

Generalization of neural networks: A Hessian view

Hongyang (Ryan) Zhang
Northeastern University, Boston

November 6, 2024

Towards better understanding why machine learning models generalize well

Mathematical setup

- **Training:** given n samples from \mathcal{D} , $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$,

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

- **Test:** expected risk over a random sample of \mathcal{D}

$$L(f_W) = \mathbb{E}_{(x,y) \sim \mathcal{D}} [\ell(f_W(x), y)]$$

- **Generalization gap:** How should we think about $L(f_W) - \hat{L}(f_W)$?

Towards better understanding why machine learning models generalize well

Mathematical setup

- **Training:** given n samples from \mathcal{D} , $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$,

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

- **Test:** expected risk over a random sample of \mathcal{D}

$$L(f_W) = \mathbb{E}_{(x,y) \sim \mathcal{D}} [\ell(f_W(x), y)]$$

- **Generalization gap:** How should we think about $L(f_W) - \hat{L}(f_W)$?

Problem

Towards better understanding why machine learning models generalize well

Mathematical setup

- **Training:** given n samples from \mathcal{D} , $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$,

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

- **Test:** expected risk over a random sample of \mathcal{D}

$$L(f_W) = \mathbb{E}_{(x,y) \sim \mathcal{D}} [\ell(f_W(x), y)]$$

- Generalization gap: How should we think about $L(f_W) - \hat{L}(f_W)$?

Towards better understanding why machine learning models generalize well

Mathematical setup

- **Training:** given n samples from \mathcal{D} , $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$,

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

- **Test:** expected risk over a random sample of \mathcal{D}

$$L(f_W) = \mathbb{E}_{(x,y) \sim \mathcal{D}} [\ell(f_W(x), y)]$$

- **Generalization gap:** How should we think about $L(f_W) - \hat{L}(f_W)$?

Modern deep networks (e.g., ResNet, BERT) have more parameters than data labels to memorize training data [ZBH+21]

UNDERSTANDING DEEP LEARNING REQUIRES RE-THINKING 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.

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 bounds:** given a hypothesis space of neural nets \mathcal{H} , what is the worst-case generalization gap over \mathcal{H} ?
 - Rademacher complexity [BFT17]
 - PAC-Bayes [AGN+18; JLZ22]
- Convergence of SGD for minimizing nonconvex functions
 - 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 (now LoRA [HWA+22])
- High-dimensional statistical analysis
 - Precise asymptotics via random matrix theory [SC19]
 - Benign overfitting [BLL+20]

- **Generalization bounds:** given a hypothesis space of neural nets \mathcal{H} , what is the worst-case generalization gap over \mathcal{H} ?
 - **Rademacher complexity** [BFT17]
 - PAC-Bayes [AGN+18; JLZ22]
- **Convergence of SGD for minimizing nonconvex functions**
 - **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 (now LoRA [HWA+22])
- **High-dimensional statistical analysis**
 - **Precise asymptotics** via random matrix theory [SC19]
 - **Benign overfitting** [BLL+20]

- **Generalization bounds:** given a hypothesis space of neural nets \mathcal{H} , what is the worst-case generalization gap over \mathcal{H} ?
 - **Rademacher complexity** [BFT17]
 - **PAC-Bayes** [AGN+18; JLZ22]
- **Convergence of SGD for minimizing nonconvex functions**
 - **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 (now LoRA [HWA+22])
- **High-dimensional statistical analysis**
 - **Precise asymptotics via random matrix theory** [SC19]
 - **Benign overfitting** [BLL+20]

- **Generalization bounds:** given a hypothesis space of neural nets \mathcal{H} , what is the worst-case generalization gap over \mathcal{H} ?
 - **Rademacher complexity** [BFT17]
 - **PAC-Bayes** [AGN+18; JLZ22]
- **Convergence of SGD for minimizing nonconvex functions**
 - **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 (now LoRA [HWA+22])
- **High-dimensional statistical analysis**
 - **Precise asymptotics** via random matrix theory [SC19]
 - **Benign overfitting** [BLL+20]

- **Generalization bounds:** given a hypothesis space of neural nets \mathcal{H} , what is the worst-case generalization gap over \mathcal{H} ?
 - **Rademacher complexity** [BFT17]
 - **PAC-Bayes** [AGN+18; JLZ22]
- **Convergence of SGD for minimizing nonconvex functions**
 - **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 (now LoRA [HWA+22])
- **High-dimensional statistical analysis**
 - **Precise asymptotics** via random matrix theory [SC19]
 - **Benign overfitting** [BLL+20]

- **Generalization bounds:** given a hypothesis space of neural nets \mathcal{H} , what is the worst-case generalization gap over \mathcal{H} ?
 - **Rademacher complexity** [BFT17]
 - **PAC-Bayes** [AGN+18; JLZ22]
- **Convergence of SGD for minimizing nonconvex functions**
 - **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 (now LoRA [HWA+22])
- **High-dimensional statistical analysis**
 - **Precise asymptotics** via random matrix theory [SC19]
 - **Benign overfitting** [BLL+20]

A flavor of the existing results

Neural network setup

- f_W : A multi-layer neural net with weight matrices $W = [W_1, W_2, \dots, W_l]$
- \mathcal{D} : An unknown distribution over feature space $\mathcal{X} \times \mathcal{Y}$

Norm Bounds for Deep Networks

Hypothesis space

$$\mathcal{H} = \left\{ \|W_1\|_2 \leq s_1, \|W_2\|_2 \leq s_2, \dots, \|W_l\|_2 \leq s_l, \right. \\ \left. \|W_1\|_F \leq s_1 r_1, \|W_2\|_F \leq s_2 r_2, \dots, \|W_l\|_F \leq s_l r_l \right\} \quad (1)$$

- Bartlett et al. [BFT17]: For any l -layer deep net f_W whose weight matrices W belong to \mathcal{H} ,

$$L(f_W) - \hat{L}(f_W) \lesssim \sqrt{\frac{\left(\prod_{i=1}^l s_i^2\right) \sum_{i=1}^l r_i^2}{n}}$$

- Grows exponentially w/ depth! Tight in the worst-case

A flavor of the existing results

Neural network setup

- f_W : A multi-layer neural net with weight matrices $W = [W_1, W_2, \dots, W_l]$
- \mathcal{D} : An unknown distribution over feature space $\mathcal{X} \times \mathcal{Y}$

Norm Bounds for Deep Networks

- Hypothesis space

$$\mathcal{H} = \left\{ \begin{aligned} &\|W_1\|_2 \leq s_1, \|W_2\|_2 \leq s_2, \dots, \|W_l\|_2 \leq s_l; \\ &\|W_1\|_F \leq s_1 r_1, \|W_2\|_F \leq s_2 r_2, \dots, \|W_l\|_F \leq s_l r_l \end{aligned} \right\} \quad (1)$$

- Bartlett et al. [BFT17]: For any l -layer deep net f_W whose weight matrices W belong to \mathcal{H} ,

$$L(f_W) - \hat{L}(f_W) \lesssim \sqrt{\frac{\left(\prod_{i=1}^l s_i^2\right) \sum_{i=1}^l r_i^2}{n}}$$

