D: An unknown distribution over feature space times label space $X \quad Y$

Norm bounds for neural nets

- Hypothesis space with bounded ℓ_2 and Frobenius norms:

$$H = \left\{ \begin{array}{l} kW_1k_2 \quad s_1; kW_2k_2 \quad s_2; \ldots; kW_lk_2 \quad s_l; \\ kW_1k_F \quad s_1r_1; kW_2k_F \quad s_2r_2; \ldots; kW_lk_F \quad s_lr_l \end{array} \right\} \qquad (1)$$

- For any l-layer deep net f_W whose weight matrices W belong to H, with Lipschitz-continuous activation functions at every layer, the following holds w.h.p.

$$L(f_W) \quad \hat{L}(f_W) \; . \; \sqrt{\frac{\prod_{i=1}^{l} s_i^2 \quad \sum_{i=1}^{l} r_i^2}{n}}$$

  - The bound grows exponentially with depth, and is tight in the worst-case

Neural network setup

- f_W: A multi-layer neural net with weight matrices $W = [W_1; W_2; \ldots; W_l]$
- D: An unknown distribution over feature space times label space $X \quad Y$

Norm bounds for neural nets

- Hypothesis space with bounded ℓ_2 and Frobenius norms:

$$H = \left\{ \begin{array}{cccccc} \|W_1\|_2 & s_1; \|W_2\|_2 & s_2; \ldots; \|W_l\|_2 & s_l; \\ \|W_1\|_F & s_1 r_1; \|W_2\|_F & s_2 r_2; \ldots; \|W_l\|_F & s_l r_l \end{array} \right\} \quad (1)$$

- For any l-layer deep net f_W whose weight matrices W belong to H, with Lipschitz-continuous activation functions at every layer, the following holds w.h.p.

$$L(f_W) \quad \hat{L}(f_W) \; . \; \sqrt{\frac{\prod_{i=1}^{l} s_i^2 \quad \sum_{i=1}^{l} r_i^2}{n}}$$

  - The bound grows exponentially with depth, and is tight in the worst-case

Neural network setup

- f_W: A multi-layer neural net with weight matrices $W = [W_1; W_2; \ldots; W_l]$
- D: An unknown distribution over feature space times label space $X \quad Y$

Norm bounds for neural nets

- Hypothesis space with bounded ℓ_2 and Frobenius norms:

$$H = \left\{ \begin{array}{l} \|W_1\|_2 \quad s_1; \|W_2\|_2 \quad s_2; \ldots; \|W_l\|_2 \quad s_l; \\ \|W_1\|_F \quad s_1 r_1; \|W_2\|_F \quad s_2 r_2; \ldots; \|W_l\|_F \quad s_l r_l \end{array} \right\} \tag{1}$$

- For any l-layer deep net f_W whose weight matrices W belong to H, with Lipschitz-continuous activation functions at every layer, the following holds w.h.p.

$$L(f_W) \quad \hat{L}(f_W) \; . \; \sqrt{\frac{\prod_{i=1}^{l} s_i^2 \quad \sum_{i=1}^{l} r_i^2}{n}}$$

  - The bound grows exponentially with depth, and is tight in the worst-case

Neural network setup

- f_W: A multi-layer neural net with weight matrices $W = [W_1; W_2; \ldots; W_l]$
- D: An unknown distribution over feature space times label space $X \quad Y$

Norm bounds for neural nets

- Hypothesis space with bounded ℓ_2 and Frobenius norms:

$$H = \left\{ \begin{array}{l} \|W_1\|_2 \quad s_1; \|W_2\|_2 \quad s_2; \ldots; \|W_l\|_2 \quad s_l; \\ \|W_1\|_F \quad s_1 r_1; \|W_2\|_F \quad s_2 r_2; \ldots; \|W_l\|_F \quad s_l r_l \end{array} \right\} \tag{1}$$

- For any l-layer deep net f_W whose weight matrices W belong to H, with Lipschitz-continuous activation functions at every layer, the following holds w.h.p.

$$L(f_W) \quad \hat{L}(f_W) \; . \; \sqrt{\frac{\prod_{i=1}^{l} s_i^2 \quad \sum_{i=1}^{l} r_i^2}{n}}$$

  - The bound grows exponentially with depth, and is tight in the worst-case

Today's talk

Central thesis: Measure generalization through the Hessian of loss surfaces

Derivation of a Hessian trace measure on the generalization gap

- Main ideas
- Hessian-based regularization for neural net training

Generalization in graph neural networks

- Setup and results
- Numerical results

Problem setup

An l-layer feedforward neural network with weight matrices $W_1; W_2; : : : ; W_l$:

$$f_W(x) = \quad_l \quad ::: \quad_3 \quad W_3 \quad_2 \quad W_2 \quad_1 (W_1 x)$$

- Supervised learning: Train weights from random initialization
- Transfer learning: Adapt weights from pretrained foundations models (e.g., ne-tuning)

