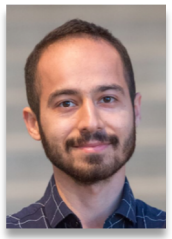
ADAHESSIAN and PyHessian Tutorial

Zhewei Yao, **Amir Gholami**, Kurt Keutzer, Michael Mahoney

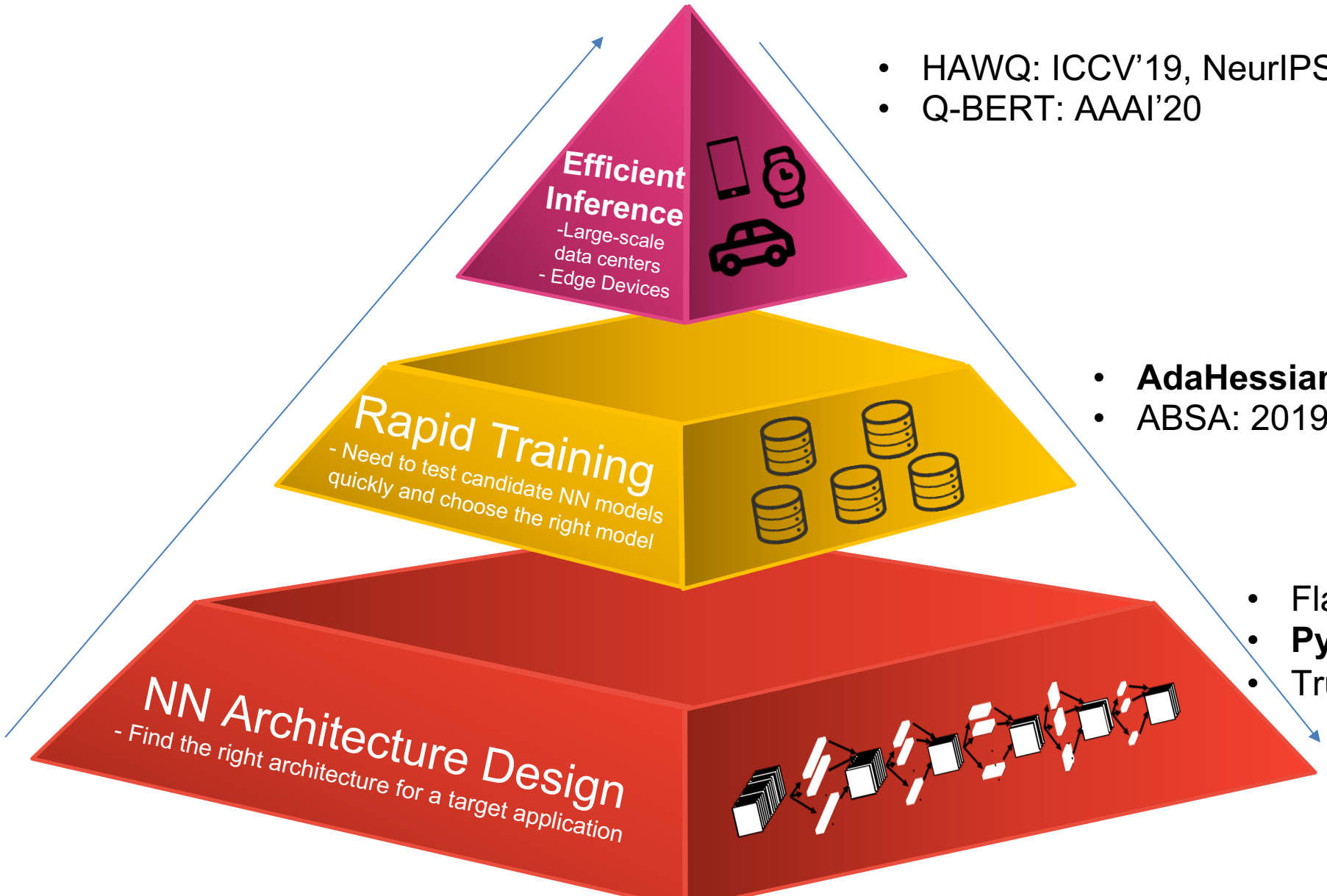
Rise Camp 2020

Berkeley
UNIVERSITY OF CALIFORNIA

riselab
UC Berkeley

Second Order Method for DNNs



- HAWQ: ICCV'19, NeurIPS'19
- Q-BERT: AAAI'20

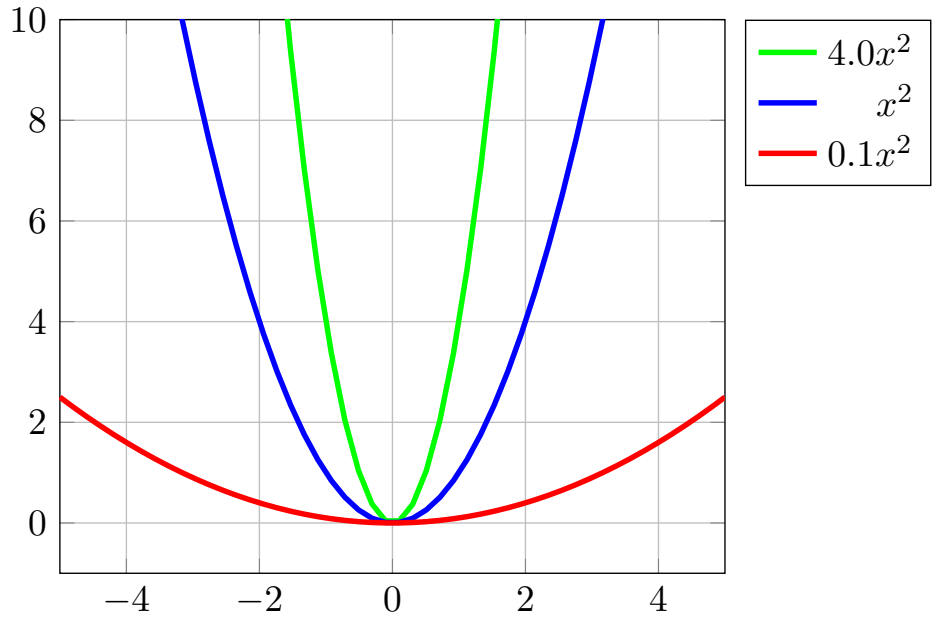
- **AdaHessian: 2020**
- ABSA: 2019

- Flat/Sharp Local Minima: NeurIPS'18
- **PyHessian: ICML'20**
- Trust Region Adversarial: CVPR'20