- Grows exponentially w/ depth! **Tight** in the worst-case

A flavor of the existing results

Neural network setup

- f_W : A multi-layer neural net with weight matrices $W = [W_1, W_2, \dots, W_l]$
- \mathcal{D} : An unknown distribution over feature space $\mathcal{X} \times \mathcal{Y}$

Norm Bounds for Deep Networks

- Hypothesis space

$$\mathcal{H} = \left\{ \begin{aligned} &\|W_1\|_2 \leq s_1, \|W_2\|_2 \leq s_2, \dots, \|W_l\|_2 \leq s_l; \\ &\|W_1\|_F \leq s_1 r_1, \|W_2\|_F \leq s_2 r_2, \dots, \|W_l\|_F \leq s_l r_l \end{aligned} \right\} \quad (1)$$

- Bartlett et al. [BFT17]: For any l -layer deep net f_W whose weight matrices W belong to \mathcal{H} ,

$$L(f_W) - \hat{L}(f_W) \lesssim \sqrt{\frac{\left(\prod_{i=1}^l s_i^2\right) \sum_{i=1}^l r_i^2}{n}}$$

- Grows exponentially w/ depth! Tight in the worst-case

A flavor of the existing results

Neural network setup

- f_W : A multi-layer neural net with weight matrices $W = [W_1, W_2, \dots, W_l]$
- \mathcal{D} : An unknown distribution over feature space $\mathcal{X} \times \mathcal{Y}$

Norm Bounds for Deep Networks

- Hypothesis space

$$\mathcal{H} = \left\{ \begin{aligned} &\|W_1\|_2 \leq s_1, \|W_2\|_2 \leq s_2, \dots, \|W_l\|_2 \leq s_l; \\ &\|W_1\|_F \leq s_1 r_1, \|W_2\|_F \leq s_2 r_2, \dots, \|W_l\|_F \leq s_l r_l \end{aligned} \right\} \quad (1)$$

- Bartlett et al. [BFT17]: For any l -layer deep net f_W whose weight matrices W belong to \mathcal{H} ,

$$L(f_W) - \hat{L}(f_W) \lesssim \sqrt{\frac{\left(\prod_{i=1}^l s_i^2\right) \sum_{i=1}^l r_i^2}{n}}$$

- Grows exponentially w/ depth! **Tight** in the worst-case

Central thesis: Geometry of loss landscapes affects generalization [KMN+17]

Derivation of a Hessian measure relating to generalization

Main results

Numerical results

An algorithm to find flat minima

Hessian-based regularization

Grokking in arithmetic tasks

A use case: Graph neural networks

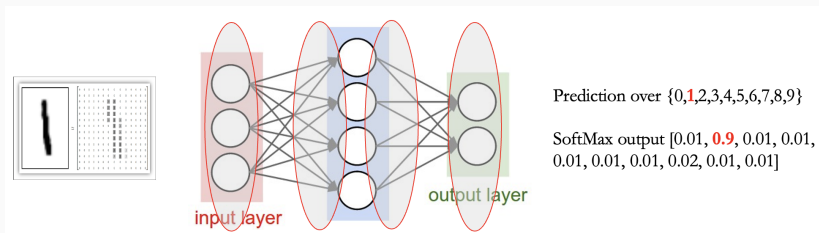
Setup and results

Numerical comparison

Feedforward neural networks

Forward pass over an l layer network with weight matrices W_1, W_2, \dots, W_l :

$$f_W(x) = \sigma_l \left(\dots \sigma_3 \left(W_3 \sigma_2 \left(W_2 \sigma_1 \left(W_1 x \right) \right) \right) \right)$$

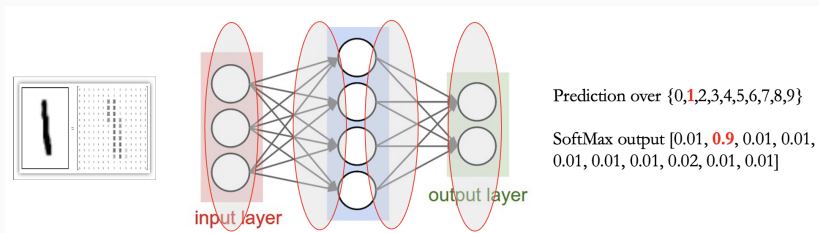


- Supervised learning: Train weights from random initialization
- Transfer learning: Adapt weights from pretrained/foundations models (fine-tuning)

Feedforward neural networks

Forward pass over an l layer network with weight matrices W_1, W_2, \dots, W_l :

$$f_W(x) = \sigma_l \left(\dots \sigma_3 \left(W_3 \sigma_2 \left(W_2 \sigma_1 \left(W_1 x \right) \right) \right) \right)$$

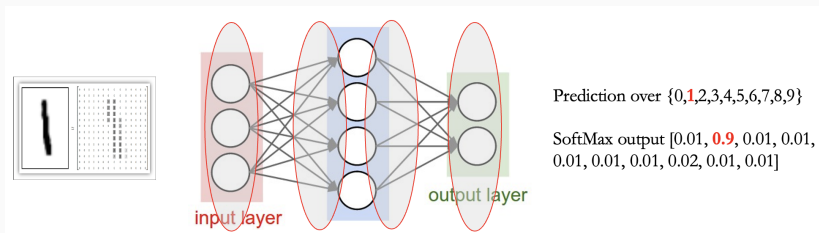


- Supervised learning: Train weights from random initialization
- Transfer learning: Adapt weights from pretrained/foundations models (fine-tuning)

Feedforward neural networks

Forward pass over an l layer network with weight matrices W_1, W_2, \dots, W_l :

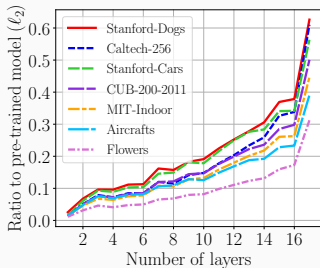
$$f_W(x) = \sigma_l \left(\dots \sigma_3 \left(W_3 \sigma_2 \left(W_2 \sigma_1 \left(W_1 x \right) \right) \right) \right)$$



- Supervised learning: Train weights from random initialization
- Transfer learning: Adapt weights from pretrained/foundations models (fine-tuning)

Factor #1: Measuring distance from initialization

Nagarajan and Kolter [NK19]: The distance between the initialization and the model can affect generalization—This corresponds to the norm of the hypothesis space



12v1 [cs.LG] 6 Nov 2014

How transferable are features in deep neural networks?