Problem setup

An l-layer feedforward neural network with weight matrices $W_1; W_2; : : : ; W_l$:

$$f_W(x) = {}_l \ ::: \ {}_3 \ W_3 \ {}_2 \ W_2 \ {}_1(W_1 x)$$

- Supervised learning: Train weights from random initialization
- Transfer learning: Adapt weights from pretrained foundations models (e.g., ne-tuning)

Problem setup

An l-layer feedforward neural network with weight matrices $W_1; W_2; : : : ; W_l$:

$$f_W(x) = {}_{l} \; ::: \; {}_{3} \; W_3 \; {}_{2} \; W_2 \; {}_{1}(W_1 x)$$

- Supervised learning: Train weights from random initialization
- Transfer learning: Adapt weights from pretrained foundations models (e.g., ne-tuning)

Distance from initialization a ects generalization

The distance between the initialization and the model can a ect generalization: This corresponds to the spectral radius of the hypothesis space [NK19]

Figure 1: Results from Li and Zhang [LZ21], by ne-tuning ResNet-18 on seven image classi cation tasks

Figure 2: Similar results have appeared in prior work on transfer learning in deep neural networks (folklore)

Our approach based on PAC-Bayes bounds

PAC-Bayes bounds

- $P = \mathcal{N}(W^{(0)}; \sigma^2 \mathrm{Id})$: prior distribution centered at initialization weights
- $Q = \mathcal{N}(W^{(T)}; \sigma^2 \mathrm{Id})$: posterior distribution centered at trained weights
- With probability at least $1 - \delta$ for any $\delta \in (0; 1)$ (see, e.g., McAllester [McA13])

$$\mathop{\mathbb{E}}_{w \sim Q}[L(f_w)] \leq \mathop{\mathbb{E}}_{w \sim Q}\left[\hat{L}(f_w)\right] + \sqrt{\frac{KL(Q \| P) + \log(4n/\delta)}{n}} \tag{2}$$

We will use a generalized result, which subsumes the above result by setting λ appropriately. For any $\lambda \in (0; 1)$, with probability at least $1 - \delta$ [Cat07]

$$\mathop{\mathbb{E}}_{w \sim Q}[L(f_w)] \leq \frac{1}{1 - \lambda/2}\left(\mathop{\mathbb{E}}_{w \sim Q}\left[\hat{L}(f_w)\right] + \frac{KL(Q \| P) + \log(1/\delta)}{\lambda(1 - \lambda/2)n}\right) \tag{3}$$

Our approach based on PAC-Bayes bounds

PAC-Bayes bounds

- $P = \mathcal{N}(W^{(0)}; \sigma^2 \mathrm{Id})$: prior distribution centered at initialization weights
- $Q = \mathcal{N}(W^{(T)}; \sigma^2 \mathrm{Id})$: posterior distribution centered at trained weights
- With probability at least $1 - \delta$ for any $\delta \in (0; 1)$ (see, e.g., McAllester [McA13])

$$\mathop{\mathbb{E}}_{w \sim Q}[L(f_w)] \leq \mathop{\mathbb{E}}_{w \sim Q}\left[\hat{L}(f_w)\right] + \sqrt{\frac{KL(Q \| P) + \log(4n\delta^{-1})}{n}} \tag{2}$$

We will use a generalized result, which subsumes the above result by setting λ appropriately. For any $\lambda \in (0; 1)$, with probability at least $1 - \delta$ [Cat07]

$$\mathop{\mathbb{E}}_{w \sim Q}[L(f_w)] \leq \frac{1}{1 - \lambda/2} \mathop{\mathbb{E}}_{w \sim Q}\left[\hat{L}(f_w)\right] + \frac{KL(Q \| P) + \log(\delta^{-1})}{\lambda(1 - \lambda/2)n} \tag{3}$$

Our approach based on PAC-Bayes bounds

PAC-Bayes bounds

- $P = \mathcal{N}(W^{(0)}; \ ^2 \mathrm{Id})$: prior distribution centered at initialization weights
- $Q = \mathcal{N}(W^{(T)}; \ ^2 \mathrm{Id})$: posterior distribution centered at trained weights
- With probability at least 1 for any $2\ (0;1)$ (see, e.g., McAllester [McA13])

$$\underset{w\ Q}{E}\ [L(f_w)] \quad \underset{w\ Q}{E}\ \left[\hat{L}(f_w)\right] + \sqrt{\frac{KL(Q||P) + \log(4\ n\ ^{1})}{n}} \qquad (2)$$

We will use a generalized result, which subsumes the above result by setting appropriately. For any $2\ (0;1)$, with probability at least 1 [Cat07]

$$\underset{w\ Q}{E}\ [L(f_w)] \quad \frac{1}{\ } \underset{w\ Q}{E}\ \left[\hat{L}(f_w)\right] + \frac{KL(Q||P) + \log(\ ^{1})}{2\ (1\ \ )n} \qquad (3)$$

Our approach based on PAC-Bayes bounds

PAC-Bayes bounds

- $P = N(W^{(0)}; \ ^2 Id)$: prior distribution centered at initialization weights
- $Q = N(W^{(T)}; \ ^2 Id)$: posterior distribution centered at trained weights
- With probability at least 1 for any $2 (0; 1)$ (see, e.g., McAllester [McA13])

$$\underset{w \ Q}{E} [L(f_w)] \quad \underset{w \ Q}{E} \left[\hat{L}(f_w) \right] + \sqrt{\frac{KL(Q||P) + \log(4 \ n \ ^1)}{n}} \qquad (2)$$