Quantifying “Sharpness”

Second-order information

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

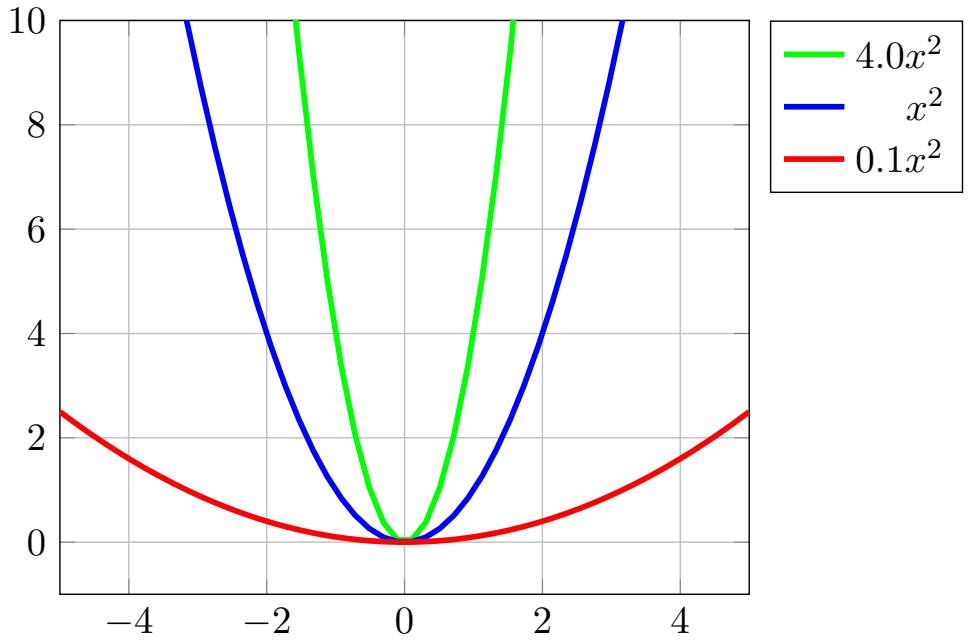


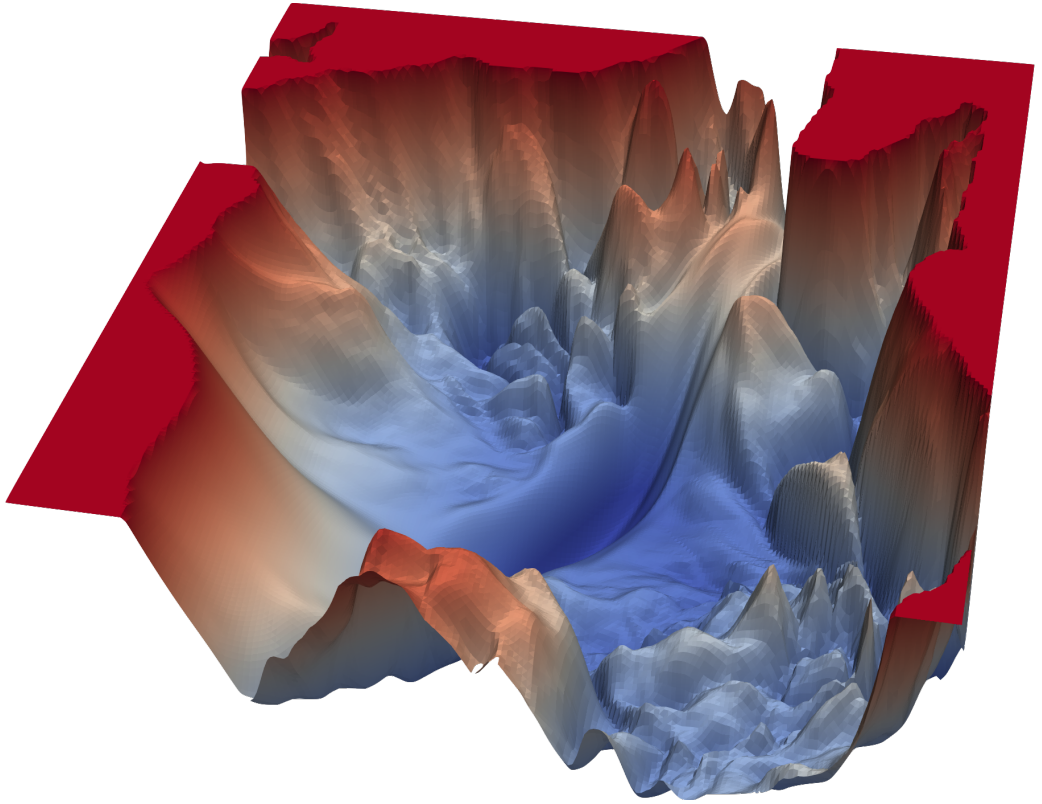
- At the origin, the first derivative of $y = 4x^2$, $y = x^2$, $y = 0.1\,x^2$ is all the same: 0
- The **second derivative** give more information: 8 , 2, and 0.2 respectively

What is second order information?