Jason Yosinski,¹ Jeff Clune,² Yoshua Bengio,³ and Hod Lipson⁴

¹ Dept. Computer Science, Cornell University

² Dept. Computer Science, University of Wyoming

³ Dept. Computer Science & Operations Research, University of Montreal

⁴ Dept. Mechanical & Aerospace Engineering, Cornell University

Abstract

Many deep neural networks trained on natural images exhibit a curious phenomenon in common: on the first layer they learn features similar to Gabor filters and color blobs. Such first-layer features appear not to be specific to a particular dataset or task, but general in that they are applicable to many datasets and tasks. Features must eventually transition from general to specific by the last layer of the network, but this transition has not been studied extensively. In this paper we experimentally quantify the generality versus specificity of neurons in each layer of a deep convolutional neural network and report a few surprising results. Transferability is negatively affected by two distinct issues: (1) the specialization of higher layer neurons to their original task at the expense of performance on the target task, which was expected, and (2) optimization difficulties related to splitting networks between co-adapted neurons, which was not expected. In an exam-

Theorem [McA13]

- $\mathcal{P} = \mathcal{N}(W^{(0)}, \sigma^2 \text{Id})$: prior distribution centered at the initialization weights
- $\mathcal{Q} = \mathcal{N}(W^{(T)}, \sigma^2 \text{Id})$: posterior distribution centered at the trained weights at epoch T
- With probability at least $1 - \delta$ for any $\delta \in (0, 1)$

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

Theorem [Cat07]

- For any $\beta \in (0, 1)$, with probability at least $1 - \delta$

$$\mathbb{E}_{w \sim \mathcal{Q}} [L(f_w)] \leq \frac{1}{\beta} \mathbb{E}_{w \sim \mathcal{Q}} [\hat{L}(f_w)] + \frac{KL(\mathcal{Q}||\mathcal{P}) + \log(\delta^{-1})}{2\beta(1-\beta)n} \quad (3)$$

Proof machinery: PAC-Bayes bound

Theorem [McA13]

- $\mathcal{P} = \mathcal{N}(W^{(0)}, \sigma^2 \text{Id})$: prior distribution centered at the initialization weights
- $\mathcal{Q} = \mathcal{N}(W^{(T)}, \sigma^2 \text{Id})$: posterior distribution centered at the trained weights at epoch T
- With probability at least $1 - \delta$ for any $\delta \in (0, 1)$

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

Theorem [Cat07]

- For any $\beta \in (0, 1)$, with probability at least $1 - \delta$

$$\mathbb{E}_{w \sim \mathcal{Q}} [L(f_w)] \leq \frac{1}{\beta} \mathbb{E}_{w \sim \mathcal{Q}} [\hat{L}(f_w)] + \frac{KL(\mathcal{Q}||\mathcal{P}) + \log(\delta^{-1})}{2\beta(1-\beta)n} \quad (3)$$

Theorem [McA13]

- $\mathcal{P} = \mathcal{N}(W^{(0)}, \sigma^2 \text{Id})$: prior distribution centered at the initialization weights
- $\mathcal{Q} = \mathcal{N}(W^{(T)}, \sigma^2 \text{Id})$: posterior distribution centered at the trained weights at epoch T
- With probability at least $1 - \delta$ for any $\delta \in (0, 1)$

$$\mathbb{E}_{w \sim \mathcal{Q}} [L(f_w)] \leq \mathbb{E}_{w \sim \mathcal{Q}} [\hat{L}(f_w)] + \sqrt{\frac{KL(\mathcal{Q} \parallel \mathcal{P}) + \log(4n\delta^{-1})}{n}} \quad (2)$$

Theorem [Cat07]

- For any $\beta \in (0, 1)$, with probability at least $1 - \delta$

$$\mathbb{E}_{w \sim \mathcal{Q}} [L(f_w)] \leq \frac{1}{\beta} \mathbb{E}_{w \sim \mathcal{Q}} [\hat{L}(f_w)] + \frac{KL(\mathcal{Q} \parallel \mathcal{P}) + \log(\delta^{-1})}{2\beta(1 - \beta)n} \quad (3)$$

Proof machinery: PAC-Bayes bound

Theorem [McA13]

- $\mathcal{P} = \mathcal{N}(W^{(0)}, \sigma^2 \text{Id})$: prior distribution centered at the initialization weights
- $\mathcal{Q} = \mathcal{N}(W^{(T)}, \sigma^2 \text{Id})$: posterior distribution centered at the trained weights at epoch T
- With probability at least $1 - \delta$ for any $\delta \in (0, 1)$

$$\mathbb{E}_{w \sim \mathcal{Q}} [L(f_w)] \leq \mathbb{E}_{w \sim \mathcal{Q}} [\hat{L}(f_w)] + \sqrt{\frac{KL(\mathcal{Q} \parallel \mathcal{P}) + \log(4n\delta^{-1})}{n}} \quad (2)$$

Theorem [Cat07]

- For any $\beta \in (0, 1)$, with probability at least $1 - \delta$

$$\mathbb{E}_{w \sim \mathcal{Q}} [L(f_w)] \leq \frac{1}{\beta} \mathbb{E}_{w \sim \mathcal{Q}} [\hat{L}(f_w)] + \frac{KL(\mathcal{Q} \parallel \mathcal{P}) + \log(\delta^{-1})}{2\beta(1 - \beta)n} \quad (3)$$

Claim #1: Relating noise stability to trace of the Hessian

Claim 1: Noise stability

Measure model stability after adding perturbations to weight parameters

[AGN+18]: Let $\ell_Q(f_W) = \mathbb{E}_U [\ell(f_{W+U})]$. We have

$$\left| \ell_Q(f_W(x), y) - \ell(f_W(x), y) - \frac{1}{2} \sigma^2 \text{Tr} [\nabla^2 [\ell(f_W(x), y)]] \right| \leq C_1 \sigma^3 \quad (4)$$

Claim #1: Relating noise stability to trace of the Hessian

Claim 1: Noise stability

Measure model stability after adding perturbations to weight parameters

[AGN+18]: Let $\ell_Q(f_w) = \mathbb{E}_U [\ell(f_{w+U})]$. We have

$$\left| \ell_Q(f_w(x), y) - \ell(f_w(x), y) - \frac{1}{2}\sigma^2 \text{Tr} \left[\nabla^2 [\ell(f_w(x), y)] \right] \right| \leq C_1 \sigma^3 \quad (4)$$

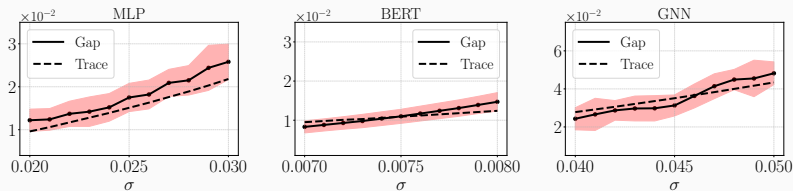


Figure 1: Illustration of the Hessian approximation. We report all measurements from the last epoch of fine-tuning. σ : standard deviation of the Gaussian noise injected into the weight matrices. σ decides the strength of regularization on the Hessian trace.

Claim #2: Uniform convergence of the Hessian operator

Claim 2: Uniform convergence

Since the Hessian operator is Lipschitz continuous, we can show that the trace of the Hessian satisfies the uniform convergence

Derivation of a Hessian measure

Proof sketch: start from PAC-Bayes bound

$$L_{\mathcal{Q}}(W) \leq \frac{1}{\beta} \hat{L}_{\mathcal{Q}}(W) + \frac{C(KL(\mathcal{Q}||\mathcal{P}) + \log(\delta^{-1}))}{2\beta(1-\beta)n} \quad (5)$$

C is a bound on the loss function. By Claim 1,

$$L_{\mathcal{Q}}(W) = L(W) + \frac{\sigma^2}{2} \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\text{Tr} \left[\nabla^2 \ell(f_W(x), y) \right] \right] + O(\sigma^3) \quad (6)$$

$$\hat{L}_{\mathcal{Q}}(W) = \hat{L}(W) + \frac{\sigma^2}{2n} \sum_{i=1}^n \text{Tr} \left[\nabla^2 \ell(f_W(x_i), y_i) \right] + O(\sigma^3) \quad (7)$$