We will use a generalized result, which subsumes the above result by setting appropriately. For any $2 (0; 1)$, with probability at least 1 [Cat07]

$$\underset{w \ Q}{E} [L(f_w)] \quad \frac{1}{} \underset{w \ Q}{E} \left[\hat{L}(f_w) \right] + \frac{KL(Q||P) + \log(\ ^1)}{2 \ (1 \ )n} \qquad (3)$$

Our nding: the Hessian of loss surface can also a ect generalization

Claim 1: Use Taylor's expansion on the perturbed loss

Measure model stability after adding perturbations to weight parameters. Let $\grave{}_Q(f_W) = E_{W\ Q}[\grave{}(f_W)] = E_{U\ N\ (0;\ ^2 Id)}[\grave{}(f_{W+U})]$. We have

$$\grave{}_Q(f_W(x); y) \quad \grave{}(f_W(x); y) \quad \frac{1}{2}\ ^2 Tr\left[r\ ^2[\grave{}(f_W(x); y)]\right] \quad C_1\ ^3 \qquad (4)$$

Our nding: the Hessian of loss surface can also a ect generalization

Claim 1: Use Taylor's expansion on the perturbed loss

Measure model stability after adding perturbations to weight parameters. Let $\grave{}_Q(f_W) = E_{W\ Q}[\grave{}(f_W)] = E_{U\ N\ (0;\ ^2 Id)}[\grave{}(f_{W+U})]$. We have

$$\grave{}_Q(f_W(x); y) \quad \grave{}(f_W(x); y) \quad \frac{1}{2}\ ^2 \mathrm{Tr}\left[\ r\ ^2[\grave{}(f_W(x); y)]\right] \quad C_1\ ^3 \qquad (4)$$

Figure 3: Illustration of the Hessian approximation in equation (4). We report the results at the last epoch. : standard deviation of Gaussian noise injected into weight matrices.

Our nding: the Hessian of loss surface can also a ect generalization

Claim 1: Use Taylor's expansion on the perturbed loss

Measure model stability after adding perturbations to weight parameters. Let $\grave{}_Q(f_W) = E_{W\ Q}\ [\grave{}(f_W)] = E_{U\ N\ (0;\ ^2\mathrm{Id})}\ [\grave{}(f_{W+U})]$. We have

$$\grave{}_Q(f_W(x); y) \quad \grave{}(f_W(x); y) \quad \frac{1}{2}\ ^2\mathrm{Tr}\left[\ r\ ^2[\grave{}(f_W(x); y)]\right] \qquad C_1\ ^3 \qquad (4)$$

Claim 2: Uniform convergence of the Hessian operator

Because the Hessian operator is Lipschitz-continuous, the trace of the Hessian satis es the uniform convergence within the hypothesis space H

A sketch for deriving the Hessian trace measure

From equation (3),

$$L_Q(W) \quad \frac{1}{-}\hat{L}_Q(W) + \frac{C(KL(Q\|P) + \log(\quad^{1}))}{2\ (1\quad)n} \tag{5}$$

By Claim 1 (applied to both the expected loss L_Q and the empirical loss $\hat{L}_Q$),

$$L_Q(W) = L(W) + \frac{^2}{2}\ \mathop{E}_{(x;y)\ D}\ \mathrm{Tr}\ r\ ^{2\grave{}}(f_W(x);y)\ \ + O(\ ^3) \tag{6}$$

$$\hat{L}_Q(W) = \hat{L}(W) + \frac{^2}{2n}\sum_{i=1}^{n}\ \mathrm{Tr}\ r\ ^{2\grave{}}(f_W(x_i);y_i)\ \ + O(\ ^3) \tag{7}$$

By Claim 2, the di erence between the second terms of equations (6), (7) is of order $O(n^{\frac{1}{2}})$

By choosing ^{2} and carefully, we get [ZLJ24]

$$L(W)\quad (1+\ )\hat{L}(W) + (1+\ )\ \sqrt{\frac{C\ r^2}{n}} + O\ n^{\frac{3}{4}}\log(\ ^{1})\ ; \tag{8}$$

where is the trace norm of the hypothesis space, and is the radius of the hypothesis space

A sketch for deriving the Hessian trace measure

From equation (3),

$$L_Q(W) \quad \frac{1}{} \hat{L}_Q(W) + \frac{C(KL(Q\|P) + \log(\quad^{1}))}{2\ (1\quad)n} \tag{5}$$

By Claim 1 (applied to both the expected loss L_Q and the empirical loss $\hat{L}_Q$),

$$L_Q(W) = L(W) + \frac{^2}{2} \underset{(x;y)\ D}{E} \left[Tr\left[r^{\ 2`}(f_W(x); y) \right] \right] + O(\ ^3) \tag{6}$$

$$\hat{L}_Q(W) = \hat{L}(W) + \frac{^2}{2n} \sum_{i=1}^{n} \left[Tr\left[r^{\ 2`}(f_W(x_i); y_i) \right] \right] + O(\ ^3) \tag{7}$$