Second-order information

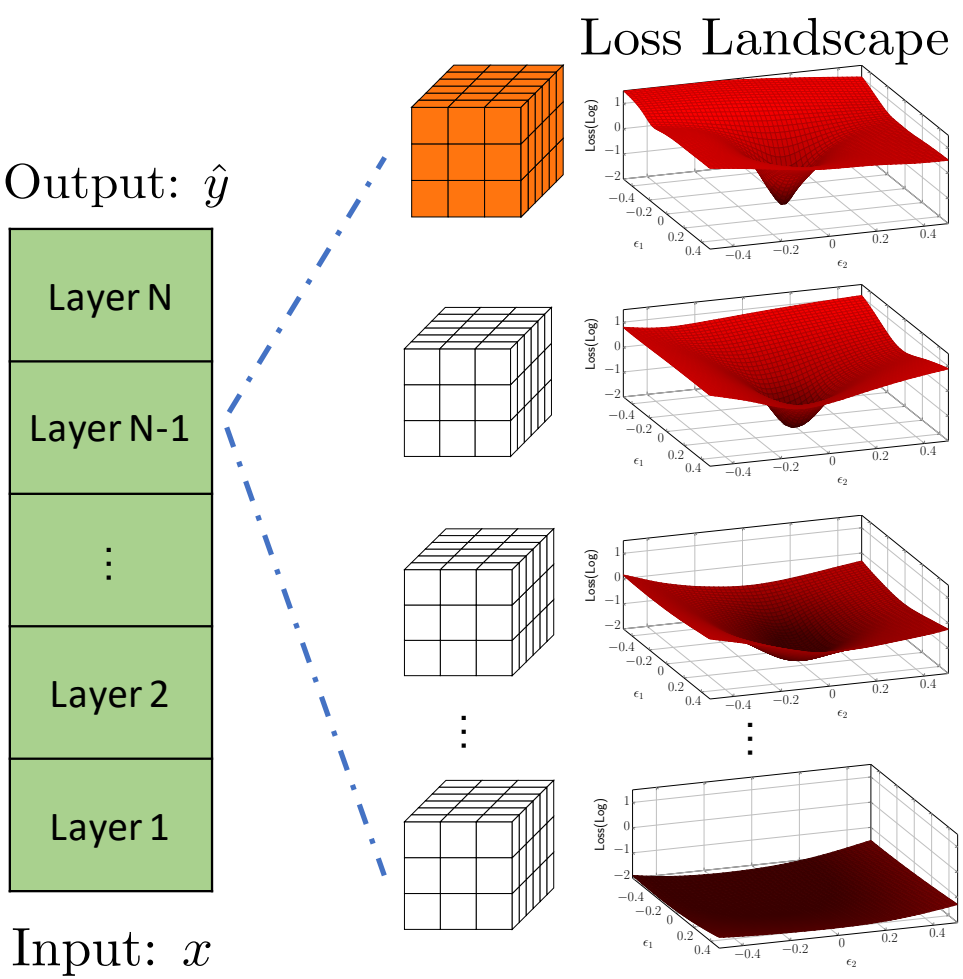
- Refers to information about a function gained by computing its second derivative
- Reveals information about the function's curvature.





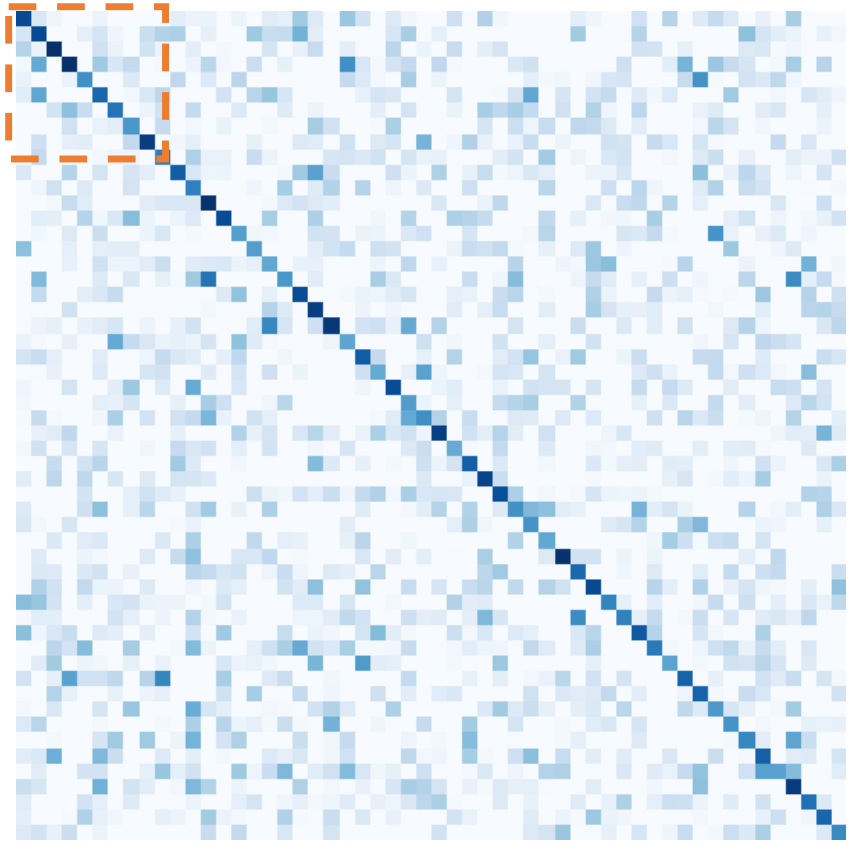
Li H, Xu Z, Taylor G, Studer C, Goldstein T. Visualizing the loss landscape of neural nets. NeurIPS'18, 2018