By Claim 2, the difference between the second terms of equations (6), (7) is of order $O(n^{-1/2})$

By plugging in these results back to PAC-Bayes bound (5), we get:

$$L(W) \leq \frac{1}{\beta} \hat{L}(W) + \frac{\sigma^2(1-\beta)\alpha}{2\beta} + \frac{Cr^2/2\sigma^2}{2\beta(1-\beta)n} + O\left(\sigma^3 + \frac{\sigma^2\sqrt{\beta}}{\sqrt{n}} + \frac{\log(\delta^{-1})}{n}\right)$$

Derivation of a Hessian measure

Proof sketch: start from PAC-Bayes bound

$$L_{\mathcal{Q}}(W) \leq \frac{1}{\beta} \hat{L}_{\mathcal{Q}}(W) + \frac{C(KL(\mathcal{Q}||\mathcal{P}) + \log(\delta^{-1}))}{2\beta(1-\beta)n} \quad (5)$$

C is a bound on the loss function. By Claim 1,

$$L_{\mathcal{Q}}(W) = L(W) + \frac{\sigma^2}{2} \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\text{Tr} \left[\nabla^2 \ell(f_W(x), y) \right] \right] + O(\sigma^3) \quad (6)$$

$$\hat{L}_{\mathcal{Q}}(W) = \hat{L}(W) + \frac{\sigma^2}{2n} \sum_{i=1}^n \text{Tr} \left[\nabla^2 \ell(f_W(x_i), y_i) \right] + O(\sigma^3) \quad (7)$$

By Claim 2, the difference between the second terms of equations (6), (7) is of order $O(n^{-1/2})$

By plugging in these results back to PAC-Bayes bound (5), we get:

$$L(W) \leq \frac{1}{\beta} \hat{L}(W) + \frac{\sigma^2(1-\beta)\alpha}{2\beta} + \frac{Cr^2/2\sigma^2}{2\beta(1-\beta)n} + O\left(\sigma^3 + \frac{\sigma^2\sqrt{p}}{\sqrt{n}} + \frac{\log(\delta^{-1})}{n}\right)$$

Derivation of a Hessian measure

Proof sketch: start from PAC-Bayes bound

$$L_{\mathcal{Q}}(W) \leq \frac{1}{\beta} \hat{L}_{\mathcal{Q}}(W) + \frac{C(KL(\mathcal{Q}||\mathcal{P}) + \log(\delta^{-1}))}{2\beta(1-\beta)n} \quad (5)$$

C is a bound on the loss function. By Claim 1,

$$L_{\mathcal{Q}}(W) = L(W) + \frac{\sigma^2}{2} \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\text{Tr} \left[\nabla^2 \ell(f_W(x), y) \right] \right] + O(\sigma^3) \quad (6)$$

$$\hat{L}_{\mathcal{Q}}(W) = \hat{L}(W) + \frac{\sigma^2}{2n} \sum_{i=1}^n \text{Tr} \left[\nabla^2 \ell(f_W(x_i), y_i) \right] + O(\sigma^3) \quad (7)$$

By Claim 2, the difference between the second terms of equations (6), (7) is of order $O(n^{-1/2})$

By plugging in these results back to PAC-Bayes bound (5), we get:

$$L(W) \leq \frac{1}{\beta} \hat{L}(W) + \frac{\sigma^2(1-\beta)\alpha}{2\beta} + \frac{Cr^2/2\sigma^2}{2\beta(1-\beta)n} + O\left(\sigma^3 + \frac{\sigma^2\sqrt{p}}{\sqrt{n}} + \frac{\log(\delta^{-1})}{n}\right)$$

Derivation of a Hessian measure

Proof sketch: start from PAC-Bayes bound

$$L_{\mathcal{Q}}(W) \leq \frac{1}{\beta} \hat{L}_{\mathcal{Q}}(W) + \frac{C(KL(\mathcal{Q}||\mathcal{P}) + \log(\delta^{-1}))}{2\beta(1-\beta)n} \quad (5)$$

C is a bound on the loss function. By Claim 1,

$$L_{\mathcal{Q}}(W) = L(W) + \frac{\sigma^2}{2} \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\text{Tr} \left[\nabla^2 \ell(f_W(x), y) \right] \right] + O(\sigma^3) \quad (6)$$

$$\hat{L}_{\mathcal{Q}}(W) = \hat{L}(W) + \frac{\sigma^2}{2n} \sum_{i=1}^n \text{Tr} \left[\nabla^2 \ell(f_W(x_i), y_i) \right] + O(\sigma^3) \quad (7)$$

By Claim 2, the difference between the second terms of equations (6), (7) is of order $O(n^{-1/2})$

By plugging in these results back to PAC-Bayes bound (5), we get:

$$L(W) \leq \frac{1}{\beta} \hat{L}(W) + \frac{\sigma^2(1-\beta)\alpha}{2\beta} + \frac{Cr^2/2\sigma^2}{2\beta(1-\beta)n} + O\left(\sigma^3 + \frac{\sigma^2\sqrt{p}}{\sqrt{n}} + \frac{\log(\delta^{-1})}{n}\right)$$

Statement of results

Fact: $KL(Q||P) \leq \frac{r^2}{2\sigma^2}$, where r is radius of hypothesis space \mathcal{H}

By choosing σ^2 and β carefully, we obtain

Theorem 1 [ZLJ24]

Assume: ℓ is bounded between 0 and C , $\ell(f_W(\cdot), \cdot)$ is twice-differentiable and $\nabla^2[\ell(f_W(\cdot), \cdot)]$ is Lipschitz continuous. Suppose for any W in \mathcal{H} , the trace norm is less than α :

$$\alpha := \max_{W \in \mathcal{H}} \max_{(x,y) \sim \mathcal{D}} \text{Tr} \left[\nabla^2 \ell(f_W(x), y) \right], \quad (8)$$

and the ℓ_2 -norm of W is at most r for any $W \in \mathcal{H}$. Then, for any W in \mathcal{H} , with probability at least $1 - \delta$ for any $\delta > 0$, the following must hold:

$$L(W) \leq (1 + \epsilon) \hat{L}(W) + (1 + \epsilon) \sqrt{\frac{C\alpha r^2}{n}} + O\left(n^{-\frac{3}{4}} \log(\delta^{-1})\right). \quad (9)$$

Statement of results

Fact: $KL(Q||P) \leq \frac{r^2}{2\sigma^2}$, where r is radius of hypothesis space \mathcal{H}

By choosing σ^2 and β carefully, we obtain

Theorem 1 [ZLJ24]

Assume: ℓ is bounded between 0 and C , $\ell(f_W(\cdot), \cdot)$ is twice-differentiable and $\nabla^2[\ell(f_W(\cdot), \cdot)]$ is Lipschitz continuous. Suppose for any W in \mathcal{H} , the trace norm is less than α :

$$\alpha := \max_{W \in \mathcal{H}} \max_{(x,y) \sim \mathcal{D}} \text{Tr} \left[\nabla^2 \ell(f_W(x), y) \right], \quad (8)$$

and the ℓ_2 -norm of W is at most r for any $W \in \mathcal{H}$. Then, for any W in \mathcal{H} , with probability at least $1 - \delta$ for any $\delta > 0$, the following must hold:

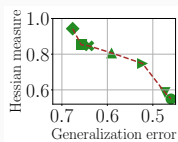
$$L(W) \leq (1 + \epsilon) \hat{L}(W) + (1 + \epsilon) \sqrt{\frac{C\alpha r^2}{n}} + O\left(n^{-\frac{3}{4}} \log(\delta^{-1})\right). \quad (9)$$

Does Hessian-based measures correlate well with empirical measurements?