By Claim 2, the di erence between the second terms of equations (6), (7) is of order $O(n^{\frac{1}{2}})$

By choosing ^{2} and carefully, we get [ZLJ24]

$$L(W) \quad (1 + \ )\hat{L}(W) + (1 + \ ) \sqrt{\frac{C\ r^2}{n}} + O\left(n^{\frac{3}{4}} \log(\quad^{1})\right); \tag{8}$$

where is the trace norm of the hypothesis space, and is the radius of the hypothesis space

A sketch for deriving the Hessian trace measure

From equation (3),

$$L_Q(W) \quad \frac{1}{-}\hat{L}_Q(W) + \frac{C(KL(Q\|P) + \log(\quad^{1}))}{2\ (1\quad)n} \tag{5}$$

By Claim 1 (applied to both the expected loss L_Q and the empirical loss $\hat{L}_Q$),

$$L_Q(W) = L(W) + \frac{^2}{2} \underset{(x;y)\ D}{E} \left[Tr\left[r^{\ 2`}(f_W(x); y) \right] \right] + O(\ ^3) \tag{6}$$

$$\hat{L}_Q(W) = \hat{L}(W) + \frac{^2}{2n} \sum_{i=1}^{n} \left[Tr\left[r^{\ 2`}(f_W(x_i); y_i) \right] \right] + O(\ ^3) \tag{7}$$

By Claim 2, the di erence between the second terms of equations (6), (7) is of order $O(n^{\ \frac{1}{2}})$

By choosing ^{2} and carefully, we get [ZLJ24]

$$L(W) \quad (1 + \ )\hat{L}(W) + (1 + \ ) \sqrt{\frac{C\ r^2}{n}} + O\left(n^{\ \frac{3}{4}} \log(\quad^{1})\right); \tag{8}$$

where is the trace norm of the hypothesis space, and is the radius of the hypothesis space

A sketch for deriving the Hessian trace measure

From equation (3),

$$L_Q(W) \quad \frac{1}{} \hat{L}_Q(W) + \frac{C(KL(Q\|P) + \log(\quad^{1}))}{2\ (1\quad)n} \tag{5}$$

By Claim 1 (applied to both the expected loss L_Q and the empirical loss $\hat{L}_Q$),

$$L_Q(W) = L(W) + \frac{^2}{2} \underset{(x;y)\ D}{E} \left[Tr\left[r^{\,2`}(f_W(x); y) \right] \right] + O(\ ^3) \tag{6}$$

$$\hat{L}_Q(W) = \hat{L}(W) + \frac{^2}{2n} \sum_{i=1}^{n} \left[Tr\left[r^{\,2`}(f_W(x_i); y_i) \right] \right] + O(\ ^3) \tag{7}$$

By Claim 2, the di erence between the second terms of equations (6), (7) is of order $O(n^{\frac{1}{2}})$

By choosing ^{2} and carefully, we get [ZLJ24]

$$L(W) \quad (1 + \ )\hat{L}(W) + (1 + \ ) \sqrt{\frac{C\ r^2}{n}} + O\left(n^{\frac{3}{4}} \log(\ ^{1})\right); \tag{8}$$

where is the trace norm of the hypothesis space, and r is the radius of the hypothesis space

Numerical results

The Hessian trace measure correlates well with empirical measurements of the generalization gap. Compare across seven ne-tuning methods: SGD, early stopping, weight decay, label smoothing, mixup, distance-based regularization, sharpness-aware minimization

(a) ResNet-50 (b) ResNet-50 (c) BERT-Base

Figure 3: The Hessian measures accurately correlate with empirical generalization errors for seven ne-tuning methods

Numerical results

A very related study pursuing a compression-based approach [LFK+22]

Noise injection induces a regularization of the Hessian

An algorithmic implication : Instead of minimizing the loss $\tilde{\ell}$, we could minimize $\tilde{\ell}_Q = E_{U \sim N(0; \sigma^2 Id)}[\tilde{\ell}(f_{W+U})]$ instead. By equation (4), this regularizes trace of the Hessian of the loss surface

Experimental results : Comparison between SGD, noise injection (directly add noise before computing gradient, i.e., WP-SGD), and a two-point noise injection algorithm (NSO)

Noise injection induces a regularization of the Hessian

An algorithmic implication : Instead of minimizing the loss $\grave{}$, we could minimize $\grave{}_Q = E_{U\ N\ (0;\ ^2\ Id)}[\grave{}(f_{W+U})]$ instead. By equation (4), this regularizes trace of the Hessian of the loss surface

Experimental results: Comparison between SGD, noise injection (directly add noise before computing gradient, i.e., WP-SGD), and a two-point noise injection algorithm (NSO)

Figure 3: Fine-tuning ResNet-34 and BERT-Base, respectively, on an image and a text classi cation dataset. Similar results for more recent architectures (multi-modal, CLIP), chain-of-thought ne-tuning (LM, transformer) [ZLJ24]