Opening the Black Box with Second Derivative



Gradient: $\mathbf{g} \in \mathbb{R}^{\mathbf{d}}$

Hessian: $\mathbf{H} \in \mathbb{R}^{\mathbf{d} \times \mathbf{d}}$



Pearlmutter BA. Fast exact multiplication by the Hessian. Neural computation. 1994.
Z. Yao*, A. Gholami*, Q. Lei, K. Keutzer, M. Mahoney, Hessian-based Analysis of Large Batch Training and Robustness to Adversaries, NeurIPS'18, 2018.
Z. Yao*, A. Gholami*, K. Keutzer, M. Mahoney, PyHessian: Neural Networks Through the Lens of the Hessian **Spotlight at ICML'20 workshop** on Beyond First-Order Optimization Methods in Machine Learning, 2020.
Code: https://github.com/amirgholami/PyHessian

Using Hessian Diagonal

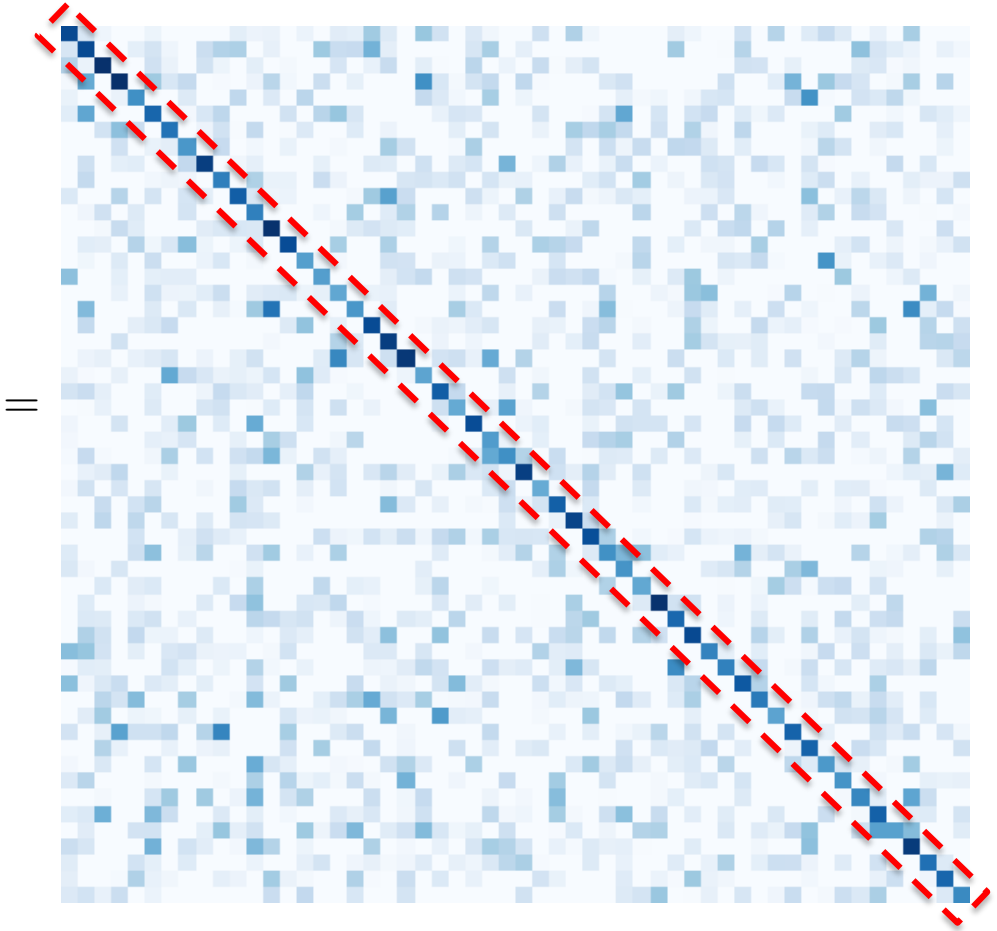
Forming the Hessian is computationally infeasible:

For ResNet50 with 24M params Hessian is a matrix of size **24Mx24M**

But what if we just approximate the Hessian?

Idea: Use Hessian diagonal

$$g = \quad Diag(H) =$$



Pearlmutter BA. Fast exact multiplication by the Hessian. Neural computation. 1994.
Costas Bekas, Effrosyni Kokiopoulou, and Yousef Saad. An estimator for the diagonal of a matrix. Applied numerical mathematics, 57(11-12):1214– 1229, 2007
Z. Yao*, A. Gholami*, Q. Lei, K. Keutzer, M. Mahoney, Hessian-based Analysis of Large Batch Training and Robustness to Adversaries, NeurIPS'18, 2018.
Z. Yao*, A. Gholami*, K. Keutzer, M. Mahoney, PyHessian: Neural Networks Through the Lens of the Hessian, Spotlight at ICML'20 workshop on Beyond First-Order Optimization Methods in Machine Learning Workshop, 2020.

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

Hutchinson's Algorithm

Algorithm 4: Hutchinson Method for Trace Computation

Input: Parameter: θ.