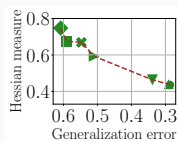
Numerical results

Does Hessian-based measures correlate well with empirical measurements?

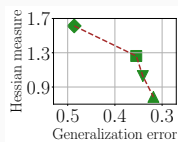
- Compare across seven fine-tuning methods: SGD, early stopping, weight decay, label smoothing, mixup, distance-based regularization, sharpness-aware minimization



(a) ResNet-50



(b) ResNet-50



(c) BERT-Base

Figure 2: The Hessian measures accurately correlate with empirical generalization errors for seven fine-tuning methods.

Does Hessian-based measures correlate well with empirical measurements?

Concurrent findings also at Lotfi et al. [LFK+22]

[cs.LG] 24 Nov 2022

PAC-Bayes Compression Bounds So Tight That They Can Explain Generalization

Sanae Lotfi* Marc Finzi* Sanyam Kapoor* Andres Potapczynski*

Micah Goldblum Andrew Gordon Wilson

New York University

Abstract

While there has been progress in developing non-vacuous generalization bounds for deep neural networks, these bounds tend to be uninformative about why deep learning works. In this paper, we develop a compression approach based on quantizing neural network parameters in a linear subspace, profoundly improving on previous results to provide state-of-the-art generalization bounds on a variety of tasks, including transfer learning. We use these tight bounds to better understand the role of model size, equivariance, and the implicit biases of optimization, for generalization in deep learning. Notably, we find large models can be compressed to a much greater extent than previously known, encapsulating Occam's razor. We also argue for data-independent bounds in explaining generalization.

Derivation of a Hessian measure relating to generalization

Main results

Numerical results

An algorithm to find flat minima

Hessian-based regularization

Grokking in arithmetic tasks

A use case: Graph neural networks

Setup and results

Numerical comparison

Noise injection induces a regularization of the Hessian

An algorithmic implication: Instead of minimizing the loss ℓ , we could minimize $\ell_{\mathcal{Q}} = \mathbb{E}_U [\ell(f_{W+U})]$ instead

Noise injection induces a regularization of the Hessian

An algorithmic implication: Instead of minimizing the loss ℓ , we could minimize $\ell_Q = \mathbb{E}_U [\ell(f_{W+U})]$ instead

By Claim 1 (equation (4)), this regularizes the trace of the Hessian of the loss surface

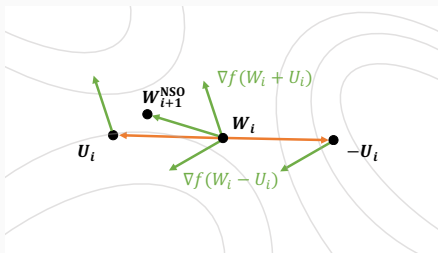


Figure 2: An illustration of one update step in our algorithm

Experimental results I

Comparison between SGD, naive noise injection (directly add noise before computing gradient, i.e., WP-SGD), and our algorithm (NSO)

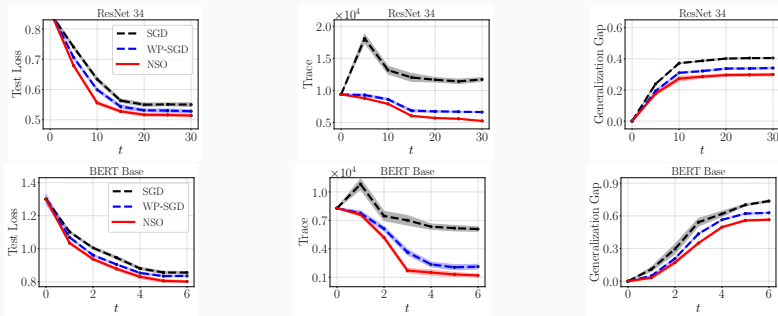


Figure 3: Fine-tuning ResNet-34 and BERT-Base, respectively, on an image and a text classification dataset

Similar results for more recent architectures (multi-modal, CLIP), chain-of-thought fine-tuning (LM, transformer) [ZLJ24]

Experimental results I

Comparison between SGD, naive noise injection (directly add noise before computing gradient, i.e., WP-SGD), and our algorithm (NSO)

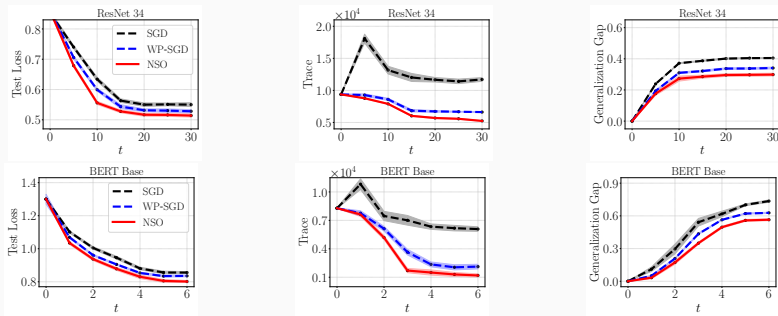


Figure 3: Fine-tuning ResNet-34 and BERT-Base, respectively, on an image and a text classification dataset

Similar results for more recent architectures (multi-modal, CLIP), chain-of-thought fine-tuning (LM, transformer) [ZLJ24]

Experimental results II

Grokking: A phenomenon of delayed generalization observed with training small arithmetic datasets [PBE+22]

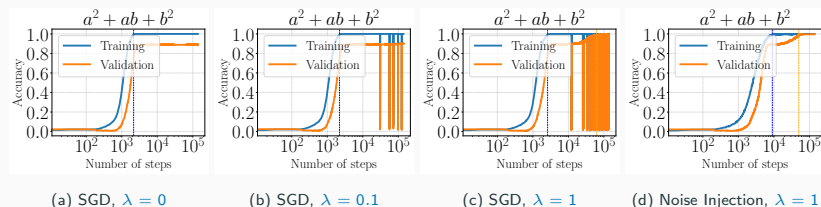


Figure 4: Training behavior with different weight decay (denoted as λ) and Hessian regularization (modulo 97).

Summary of our findings

- Fig. 4a: No weight decay; stable, but validation acc does not reach 100%
- Fig. 4b: Small weight decay; fluctuations, and validation acc does not reach 100%
- Fig. 4c: High weight decay; significant fluctuations, grokking is strong
- Fig. 4d: High weight decay plus noise injection to regularize Hessian; stable, strong grokking

Experimental results II

Grokking: A phenomenon of delayed generalization observed with training small arithmetic datasets [PBE+22]

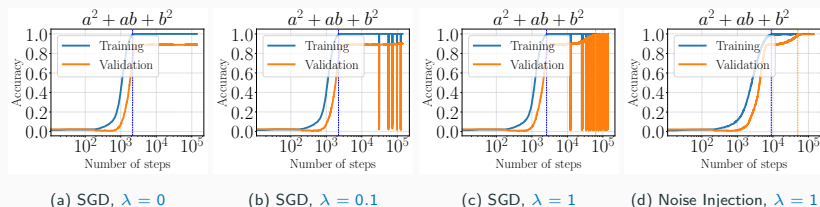


Figure 4: Training behavior with different weight decay (denoted as λ) and Hessian regularization (modulo 97).