Outline

Message-passing neural networks

Graph neural networks are neural networks designed for working with graphs

- Graph $G = (V, E)$: V is a set of vertices and E is a set of edges
- Every node has a feature vector x_v for all $v \in V$: let X be a feature matrix
- Graph-level prediction: a prediction label $y \in Y$ for each graph instance G

Message passing neural networks (MPNNs) with a pre-defined graph diffusion matrix/mechanism P_G (on each graph G)

- Let $H^{(0)} = X$
- For the first $l - 1$ layers, recursively compute node embedding

$$H^{(t)} = \alpha_t X U^{(t)} + \beta_t P_G \sigma_t(H^{(t-1)}) W^{(t)}, \text{ for } t = 1, 2, \ldots, l \quad (9)$$

- For the last layer l, aggregate the embedding of all nodes

$$H^{(l)} = \frac{1}{n} 1_n^\top H^{(l-1)} W^{(l)}. \quad (10)$$

Message-passing neural networks

Graph neural networks are neural networks designed for working with graphs

- Graph $G = (V, E)$: V is a set of vertices and E is a set of edges
- Every node has a feature vector x_v for all $v \in V$: let X be a feature matrix
- Graph-level prediction: a prediction label $y \in Y$ for each graph instance G

Message passing neural networks (MPNNs) with a pre-defined graph diffusion matrix/mechanism P_G (on each graph G)

- Let $H^{(0)} = X$
- For the first $l - 1$ layers, recursively compute node embedding

$$H^{(t)} = {}_t XU^{(t)} + {}_t P_G {}_t(H^{(t-1)}) W^{(t)}, \text{ for } t = 1, 2, \ldots, l \quad (9)$$

- For the last layer l, aggregate the embedding of all nodes

$$H^{(l)} = \frac{1}{n} 1_n^\top H^{(l-1)} W^{(l)}. \quad (10)$$

Message-passing neural networks

Graph neural networks are neural networks designed for working with graphs

- Graph $G = (V, E)$: V is a set of vertices and E is a set of edges
- Every node has a feature vector x_v for all $v \in V$: let X be a feature matrix
- Graph-level prediction: a prediction label $y \in Y$ for each graph instance G

Message passing neural networks (MPNNs) with a pre-defined graph diffusion matrix/mechanism P_G (on each graph G)

- Let $H^{(0)} = X$
- For the first $l - 1$ layers, recursively compute node embedding

$$H^{(t)} = {}_t XU^{(t)} + {}_t P_G {}_t(H^{(t-1)}) W^{(t)}, \text{ for } t = 1, 2, \ldots, l \quad (9)$$

- For the last layer l, aggregate the embedding of all nodes

$$H^{(l)} = \frac{1}{n} 1_n^\top H^{(l-1)} W^{(l)}. \quad (10)$$

Message-passing neural networks

Graph neural networks are neural networks designed for working with graphs

- Graph $G = (V, E)$: V is a set of vertices and E is a set of edges
- Every node has a feature vector x_v for all $v \in V$: let X be a feature matrix
- Graph-level prediction: a prediction label $y \in Y$ for each graph instance G

Message passing neural networks (MPNNs) with a pre-defined graph diffusion matrix/mechanism P_G (on each graph G)

- Let $H^{(0)} = X$
- For the first $l - 1$ layers, recursively compute node embedding

$$H^{(t)} = {}_t XU^{(t)} + {}_t P_G {}_t(H^{(t-1)}) W^{(t)}, \text{ for } t = 1, 2, \ldots, l \quad (9)$$

- For the last layer l, aggregate the embedding of all nodes

$$H^{(l)} = \frac{1}{n} 1_n^\top H^{(l-1)} W^{(l)}. \quad (10)$$

Message-passing neural networks

Graph neural networks are neural networks designed for working with graphs

- Graph $G = (V, E)$: V is a set of vertices and E is a set of edges
- Every node has a feature vector x_v for all $v \in V$: let X be a feature matrix
- Graph-level prediction: a prediction label $y \in Y$ for each graph instance G

Message passing neural networks (MPNNs) with a pre-defined graph diffusion matrix/mechanism P_G (on each graph G)

- Let $H^{(0)} = X$
- For the first $l - 1$ layers, recursively compute node embedding

$$H^{(t)} = {}_t XU^{(t)} + {}_t P_G {}_t(H^{(t-1)}) W^{(t)}, \text{ for } t = 1, 2, \ldots, l \quad (9)$$

- For the last layer l, aggregate the embedding of all nodes

$$H^{(l)} = \frac{1}{n} 1_n^\top H^{(l-1)} W^{(l)}. \quad (10)$$

Message-passing neural networks

Graph neural networks are neural networks designed for working with graphs

- Graph $G = (V, E)$: V is a set of vertices and E is a set of edges
- Every node has a feature vector x_v for all $v \in V$: let X be a feature matrix
- Graph-level prediction: a prediction label $y \in Y$ for each graph instance G

Message passing neural networks (MPNNs) with a pre-defined graph diffusion matrix/mechanism P_G (on each graph G)