Compute the gradient of θ by backpropagation, *i.e.*, compute $g_\theta = \frac{dL}{d\theta}$.

for $i = 1, 2, \ldots$ **do** // Hutchinson Steps

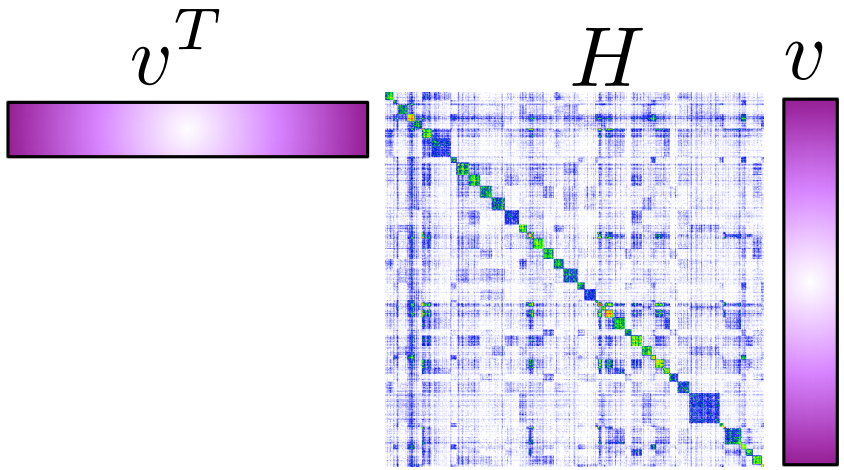
  Draw a random vector v from Rademacher distribution (same dimension as θ).

  Compute $gv = g_\theta^T v$

  Compute Hv by backpropagation, $Hv = \frac{d(gv)}{d\theta}$

  Compute and record $v^T H v$

Return the average of all computed $v^T H v$.

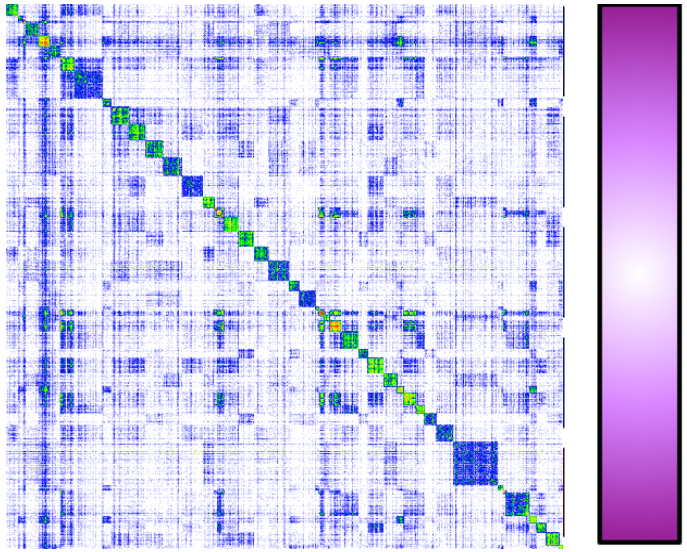


Z. Yao*, A. Gholami*, Q. Lei, K. Keutzer, M. Mahoney, Hessian-based Analysis of Large Batch Training and Robustness to Adversaries, NeurIPS'18, 2018.
Z. Yao*, A. Gholami*, K. Keutzer, M. Mahoney, PyHessian: Neural Networks Through the Lens of the Hessian, arxiv:1912.07145
Code: https://github.com/amirgholami/PyHessian

How to Compute Sharpness?

- How to compute Hessian matrix vector multiply without forming Hessian?

H v



$$g = \frac{dL}{d\theta}$$

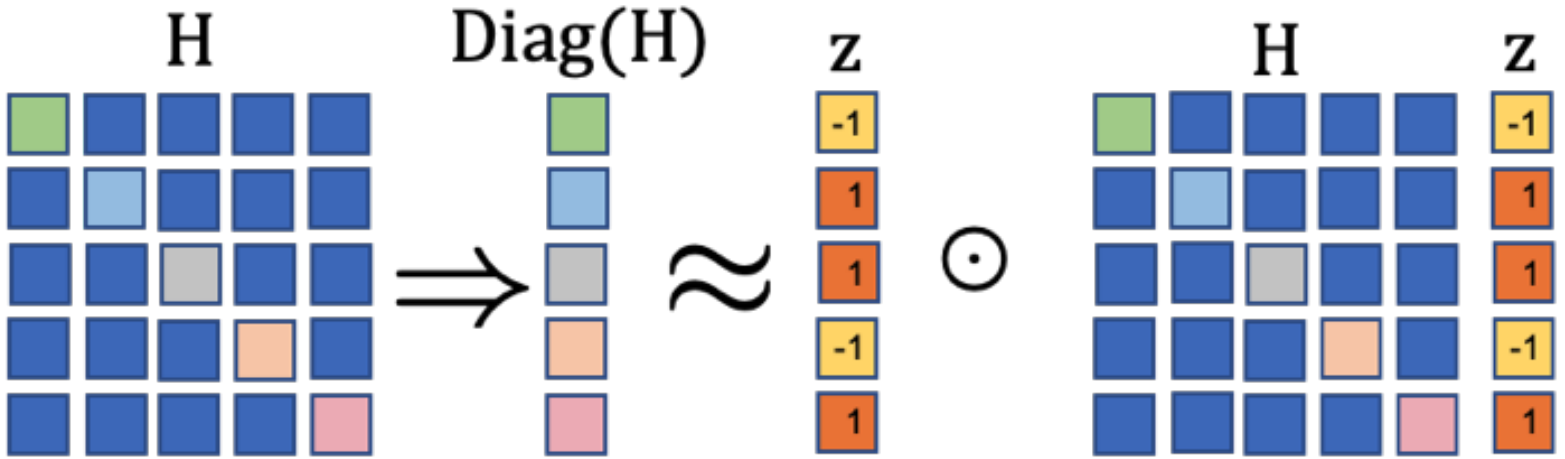
$$Hv = \frac{dg^T v}{d\theta} = \frac{dg^T}{d\theta} v + g^T \frac{dv}{d\theta} = \frac{dg^T}{d\theta} v$$

Z. Yao*, A. Gholami*, Q. Lei, K. Keutzer, M. Mahoney, Hessian-based Analysis of Large Batch Training and Robustness to Adversaries, NeurIPS'18, 2018.
Code: https://github.com/amirgholami/PyHessian

