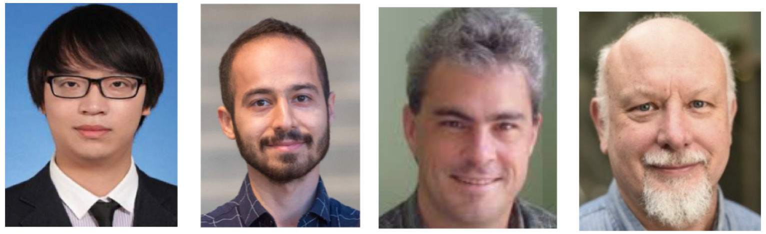
PyHessian: Neural Networks Through the Lens of the Hessian

Zhewei Yao, Amir Gholami, Kurt Keutzer, Michael W. Mahoney

Ph.D. Student, UC Berkeley

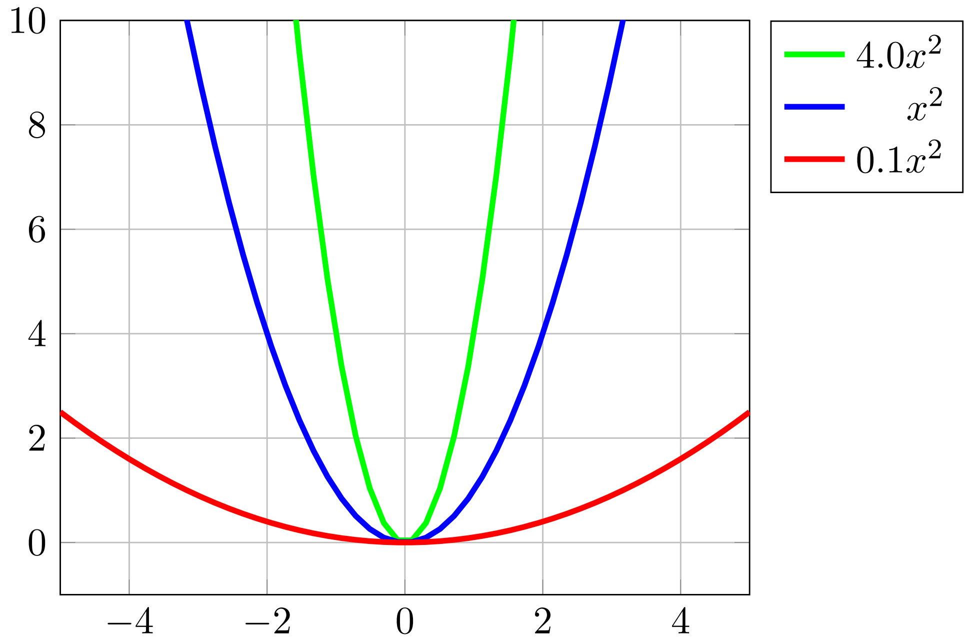
IEEE BigData, 2020

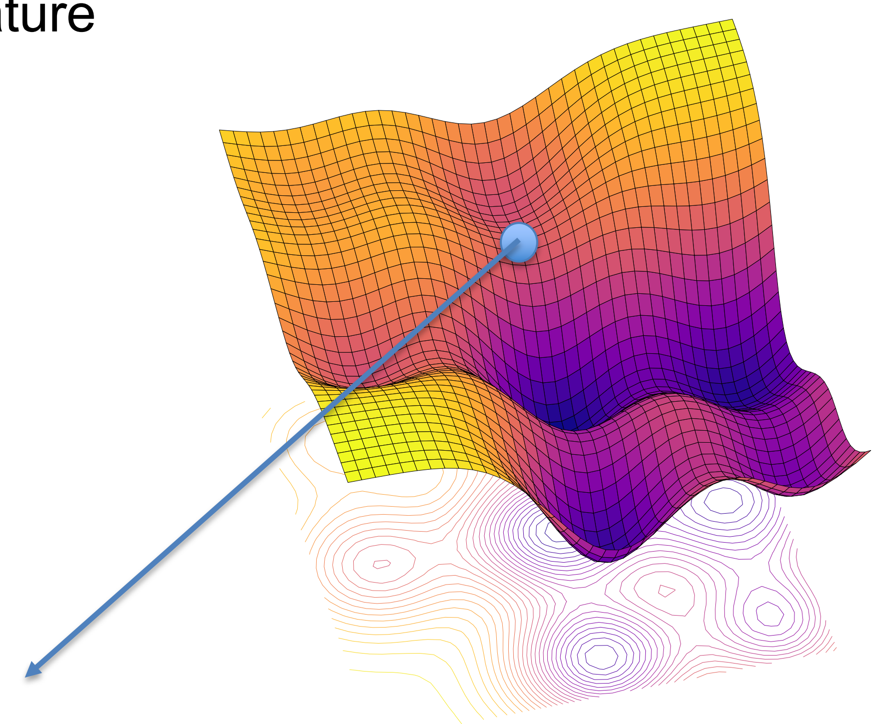


What is Second Order Information?

Second-order information

- Refers to information about a function gained by computing its second derivative
- Reveals information about the function's curvature





- At the origin, both the value and the first derivative of $y = 4x^2$, $y = x^2$, $y = 0.1x^2$ are all the same: 0
- But, the second derivatives give more information: 8 , 2, and 0.2 respectively

- Gradient is zero, but the current point is a saddle point, either minima or maxima

Z. Yao, P. Xu, F. Roosta-Khorasani, M. W. Mahoney, Inexact non-convex Newton-type methods, Informs Journal On Optimization