- Let $H^{(0)} = X$
- For the first $l - 1$ layers, recursively compute node embedding

$$H^{(t)} = {}_t XU^{(t)} + {}_t P_G {}_t (H^{(t-1)}) W^{(t)}; \text{ for } t = 1, 2, \ldots, l \quad (9)$$

- For the last layer l, aggregate the embedding of all nodes

$$H^{(l)} = \frac{1}{n} 1_n^\top H^{(l-1)} W^{(l)}. \quad (10)$$

Message-passing neural networks

Graph neural networks are neural networks designed for working with graphs

- Graph $G = (V, E)$: V is a set of vertices and E is a set of edges
- Every node has a feature vector x_v for all $v \in V$: let X be a feature matrix
- Graph-level prediction: a prediction label $y \in Y$ for each graph instance G

Message passing neural networks (MPNNs) with a pre-de ned graph di usion matrix/mechanism P_G (on each graph G)

- Let $H^{(0)} = X$
- For the rst l 1 layers, recursively compute node embedding

$$H^{(t)} = {}_t XU^{(t)} + {}_t P_G {}_t(H^{(t\ 1)})\ W^{(t)} \ ; \text{ for } t = 1, 2, \ldots, l \qquad (9)$$

- For the last layer l, aggregate the embedding of all nodes

$$H^{(l)} = \frac{1}{n} 1_n^{>} H^{(l\ 1)} W^{(l)}. \qquad (10)$$

Generalization bounds for MPNNs

How does the generalization bound of GNNs scale with graph structures?

Known results apply to the following GNN architectures

- Graph convolutional networks (GCN) [KW17]
- GraphSAGE (SAmple and aggreGatE) [HYL17]
- Graph isomorphism networks (GIN) [XHL+19]

Table 1: Illustration of the dependence on graph structure of existing results. d: max degree, l: number of GNN layers (depth), A: adjacency matrix, D: degree-diagonal matrix of A

	GCN	MPNN	GIN	GraphSAGE
Garg et al. [GJJ20]	$d^{l\ \ 1}$	$d^{l\ \ 1}$	-	-
Liao et al. [LUZ21]	$d^{\frac{l\ \ 1}{2}}$	$d^{l\ \ 1}$	-	-
Ours [JLS+23]	1	$kAk_2^{l\ \ 1}$	$P_{i=1}^{l\ \ 1} \frac{kAk_2^i}{l\ \ 1}$	$D^{\ 1}A^{\frac{l\ \ 1}{2}}$

The dependence on graph structure, scaled with maximum degree d, or the spectral norm of P_G, which is provably d

Generalization bounds for MPNNs

How does the generalization bound of GNNs scale with graph structures?

Known results apply to the following GNN architectures

- Graph convolutional networks (GCN) [KW17]
- GraphSAGE (SAmple and aggreGatE) [HYL17]
- Graph isomorphism networks (GIN) [XHL+19]

Figure 4: Illustration of graph statistics: Comparison on ve social networks from SNAP, measured based on the GCN architecture

Example on a one-layer linear GNN

One-layer linear GNN with average pooling of node embeddings $n = (|V|)$

$$f(X; G) = \frac{1}{n} 1_n^\top P_G X W^{(1)}$$

Thus, the Euclidean norm of $f(X; G)$ is less than

$$\|f(X; G)\| = \left\| \frac{1}{n} 1_n^\top P_G X W^{(1)} \right\|$$

$$\left\| \frac{1}{n} 1_n^\top \right\|_2 \|P_G\|_2 \|X\|_2 \|W^{(1)}\| := C$$

Provided that the loss function $\ell(\cdot; y)$ is Lipschitz-continuous, given N samples, we know from standard results that (see, e.g., Zhang [Zha23])

$$L(f) - \hat{L}(f) \lesssim \sqrt{\frac{C}{N}}.$$

Example on a one-layer linear GNN

One-layer linear GNN with average pooling of node embeddings: $n = |V|$

$$f(X; G) = \frac{1}{n} 1_n^\top P_G X W^{(1)}$$

Thus, the Euclidean norm of $f(X; G)$ is less than

$$\|f(X; G)\| = \left\| \frac{1}{n} 1_n^\top P_G X W^{(1)} \right\|$$

$$\left\| \frac{1}{n} 1_n^\top \right\|_2 \|P_G\|_2 \|X\|_2 \|W^{(1)}\| := C$$

Provided that the loss function $\ell(\cdot; y)$ is Lipschitz-continuous, given N samples, we know from standard results that (see, e.g., Zhang [Zha23])

$$L(f) - \hat{L}(f) \lesssim \frac{C}{\sqrt{N}}.$$

Example on a one-layer linear GNN

One-layer linear GNN with average pooling of node embeddings $n = |V|$

$$f(X; G) = \frac{1}{n} 1_n^\top P_G X W^{(1)}$$

Thus, the Euclidean norm of $f(X; G)$ is less than

$$\|f(X; G)\| = \left\| \frac{1}{n} 1_n^\top P_G X W^{(1)} \right\|$$

$$\left\| \frac{1}{n} 1_n^\top \right\|_2 \|P_G\|_2 \|X\|_2 \|W^{(1)}\| := C$$

Provided that the loss function $\ell(\cdot; y)$ is Lipschitz-continuous, given N samples, we know from standard results that (see, e.g., Zhang [Zha23])

$$L(f) - \hat{L}(f) \lesssim \sqrt{\frac{C}{N}}.$$

Generalizing the simple example to MPNNs

Generalization bounds for MPNNs (informal) [JLS+23]