Hessian-vector Product

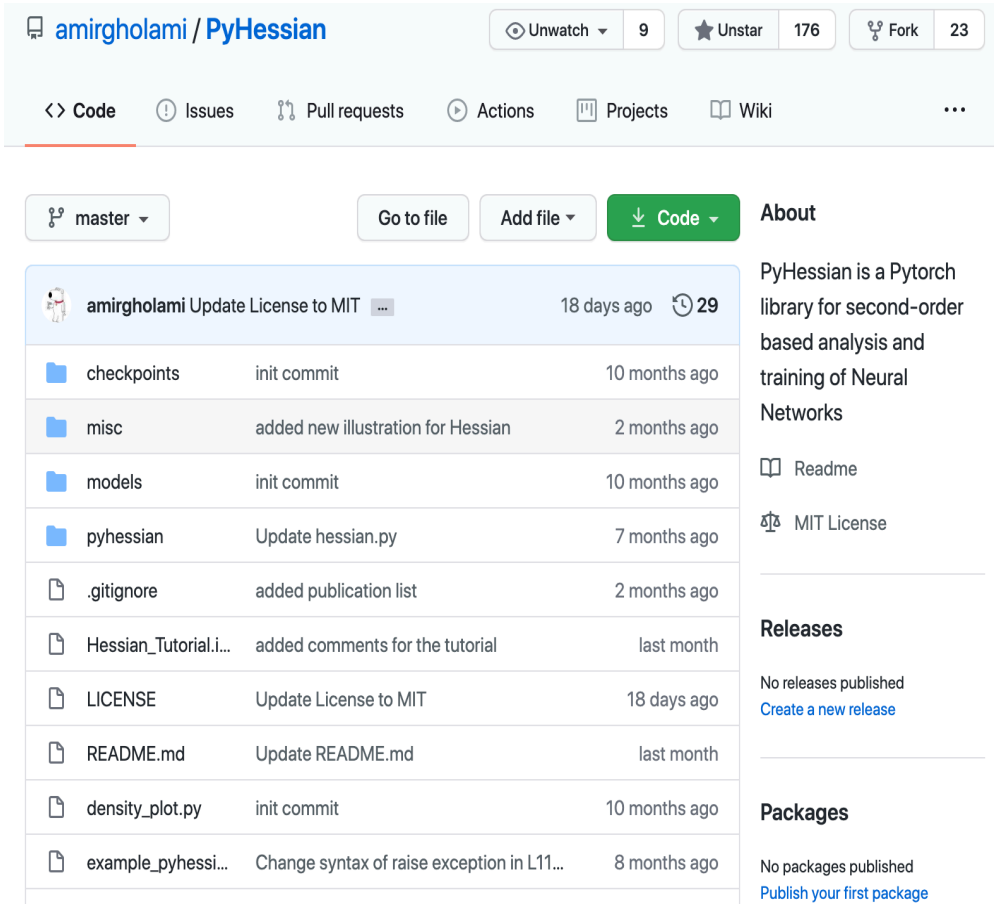
For a lot of applications, the explicit form of Hessian is not needed. The only requirement is the Hessian-vector product:

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



$$\text{Diag(H)} = \mathbb{E}[\text{z} \odot (\text{Hz})]$$
$$\text{s.t.} \quad \text{z} \sim \text{Rademacher(0.5)}$$

PyHessian Library



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

PyHessian enables:

- Top-k eigenvalues
- Hessian Diagonal
- Hessian Trace
- Estimated Spectral Distribution

Motivation

Choosing the right hyper-parameters for optimizing a NN training has become a **dark-art!**

Problems with existing first-order solutions:

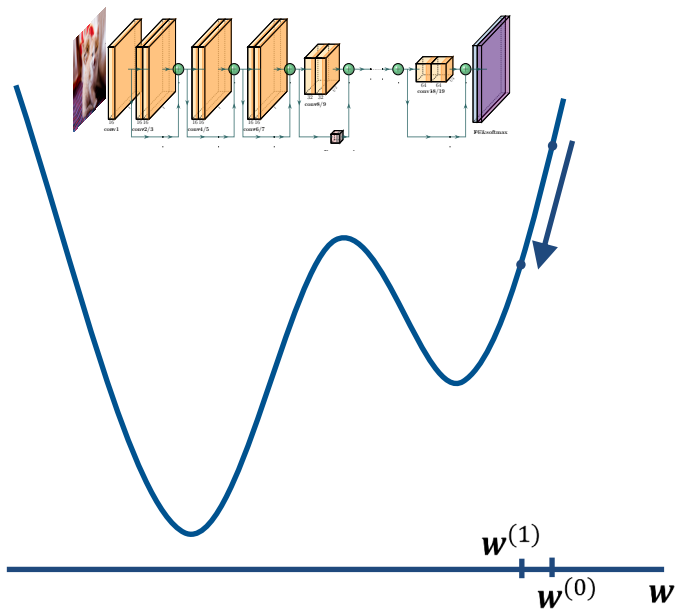
- Brute force hyper-parameters tuning
- Even the choice of the optimizer is a hyper-parameter!