Executive Summary

PyHessian enables fast computation of Hessian information:

- Top-k eigenvalues and their corresponding eigenvectors (Power iteration)
- Trace (Hutchinson method)
- Full Spectral Distribution (Stochastic Lanczos algorithm)

As a use case, we analyzed

- The effect of BatchNorm (BN)
  - Shallow NN without BN has flatter Hessian spectrum
  - Removing BN results sharper Hessian spectrum in deep NNs
- The effect of Residual Connection:
  - NNs with residual connection always have flatter Hessian specturm

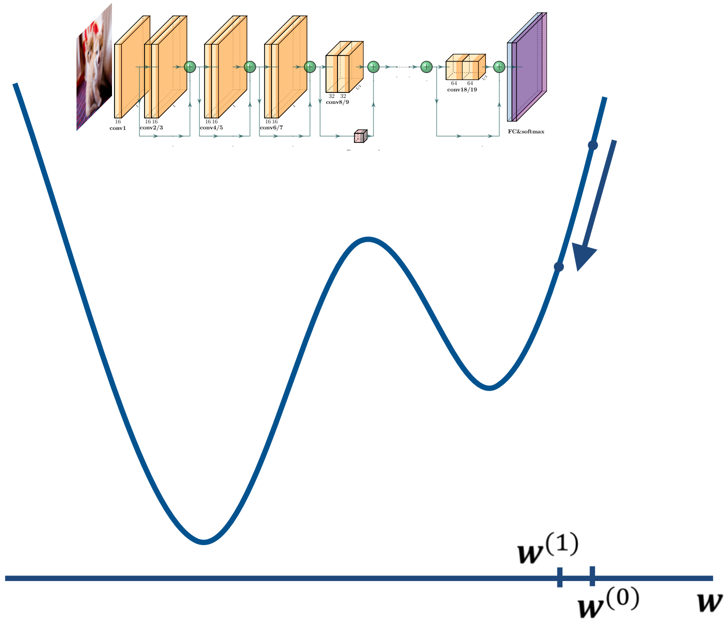
Z Yao, A Gholami, K Keutzer, M Mahoney, Hessian-based Analysis of Large Batch Training and Robustness to Adversaries, NeurIPS'2018
Z Yao, A Gholami, K Keutzer, M Mahoney, PyHessian: Neural Networks Through the Lens of the Hessian, Workshop at ICML'2020