Suppose activations and loss function are all twice-di erentiable, Lipschitz-continuous, rst-order and second-order derivatives are both Lipschitz-continuous. d_i: number of neurons at layer i, for $i = 1; 2; : : : ; l$

With probability at least 1 over N samples, for any > 0, and any > 0, any GNN f with weights in H (recall equation (1)) satis es:

$$L(f) \quad (1 + \ )\hat{L}(f) + \sum_{i=1}^{l} \sqrt{\frac{d_i \max_{(X;G;y) D} kXk_2^2 kP_G k_2^{2(l \ 1)} r_i^2 \prod_{j=1} s_j^2}{N}} \qquad (11)$$

Example: for GCN, $P_G = D^{1=2}AD^{1=2}$, hence $kP_G k_2 \quad 1$

Open problem: remove the dependence on d_i to get size-independent sample complexity for GNNs? Known for feedforward NN [GRS18]

Generalizing the simple example to MPNNs

Generalization bounds for MPNNs (informal) [JLS+23]

Suppose activations and loss function are all twice-di erentiable, Lipschitz-continuous, rst-order and second-order derivatives are both Lipschitz-continuous. d_i : number of neurons at layer i, for $i = 1; 2; : : : ; l$

With probability at least 1 over N samples, for any > 0, and any > 0, any GNN f with weights in H (recall equation (1)) satis es:

$$L(f) \quad (1 + \ )\hat{L}(f) + \sum_{i=1}^{l} \sqrt{\frac{d_i \max_{(X;G;y) \ D} kX k_2^2 kP_G k_2^{2(l \ 1)} r_i^2 \prod_{j=1}^{l} s_j^2}{N}} \qquad (11)$$

Example: for GCN, $P_G = D^{\ 1=2}AD^{\ 1=2}$, hence $kP_G k_2 \quad 1$

Open problem: remove the dependence on d_i to get size-independent sample complexity for GNNs? Known for feedforward NN [GRS18]

Generalizing the simple example to MPNNs

Generalization bounds for MPNNs (informal) [JLS+23]

Suppose activations and loss function are all twice-di erentiable, Lipschitz-continuous, rst-order and second-order derivatives are both Lipschitz-continuous. d_i: number of neurons at layer i, for $i = 1; 2; : : : ; l$

With probability at least 1 over N samples, for any > 0, and any > 0, any GNN f with weights in H (recall equation (1)) satis es:

$$L(f) \quad (1 + \ )\hat{L}(f) + \sum_{i=1}^{l} \sqrt{\frac{d_i \max_{(X;G;y) \ D} kXk_2^2 kP_G k_2^{2(l \ 1)} r_i^2 \prod_{j=1}^{l} s_j^2}{N}} \qquad (11)$$

Example: for GCN, $P_G = D^{\ 1=2}AD^{\ 1=2}$, hence $kP_G k_2 \quad 1$

Open problem: remove the dependence on d_i to get size-independent sample complexity for GNNs? Known for feedforward NN [GRS18]

Numerical results

(a) Two-layer GCN (b) Four-layer GCN (c) Six-layer GCN

(d) Two-layer MPNN (e) Four-layer MPNN (f) Six-layer MPNN

Figure 5: Comparing our result and prior results Garg et al. [GJJ20] and Liao et al. [LUZ21] on three graph classi cation tasks conducted on GCNs and MPNNs

Conclusion

The use of Hessian to study neural networks seems to have been abandoned by the community due to their high computational costs

In this work, we propose to use Hessian of the loss surface as a measure of the generalization gap of neural networks and language models

Some partial progress

- Can explain a variety of phenomena observed with neural network training
- Provides a new approach to prove generalization bounds for GNNs

Next steps

- Better understand the structure of Hessian of loss surfaces in large models
- Known sample complexity for GNN only works for graph-level prediction; what about the sample complexity for learning node-level prediction tasks

Thank you for listening!

Conclusion

The use of Hessian to study neural networks seems to have been abandoned by the community due to their high computational costs

In this work, we propose to use Hessian of the loss surface as a measure of the generalization gap of neural networks and language models

Some partial progress

- Can explain a variety of phenomena observed with neural network training
- Provides a new approach to prove generalization bounds for GNNs

Next steps

- Better understand the structure of Hessian of loss surfaces in large models
- Known sample complexity for GNN only works for graph-level prediction; what about the sample complexity for learning node-level prediction tasks

Thank you for listening!