Summary of our findings

- Fig. 4a: No weight decay; stable, but validation acc does not reach 100%
- Fig. 4b: Small weight decay; fluctuations, and validation acc does not reach 100%
- Fig. 4c: High weight decay; significant fluctuations, grokking is strong
- Fig. 4d: High weight decay plus noise injection to regularize Hessian; stable, strong grokking

Experimental results II

Grokking: A phenomenon of delayed generalization observed with training small arithmetic datasets [PBE+22]

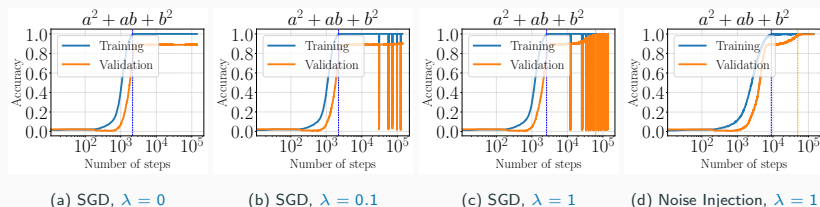


Figure 4: Training behavior with different weight decay (denoted as λ) and Hessian regularization (modulo 97).

Summary of our findings

- Fig. 4a: No weight decay; stable, but validation acc does not reach 100%
- Fig. 4b: Small weight decay; fluctuations, and validation acc does not reach 100%
- Fig. 4c: High weight decay; significant fluctuations, grokking is strong
- Fig. 4d: High weight decay plus noise injection to regularize Hessian; stable, strong grokking

Experimental results II

Grokking: A phenomenon of delayed generalization observed with training small arithmetic datasets [PBE+22]

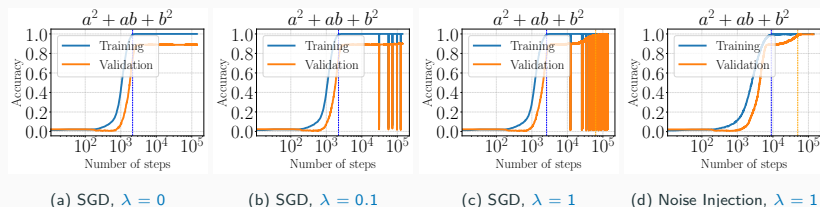


Figure 4: Training behavior with different weight decay (denoted as λ) and Hessian regularization (modulo 97).

Summary of our findings

- Fig. 4a: No weight decay; stable, but validation acc does not reach 100%
- Fig. 4b: Small weight decay; fluctuations, and validation acc does not reach 100%
- Fig. 4c: High weight decay; significant fluctuations, grokking is strong
- Fig. 4c: High weight decay plus noise injection to regularize Hessian; stable, strong grokking

Experimental results II

Grokking: A phenomenon of delayed generalization observed with training small arithmetic datasets [PBE+22]

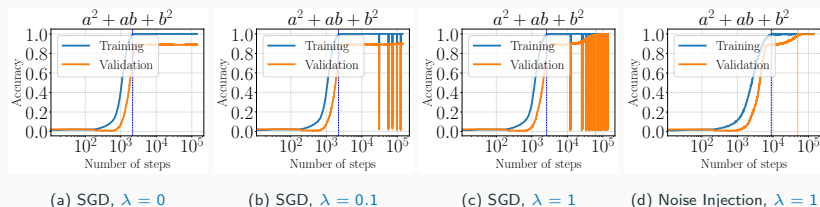


Figure 4: Training behavior with different weight decay (denoted as λ) and Hessian regularization (modulo 97).

Summary of our findings

- Fig. 4a: No weight decay; stable, but validation acc does not reach 100%
- Fig. 4b: Small weight decay; fluctuations, and validation acc does not reach 100%
- Fig. 4c: High weight decay; significant fluctuations, grokking is strong
- Fig. 4d: High weight decay plus noise injection to regularize Hessian; stable, strong grokking

Derivation of a Hessian measure relating to generalization

- Main results

- Numerical results

An algorithm to find flat minima

- Hessian-based regularization

- Grokking in arithmetic tasks

A use case: Graph neural networks

- Setup and results

- Numerical comparison

Message-passing neural networks

Motivation: Graph neural networks are one type of neural networks designed for working with graph-structured data

Notations:

- 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 $v \in V$: X is the feature matrix
- Graph-level prediction: a label $y \in \mathcal{Y}$ for each graph G

Setup: Message passing with graph diffusion matrix P_G

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

$$H^{(t)} = \phi_t \left(XU^{(t)} + \rho_t (P_G \psi_t(H^{(t-1)})) W^{(t)} \right), \text{ for } t = 1, 2, \dots, l \quad (10)$$

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

$$H^{(l)} = \frac{1}{n} \mathbf{1}_n^\top H^{(l-1)} W^{(l)}. \quad (11)$$

Message-passing neural networks

Motivation: Graph neural networks are one type of neural networks designed for working with graph-structured data

Notations:

- 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 $v \in V$: X is the feature matrix
- Graph-level prediction: a label $y \in \mathcal{Y}$ for each graph G

Setup: Message passing with graph diffusion matrix P_G

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

$$H^{(t)} = \phi_t \left(XU^{(t)} + \rho_t (P_G \psi_t(H^{(t-1)})) W^{(t)} \right), \text{ for } t = 1, 2, \dots, l \quad (10)$$

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

$$H^{(l)} = \frac{1}{n} \mathbf{1}_n^\top H^{(l-1)} W^{(l)}. \quad (11)$$

Message-passing neural networks

Motivation: Graph neural networks are one type of neural networks designed for working with graph-structured data

Notations:

- 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 $v \in V$: X is the feature matrix
- Graph-level prediction: a label $y \in \mathcal{Y}$ for each graph G

Setup: Message passing with graph diffusion matrix P_G

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

$$H^{(t)} = \phi_t \left(XU^{(t)} + \rho_t (P_G \psi_t(H^{(t-1)})) W^{(t)} \right), \text{ for } t = 1, 2, \dots, l \quad (10)$$

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

$$H^{(l)} = \frac{1}{n} \mathbf{1}_n^\top H^{(l-1)} W^{(l)}. \quad (11)$$

Message-passing neural networks

Motivation: Graph neural networks are one type of neural networks designed for working with graph-structured data

Notations:

- 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 $v \in V$: X is the feature matrix
- Graph-level prediction: a label $y \in \mathcal{Y}$ for each graph G

Setup: Message passing with graph diffusion matrix P_G

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

$$H^{(t)} = \phi_t \left(XU^{(t)} + \rho_t (P_G \psi_t(H^{(t-1)})) W^{(t)} \right), \text{ for } t = 1, 2, \dots, l \quad (10)$$

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

$$H^{(l)} = \frac{1}{n} \mathbf{1}_n^\top H^{(l-1)} W^{(l)}. \quad (11)$$

Message-passing neural networks

Motivation: Graph neural networks are one type of neural networks designed for working with graph-structured data

Notations:

- 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 $v \in V$: X is the feature matrix
- Graph-level prediction: a label $y \in \mathcal{Y}$ for each graph G

Setup: Message passing with graph diffusion matrix P_G

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

$$H^{(t)} = \phi_t \left(XU^{(t)} + \rho_t (P_G \psi_t(H^{(t-1)})) W^{(t)} \right), \text{ for } t = 1, 2, \dots, l \quad (10)$$

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

$$H^{(l)} = \frac{1}{n} \mathbf{1}_n^\top H^{(l-1)} W^{(l)}. \quad (11)$$

Summary of existing results

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

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

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

Graph Dependence	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	$\ A\ _2^{l-1}$	$\sum_{i=1}^{l-1} \frac{\ A\ _2^i}{i-1}$	$\ D^{-1}A\ _2^{l-1}$

Takeaway: Existing results scale with maximum degree d , whereas we reduce this to spectral norm of P_c (provably $\leq d$ —spectral graph theory!)