Hessian for DNNs

Loss: $\min_{w} E = \sum_{i=1}^{N} l(f(x_i; w), y_i)$

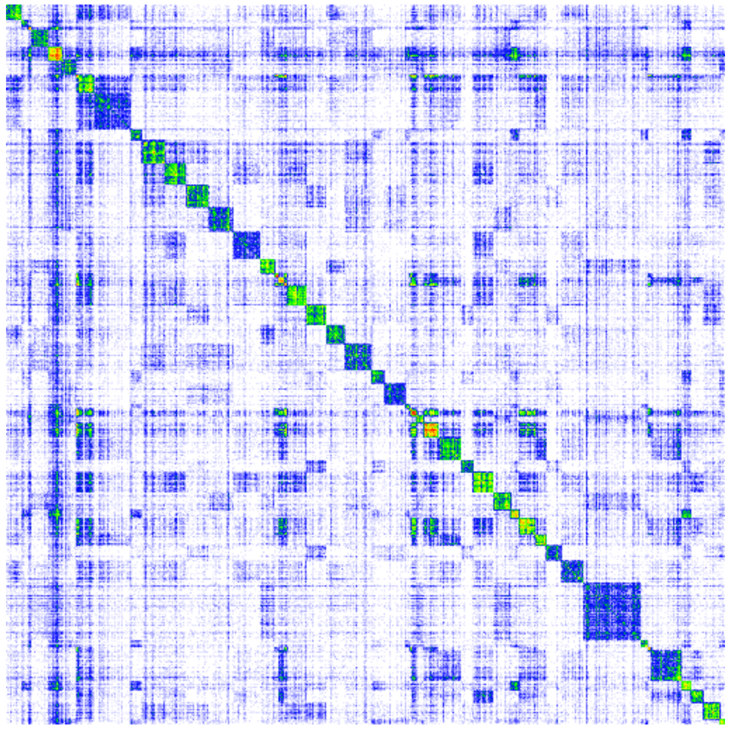
Gradient: $\frac{\partial E}{\partial w} \in \mathcal{R}^{|W|}$

Hessian: $\frac{\partial^2 E}{\partial w^2} \in \mathcal{R}^{|W| \times |W|}$



$|W|$

$|W|$



$|W|$

Forming the Hessian is computationally infeasible: For ResNet50 with 24M parameters, the Hessian is a matrix of size 24Mx24M (more than 2PB storage).

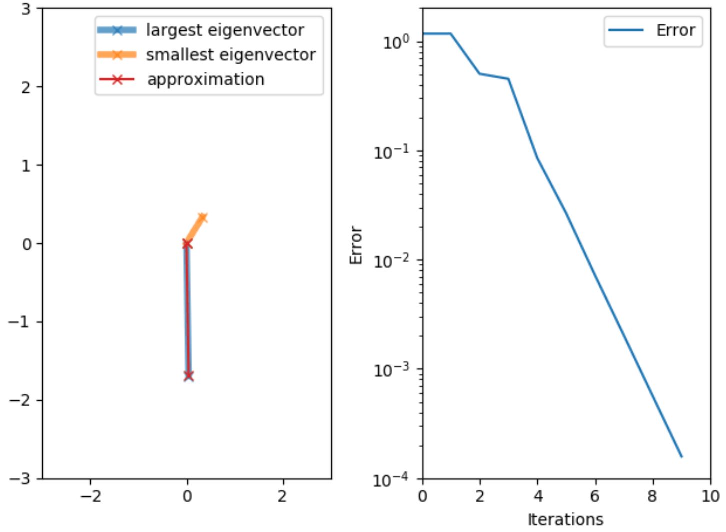
Hessian-vector Product

For a lot of applications, the explicit form of Hessian is not needed. The only requirement is the Hessian-vector product:

$$\frac{\partial g^T v}{\partial w} = \frac{\partial g^T}{\partial w} v + g^T \frac{\partial v}{\partial w} = \frac{\partial g^T}{\partial w} v = Hv.$$

Top eigenvalue (Power iteration):