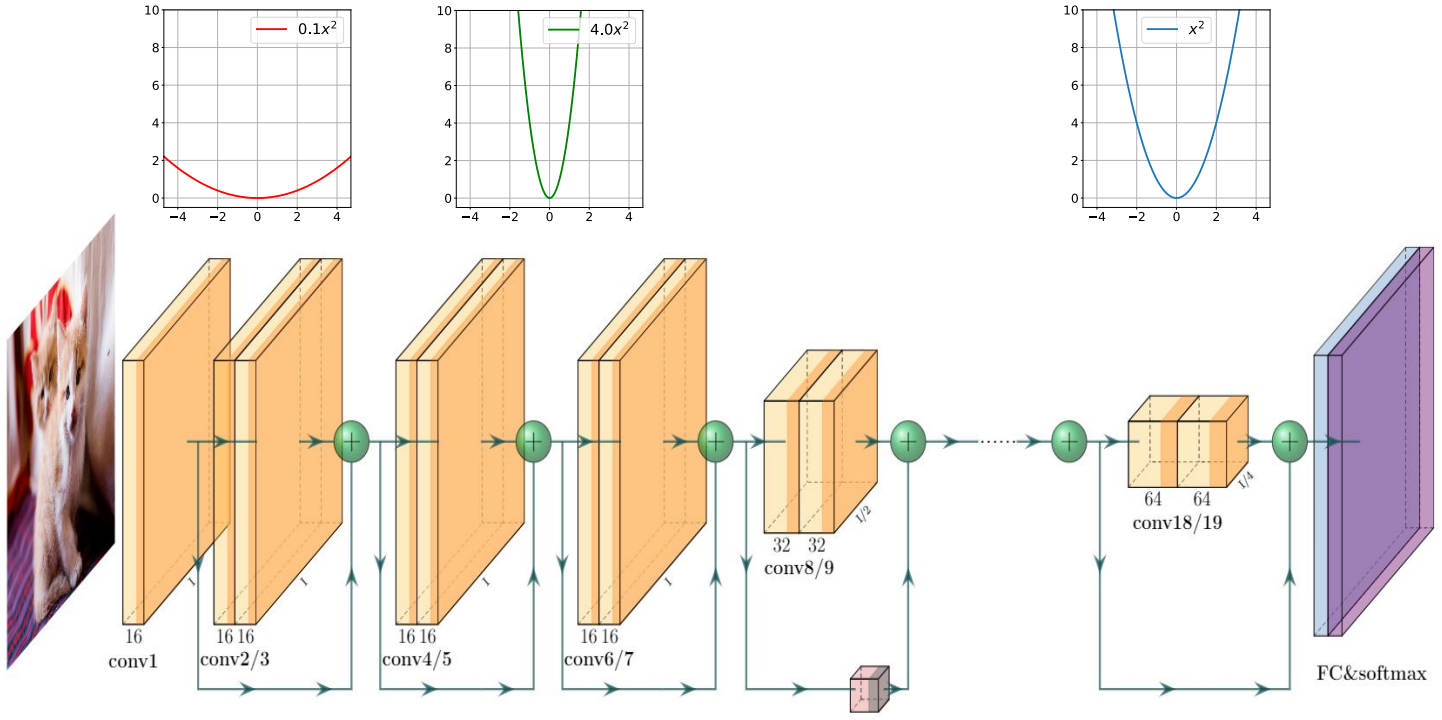
Task	CV	NLP	Recommendation System
Optimizer Choice	SGD	AdamW	Adagrad

SGD Based Training

$$\min_w E(w) = \frac{1}{N}\sum_{i=1}^{N} cost(w, x_i)$$

$$w^1 = w^0 - \frac{\lambda}{B}\sum_{i=1}^{B}\frac{\partial E_i(w^0)}{\partial w}$$



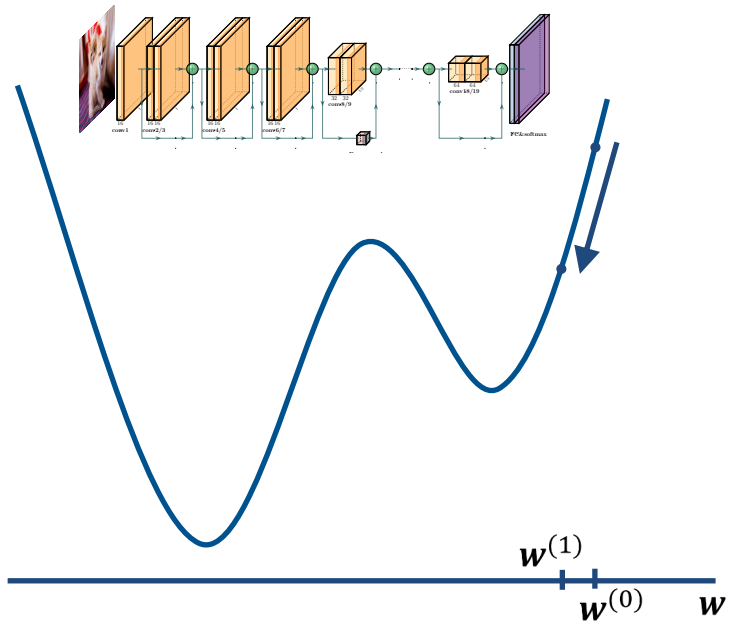


Hessian for DNNs

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

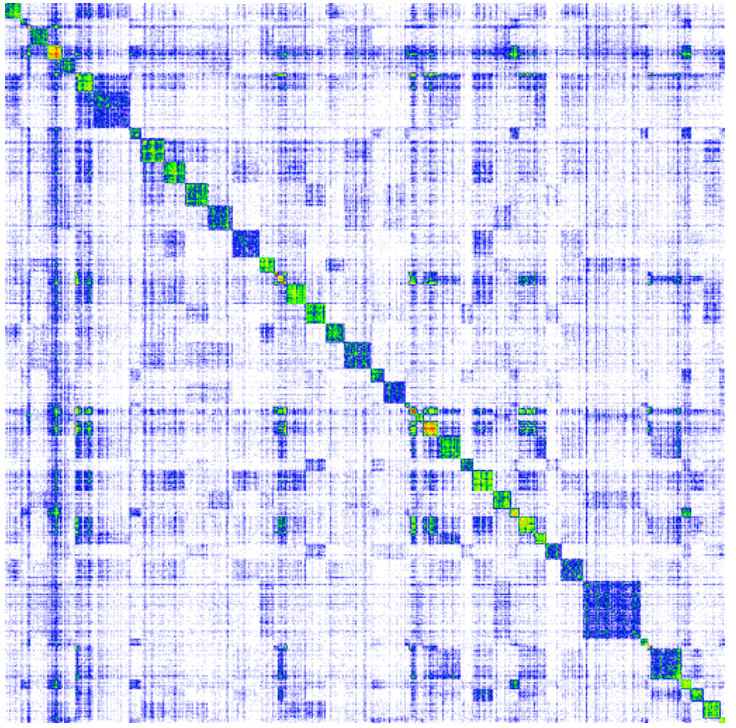
Gradient: $\frac{\partial E}{\partial w} \in \mathcal{R}^{|W|}$