Summary of existing results

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

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

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

Graph Dependence	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	$\ A\ _2^{l-1}$	$\sum_{i=1}^{l-1} \frac{\ A\ _2^i}{l-1}$	$\ D^{-1}A\ _2^{l-1}$

Takeaway: Existing results scale with maximum degree d , whereas we reduce this to spectral norm of P_c (provably $\leq d$ —spectral graph theory!)

Summary of existing results

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

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

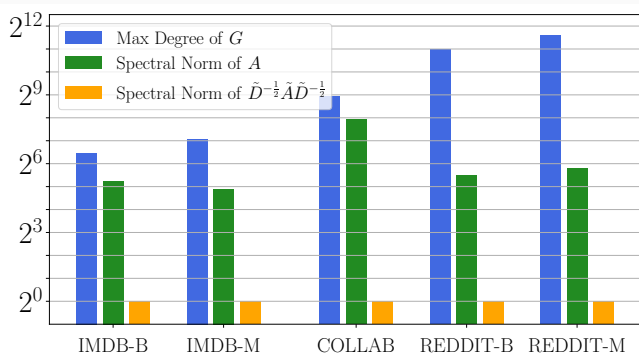
Table 1: Illustration of the dependence on graph structure of existing results. A : adjacency matrix, D : degree-diagonal matrix of A , l : depth of GNN

Graph Dependence	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	$\ A\ _2^{l-1}$	$\sum_{i=1}^{l-1} \frac{\ A\ _2^i}{l-1}$	$\ D^{-1}A\ _2^{l-1}$

Takeaway: Existing results scale with maximum degree d , whereas we reduce this to spectral norm of P_G (provably $\leq d$ —spectral graph theory!)

Illustration of graph statistics

Comparison on five social networks from SNAP



A toy example

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

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

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

$$\begin{aligned} \|f(X, G)\| &= \left\| \frac{1}{n} \mathbf{1}_n^\top P_G X W^{(1)} \right\| \\ &\leq \left\| \frac{1}{n} \mathbf{1}_n^\top \right\|_2 \cdot \|P_G\|_2 \cdot \|X\|_2 \cdot \|W^{(1)}\| := C \end{aligned}$$

Provided that the loss function $\ell(\cdot, y)$ is Lipschitz-continuous, given N samples, we know

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

A toy example

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

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

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

$$\begin{aligned} \|f(X, G)\| &= \left\| \frac{1}{n} \mathbf{1}_n^\top P_G X W^{(1)} \right\| \\ &\leq \left\| \frac{1}{n} \mathbf{1}_n^\top \right\|_2 \cdot \|P_G\|_2 \cdot \|X\|_2 \cdot \|W^{(1)}\| := C \end{aligned}$$

Provided that the loss function $\ell(\cdot, y)$ is Lipschitz-continuous, given N samples, we know

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

A toy example

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

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

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

$$\begin{aligned} \|f(X, G)\| &= \left\| \frac{1}{n} \mathbf{1}_n^\top P_G X W^{(1)} \right\| \\ &\leq \left\| \frac{1}{n} \mathbf{1}_n^\top \right\|_2 \cdot \|P_G\|_2 \cdot \|X\|_2 \cdot \|W^{(1)}\| := C \end{aligned}$$

Provided that the loss function $\ell(\cdot, y)$ is Lipschitz-continuous, given N samples, we know

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

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

Suppose: activations and loss function are all twice-differentiable, Lipschitz-continuous, first-order and second-order derivatives are both Lipschitz-continuous. d_i : number of neurons at layer i , for $i = 1, 2, \dots, l$

With probability at least $1 - \delta$ over N samples, for any $\delta > 0$, and any $\epsilon > 0$, any GNN f with weights in \mathcal{H} (recall equation (1)) satisfies:

$$L(f) \leq (1 + \epsilon)\hat{L}(f) + \sum_{i=1}^l \sqrt{\frac{d_i \left(\max_{(X, G, y) \sim \mathcal{D}} \|X\|_2^2 \|P_G\|_2^{2(l-1)} \right) \left(r_i^2 \prod_{j=1}^i s_j^2 \right)}{N}} \quad (12)$$

Example: for GCN, $P_G = D^{-1/2} A D^{-1/2}$, hence $\|P_G\|_2 \leq 1$

Key step: bound the trace of the Hessian recursively for each layer

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

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

Suppose: activations and loss function are all twice-differentiable, Lipschitz-continuous, first-order and second-order derivatives are both Lipschitz-continuous. d_i : number of neurons at layer i , for $i = 1, 2, \dots, l$

With probability at least $1 - \delta$ over N samples, for any $\delta > 0$, and any $\epsilon > 0$, any GNN f with weights in \mathcal{H} (recall equation (1)) satisfies:

$$L(f) \leq (1 + \epsilon)\hat{L}(f) + \sum_{i=1}^l \sqrt{\frac{d_i \left(\max_{(X, G, y) \sim \mathcal{D}} \|X\|_2^2 \|P_G\|_2^{2(l-1)} \right) \left(r_i^2 \prod_{j=1}^i s_j^2 \right)}{N}} \quad (12)$$

Example: for GCN, $P_G = D^{-1/2}AD^{-1/2}$, hence $\|P_G\|_2 \leq 1$

Key step: bound the trace of the Hessian recursively for each layer

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

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

Suppose: activations and loss function are all twice-differentiable, Lipschitz-continuous, first-order and second-order derivatives are both Lipschitz-continuous. d_i : number of neurons at layer i , for $i = 1, 2, \dots, l$

With probability at least $1 - \delta$ over N samples, for any $\delta > 0$, and any $\epsilon > 0$, any GNN f with weights in \mathcal{H} (recall equation (1)) satisfies:

$$L(f) \leq (1 + \epsilon)\hat{L}(f) + \sum_{i=1}^l \sqrt{\frac{d_i \left(\max_{(X, G, y) \sim \mathcal{D}} \|X\|_2^2 \|P_G\|_2^{2(l-1)} \right) \left(r_i^2 \prod_{j=1}^i s_j^2 \right)}{N}} \quad (12)$$

Example: for GCN, $P_G = D^{-1/2}AD^{-1/2}$, hence $\|P_G\|_2 \leq 1$

Key step: bound the trace of the Hessian recursively for each layer

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

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

Suppose: activations and loss function are all twice-differentiable, Lipschitz-continuous, first-order and second-order derivatives are both Lipschitz-continuous. d_i : number of neurons at layer i , for $i = 1, 2, \dots, l$