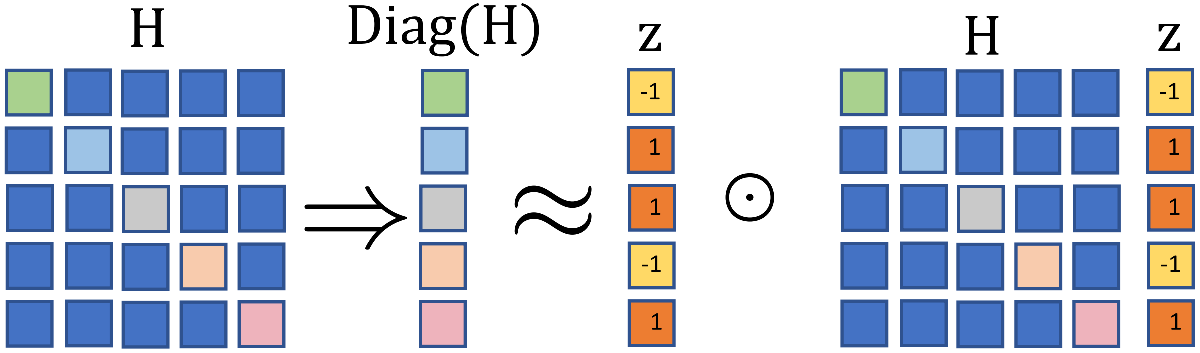
$$v_{k+1} = \frac{Hv_k}{\|Hv_k\|}$$



From Wikipedia

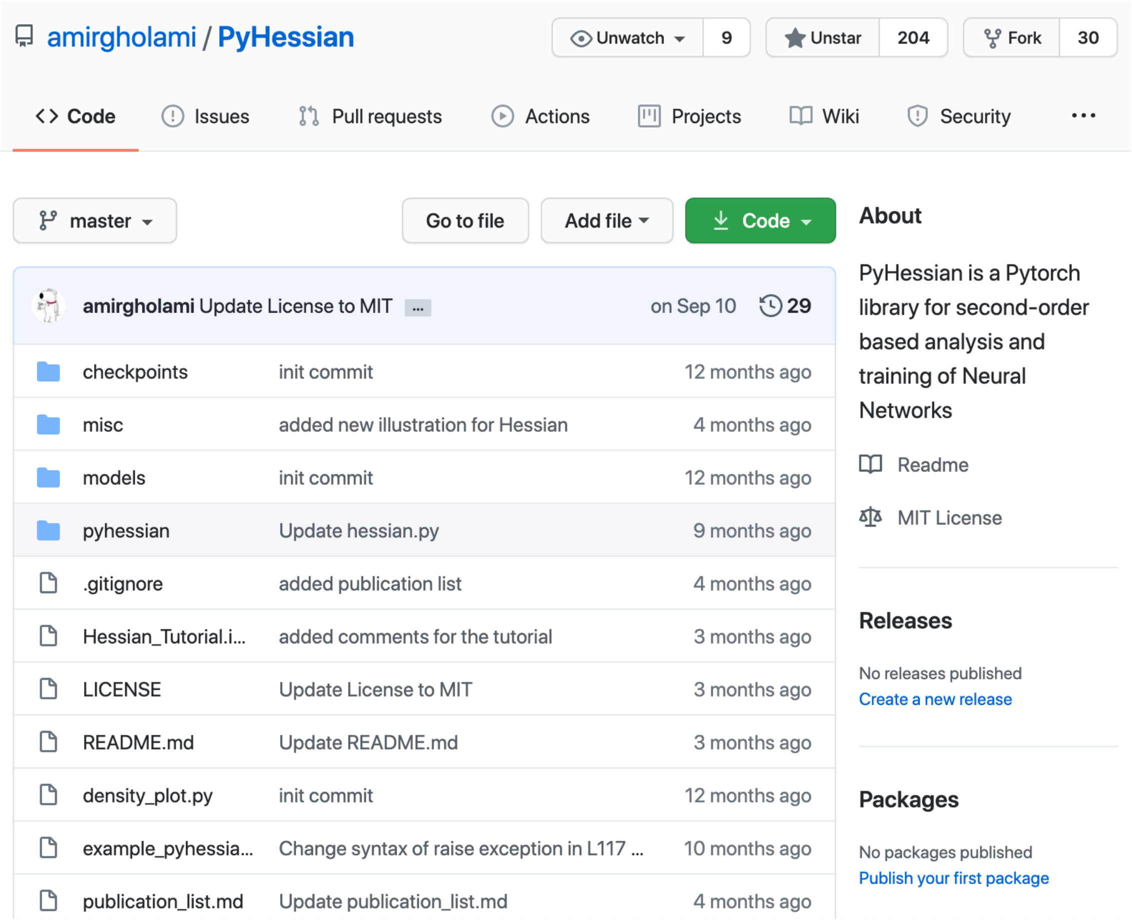
Hessian Trace (Hutchison method)