Conclusion

The use of Hessian to study neural networks seems to have been abandoned by the community due to their high computational costs

In this work, we propose to use Hessian of the loss surface as a measure of the generalization gap of neural networks and language models

Some partial progress

- Can explain a variety of phenomena observed with neural network training
- Provides a new approach to prove generalization bounds for GNNs

Next steps

- Better understand the structure of Hessian of loss surfaces in large models
- Known sample complexity for GNN only works for graph-level prediction; what about the sample complexity for learning node-level prediction tasks

Thank you for listening!

Conclusion

The use of Hessian to study neural networks seems to have been abandoned by the community due to their high computational costs

In this work, we propose to use Hessian of the loss surface as a measure of the generalization gap of neural networks and language models

Some partial progress

- Can explain a variety of phenomena observed with neural network training
- Provides a new approach to prove generalization bounds for GNNs

Next steps

- Better understand the structure of Hessian of loss surfaces in large models
- Known sample complexity for GNN only works for graph-level prediction; what about the sample complexity for learning node-level prediction tasks

Thank you for listening!

Conclusion

The use of Hessian to study neural networks seems to have been abandoned by the community due to their high computational costs

In this work, we propose to use Hessian of the loss surface as a measure of the generalization gap of neural networks and language models

Some partial progress

- Can explain a variety of phenomena observed with neural network training
- Provides a new approach to prove generalization bounds for GNNs

Next steps

- Better understand the structure of Hessian of loss surfaces in large models
- Known sample complexity for GNN only works for graph-level prediction; what about the sample complexity for learning node-level prediction tasks

Thank you for listening!

Conclusion

The use of Hessian to study neural networks seems to have been abandoned by the community due to their high computational costs

In this work, we propose to use Hessian of the loss surface as a measure of the generalization gap of neural networks and language models

Some partial progress

- Can explain a variety of phenomena observed with neural network training
- Provides a new approach to prove generalization bounds for GNNs

Next steps

- Better understand the structure of Hessian of loss surfaces in large models
- Known sample complexity for GNN only works for graph-level prediction; what about the sample complexity for learning node-level prediction tasks

Thank you for listening!

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{13).

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. (2018)\Stronger generalization bounds for deep nets via a compression approach". In: International Conference on Machine Learning PMLR, pp. 254{263 (6{13).

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017). \Spectrally-normalized margin bounds for neural networks". In: Advances in neural information processing systems30 (6{13).

Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A. (2020). \Benign over tting in linear regression". In: Proceedings of the National Academy of Sciences117.48, pp. 30063{30070 (6{13).

Catoni, O. (2007). \PAC-Bayesian supervised classi cation: the thermodynamics of statistical learning". In: arXiv preprint arXiv:0712.0248 (23{26).

Garg, V., Jegelka, S., and Jaakkola, T. (2020)\Generalization and representational limits of graph neural networks". In: ICML (46, 54).

Golowich, N., Rakhlin, A., and Shamir, O. (2018).\Size-independent sample complexity of neural networks". In: Conference On Learning Theory PMLR, pp. 297{299 (51{53).

Hamilton, W., Ying, Z., and Leskovec, J. (2017).\Inductive representation learning on large graphs". In: NeurIPS (46, 47).

Ju, H., Li, D., Sharma, A., and Zhang, H. R. (2023).\Generalization in graph neural networks: Improved pac-bayesian bounds on graph di usion". In: International Conference on Arti cial Intelligence and Statistics. PMLR, pp. 6314{6341 (46, 51{53).

Ju, H., Li, D., and Zhang, H. R. (2022). \Robust ne-tuning of deep neural networks with hessian-based generalization guarantees". In: International Conference on Machine Learning PMLR, pp. 10431{10461 (6{13).

Kipf, T. N. and Welling, M. (2017). \Semi-supervised classi cation with graph convolutional networks". In: ICLR (46, 47).

Li, D. and Zhang, H. (2021). \Improved regularization and robustness for ne-tuning in neural networks". In: Advances in Neural Information Processing Systems34, pp. 27249{27262 (22).

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* (6–13).

Liao, R., Urtasun, R., and Zemel, R. (2021). **"A PAC-Bayesian Approach to Generalization Bounds for Graph Neural Networks".** In: *ICLR* (46, 54).

Lot , S., Finzi, M., Kapoor, S., Potapczynski, A., Goldblum, M., and Wilson, A. G. (2022). **"PAC-Bayes compression bounds so tight that they can explain generalization".** In: *Advances in Neural Information Processing Systems* 35, pp. 31459–31473 (35).

McAllester, D. (2013). **"A PAC-Bayesian tutorial with a dropout bound".** In: *arXiv preprint arXiv:1307.2118* (23–26).

Nagarajan, V. and Kolter, J. Z. (2019). **"Uniform convergence may be unable to explain generalization in deep learning".** In: *Advances in Neural Information Processing Systems* 32 (22).

Shannon, C. E. (1948). **"A mathematical theory of communication".** In: *The Bell system technical journal* 27.3, pp. 379–423 (5).

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). **"How powerful are graph neural networks?"** In: *ICLR* (46, 47).

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).

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* (30–33, 37).

Zhang, T. (2023). **Mathematical analysis of machine learning algorithms.** Cambridge University Press (6–13, 48–50).