With probability at least $1 - \delta$ over N samples, for any $\delta > 0$, and any $\epsilon > 0$, any GNN f with weights in \mathcal{H} (recall equation (1)) satisfies:

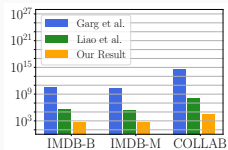
$$L(f) \leq (1 + \epsilon)\hat{L}(f) + \sum_{i=1}^l \sqrt{\frac{d_i \left(\max_{(X, G, y) \sim \mathcal{D}} \|X\|_2^2 \|P_G\|_2^{2(l-1)} \right) \left(r_i^2 \prod_{j=1}^i s_j^2 \right)}{N}} \quad (12)$$

Example: for GCN, $P_G = D^{-1/2}AD^{-1/2}$, hence $\|P_G\|_2 \leq 1$

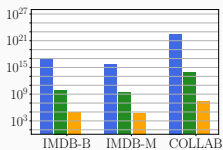
Key step: bound the trace of the Hessian recursively for each layer

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

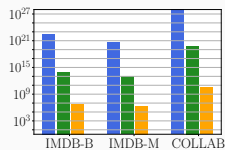
Numerical comparisons



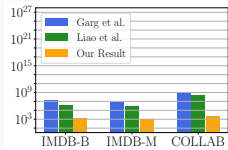
(a) Two-layer GCN



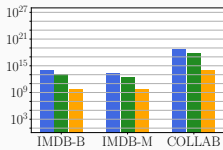
(b) Four-layer GCN



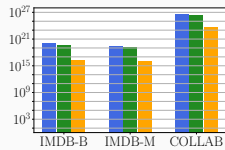
(c) Six-layer GCN



(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 classification tasks conducted on GCNs and MPNNs

Summary

The use of Hessian to study neural networks dates back to early work by Yann LeCun in the 1990s through the development of second-order methods [BL+88]

This work: [revisit this Hessian view to measure the generalization of neural networks](#)

- Can be used to explain a variety of phenomenon observed with neural network training, including grokking
- Provides a new approach to prove generalization bounds for GNNs by examining the Hessian

Open questions:

- Known sample complexity for GNN only works for graph-level prediction, how about a framework for node-level prediction?
- Better understand the structure of Hessian? Might also require faster algorithms to perform Hessian related computation

Thank you for listening!

Summary

The use of Hessian to study neural networks dates back to early work by Yann LeCun in the 1990s through the development of second-order methods [BL+88]

This work: revisit this Hessian view to measure the generalization of neural networks

- Can be used to explain a variety of phenomenon observed with neural network training, including grokking
- Provides a new approach to prove generalization bounds for GNNs by examining the Hessian

Open questions:

- Known sample complexity for GNN only works for graph-level prediction, how about a framework for node-level prediction?
- Better understand the structure of Hessian? Might also require faster algorithms to perform Hessian related computation

Thank you for listening!

Summary

The use of Hessian to study neural networks dates back to early work by Yann LeCun in the 1990s through the development of second-order methods [BL+88]

This work: revisit this Hessian view to measure the generalization of neural networks

- Can be used to explain a variety of phenomenon observed with neural network training, including grokking
- Provides a new approach to prove generalization bounds for GNNs by examining the Hessian

Open questions:

- Known sample complexity for GNN only works for graph-level prediction, how about a framework for node-level prediction?
- Better understand the structure of Hessian? Might also require faster algorithms to perform Hessian related computation

Thank you for listening!

Summary

The use of Hessian to study neural networks dates back to early work by Yann LeCun in the 1990s through the development of second-order methods [BL+88]

This work: revisit this Hessian view to measure the generalization of neural networks

- Can be used to explain a variety of phenomenon observed with neural network training, including grokking
- Provides a new approach to prove generalization bounds for GNNs by examining the Hessian

Open questions:

- Known sample complexity for GNN only works for graph-level prediction, how about a framework for node-level prediction?
- Better understand the structure of Hessian? Might also require faster algorithms to perform Hessian related computation

Thank you for listening!

Summary

The use of Hessian to study neural networks dates back to early work by Yann LeCun in the 1990s through the development of second-order methods [BL+88]

This work: revisit this Hessian view to measure the generalization of neural networks

- Can be used to explain a variety of phenomenon observed with neural network training, including grokking
- Provides a new approach to prove generalization bounds for GNNs by examining the Hessian

Open questions:

- Known sample complexity for GNN only works for graph-level prediction, how about a framework for node-level prediction?
- Better understand the structure of Hessian? Might also require faster algorithms to perform Hessian related computation

Thank you for listening!

Summary

The use of Hessian to study neural networks dates back to early work by Yann LeCun in the 1990s through the development of second-order methods [BL+88]




This work: revisit this Hessian view to measure the generalization of neural networks







- Can be used to explain a variety of phenomenon observed with neural network training, including grokking
- Provides a new approach to prove generalization bounds for GNNs by examining the Hessian







Open questions:







- Known sample complexity for GNN only works for graph-level prediction, how about a framework for node-level prediction?
- Better understand the structure of Hessian? Might also require faster algorithms to perform Hessian related computation

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 (8–13).
-  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 (8–13, 27, 28).
-  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* (8–17).
-  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 (8–13).
-  Becker, S., Le Cun, Y., et al. (1988). **“Improving the convergence of back-propagation learning with second order methods”**. In: *Proceedings of the 1988 connectionist models summer school*, pp. 29–37 (67–72).

-  Catoni, O. (2007). **“PAC-Bayesian supervised classification: 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* (55–57, 66).
-  Golowich, N., Rakhlin, A., and Shamir, O. (2018). **“Size-independent sample complexity of neural networks”**. In: *Conference On Learning Theory*. PMLR, pp. 297–299 (62–65).
-  Hamilton, W., Ying, Z., and Leskovec, J. (2017). **“Inductive representation learning on large graphs”**. In: *NeurIPS* (55–57).
-  Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2022). **“LoRA: Low-Rank Adaptation of Large Language Models”**. In: *International Conference on Learning Representations* (8–13).
-  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 (55–57, 62–65).

-  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 (8–13).
-  Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). **“On large-batch training for deep learning: Generalization gap and sharp minima”**. In: *ICLR* (18).
-  Kipf, T. N. and Welling, M. (2017). **“Semi-supervised classification with graph convolutional networks”**. In: *ICLR* (55–57).
-  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* (8–13).
-  Liao, R., Urtasun, R., and Zemel, R. (2021). **“A PAC-Bayesian Approach to Generalization Bounds for Graph Neural Networks”**. In: *ICLR* (55–57, 66).
-  Lotfi, 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 (38).

-  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).
-  Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. (2022). **“Grokking: Generalization beyond overfitting on small algorithmic datasets”**. In: *arXiv preprint arXiv:2201.02177* (44–48).
-  Sur, P. and Candès, E. J. (2019). **“A modern maximum-likelihood theory for high-dimensional logistic regression”**. In: *Proceedings of the National Academy of Sciences* 116.29, pp. 14516–14525 (8–13).
-  Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). **“How powerful are graph neural networks?”** In: *ICLR* (55–57).
-  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 (6).



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* (34, 35, 42, 43).