$$Trace(H) = \mathbb{E}_{z \sim \{-1,1\}}[z^T H z]$$



$$Trace(\mathrm{H}) = \mathbb{E}[\, z^T \mathrm{H} z]$$
$$\text{s.t.} \quad z \sim \text{Rademacher}(0.5)$$

PyHessian Library



PyHessian: https://github.com/amirgholami/PyHessian

PyHessian enables:

- Top-k Eigenvalues
- Hessian Trace
- Estimated Spectral Distribution

For a 1000 by 1000 matrix,
we use 20 iterations to compute its Hessian information

	Using Numpy	Using PyHessian	Relative Error
Top Eigenvalues	3958.4	3944.5	0.3%
Trace	1001574	1000153	0.1%
ESD (Used for Trace)	1001574	1005225	0.4%

BatchNorm in Deep Learning

- **BatchNorm** is one of **the key ingredients** for modern deep NNs
- When and why this popular architectural ingredient helps or hurts training/generalization is still largely unsolved

Algorithm 1 Batch Normalization (Every Iteration)

begin Forward Propagation:

Input: $\boldsymbol{X} \in \boldsymbol{R}^{B \times d}$

Output: $\boldsymbol{Y} \in \boldsymbol{R}^{B \times d}$

$\mu_B = \frac{1}{B} \sum_{i=1}^{B} \boldsymbol{x}_i$ // Get mini-batch mean

$\sigma_B^2 = \frac{1}{B} \sum_{i=1}^{B} (\boldsymbol{x}_i - \mu_B)^2$ // Get mini-batch variance

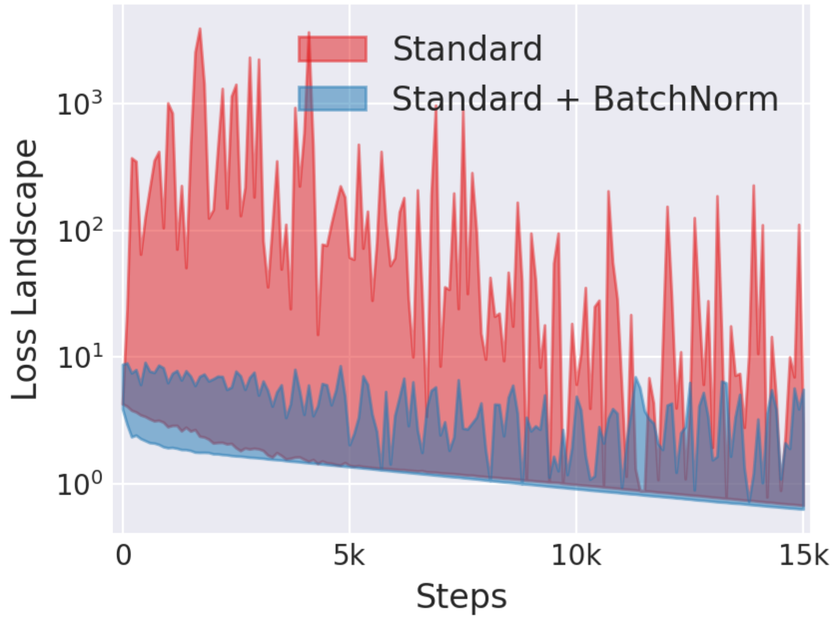
$\widecheck{\boldsymbol{X}} = \frac{\boldsymbol{X} - \mu_B}{\sigma_B}$ // Normalize

$\boldsymbol{Y} = \gamma \odot \widecheck{\boldsymbol{X}} + \beta$ // Scale and shift

$\mu = \alpha \mu + (1 - \alpha) \mu_B$ // Update running mean

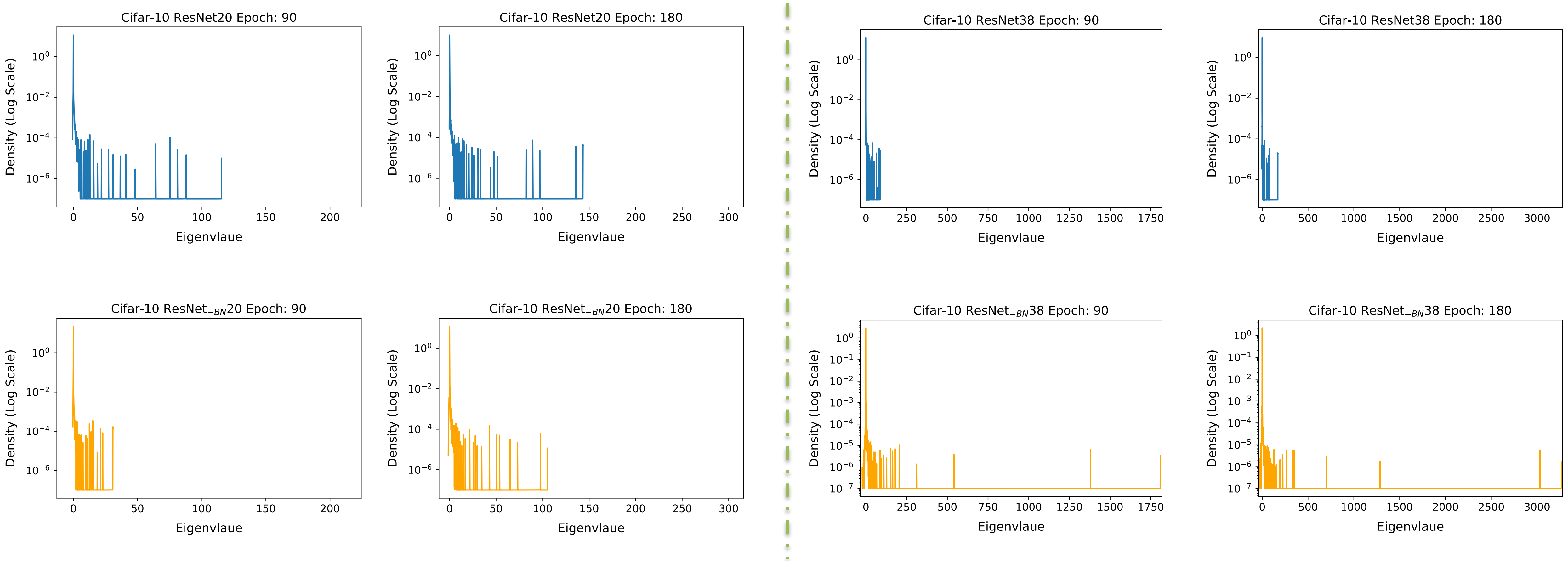
$\sigma^2 = \alpha \sigma^2 + (1 - \alpha) \sigma_B^2$ // Update running variance

One hypothesis is that BatchNorm can help **smooth** the loss landscape.



S Ioffe, C Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ICML'2015
Santurkar et al, How Does Batch Normalization Help Optimization? NeurIPS'18

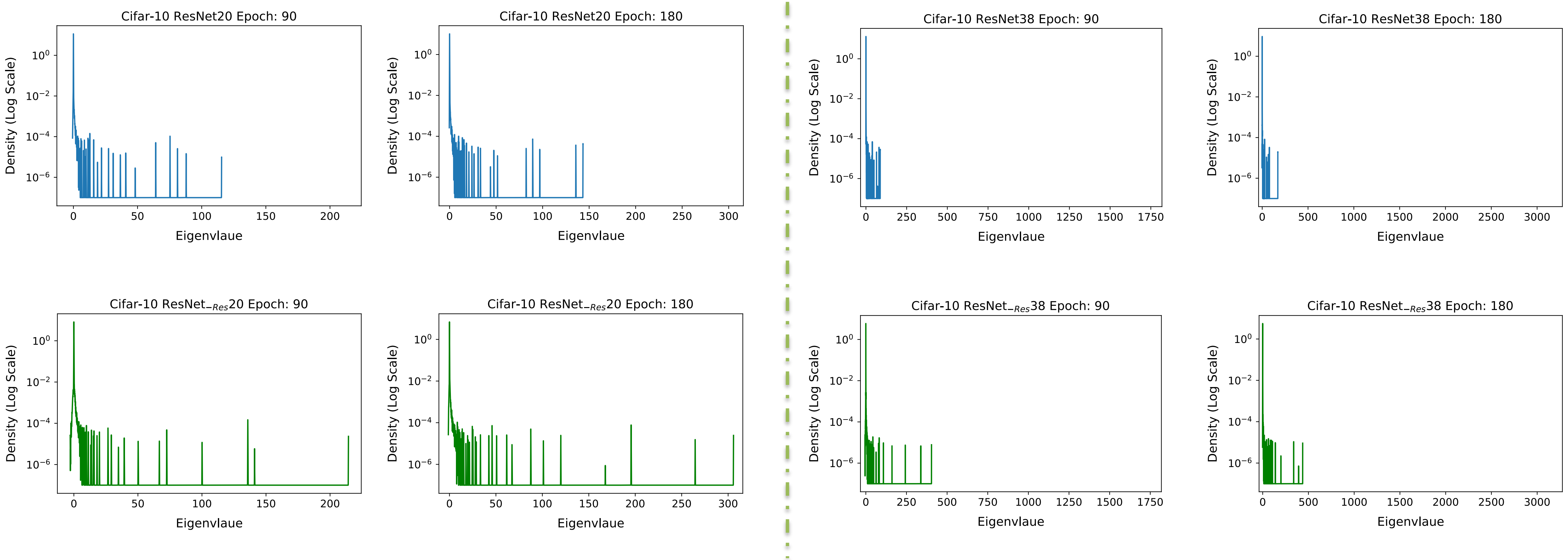
ESD of Shallow/Deep Neural Networks



- For shallow (left) networks, **NN without BatchNorm** has **flatter** Hessian spectrum
- For deep (right) networks, **NN with BatchNorm** has **flatter** Hessian spectrum

Z Yao, A Gholami, K Keutzer, M Mahoney, PyHessian: Neural Networks Through the Lens of the Hessian, Workshop at ICML’2020

ESD of Shallow/Deep Neural Networks



- NNs with residual connection typically have flatter Hessian spectrum.

Z Yao, A Gholami, K Keutzer, M Mahoney, PyHessian: Neural Networks Through the Lens of the Hessian, Workshop at ICML'2020

Usage in other Papers

PyHessian has been used

- as an analysis tool:
  - Yang et al., G-DAUG: Generative Data Augmentation for Commonsense Reasoning, arxiv: 2004.11546
- as a second order method tool:
  - Yao et al., ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning, arxiv: 2006.00719

Model	**IWSLT14** small	**WMT14** base
SGD	$28.57 \pm .15$	26.04
AdamW [34]	$35.66 \pm .11$	28.19
ADAHESSIAN	$\mathbf{35.79 \pm .06}$	**28.52**

Model	**PTB** Three-Layer	**Wikitext-103** Six-Layer
SGD	59.9 ± 3.0	78.5
AdamW [34]	54.2 ± 1.6	20.9
ADAHESSIAN	$\mathbf{51.5 \pm 1.2}$	**19.9**

Thank You

Please contact us if you have any questions:

{zheweiy, amirgh} @ berkeley.edu

Paper link: https://arxiv.org/pdf/1912.07145.pdf

Code link: https://github.com/amirgholami/PyHessian