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



$|W|$

$|W|$



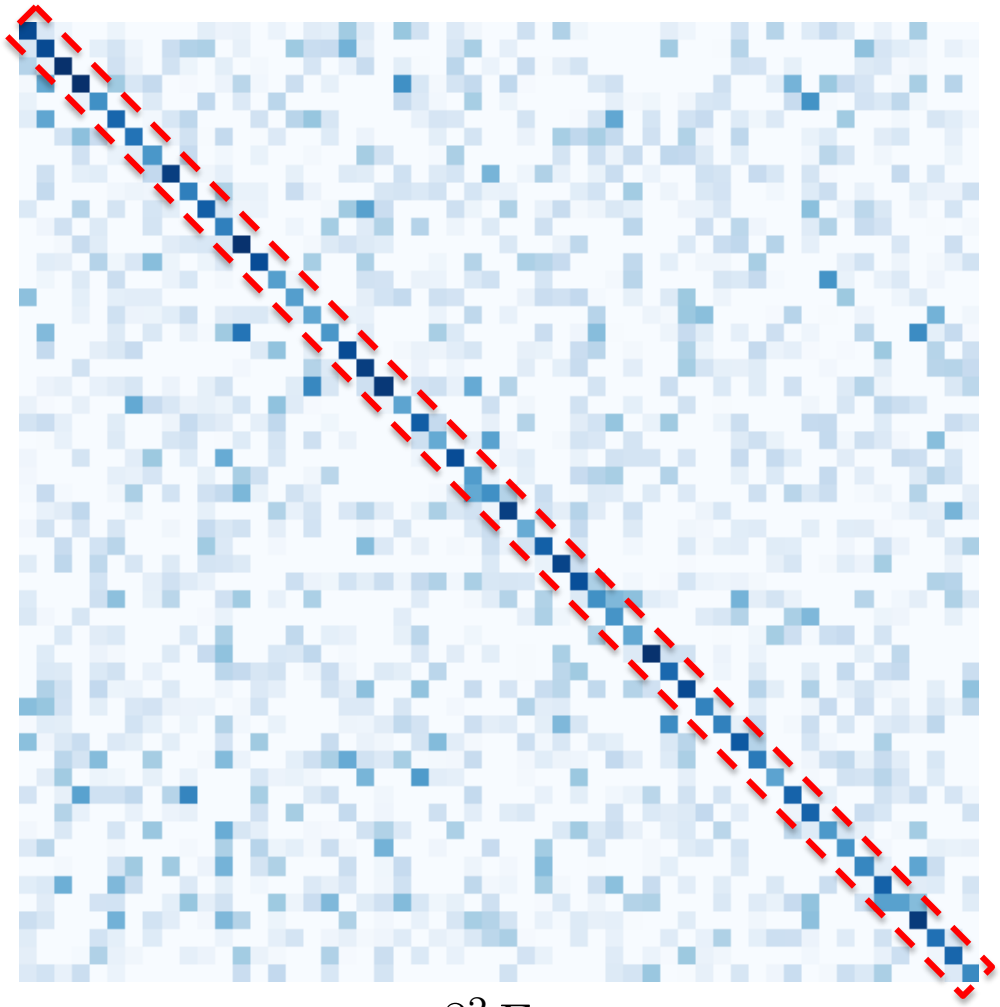
$|W|$

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

AdaHessian

Table 1: *Summary of the first and second moments used in different optimization algorithms for updating model parameters ($w_{t+1} = w_t - \eta m_t / v_t$). Here β_1 and β_2 are first and second moment hyperparameters.*

Optimizer	m_t	v_t
SGD [36]	$\beta_1 m_{t-1} + (1-\beta_1)\mathbf{g}_t$	1
Adagrad [16]	$\mathbf{g}_t$	$\sqrt{\sum_{i=1}^{t} \mathbf{g}_i \mathbf{g}_i}$
Adam [21]	$\frac{(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} \mathbf{g}_i}{1-\beta_1^t}$	$\sqrt{\frac{(1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} \mathbf{g}_i \mathbf{g}_i}{1-\beta_2^t}}$
RMSProp [40]	$\mathbf{g}_t$	$\sqrt{\beta_2 v_{t-1}^2 + (1-\beta_2)\mathbf{g}_t \mathbf{g}_t}$
ADAHESSIAN	$\frac{(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} \mathbf{g}_i}{1-\beta_1^t}$	$\left(\sqrt{\frac{(1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} \boldsymbol{D}_i^{(s)} \boldsymbol{D}_i^{(s)}}{1-\beta_2^t}}\right)^k$



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

Results on Machine Translation/Language Modeling

Only learning rate and space averaging block size are tuned for ADAHESSIAN

Higher BLEU score is better

Model	IWSLT14 small	WMT14 base
SGD	28.57 ± .15	26.04
AdamW [24]	35.66 ± .11	28.19
ADAHESSIAN	**35.79 ± .06**	**28.52**

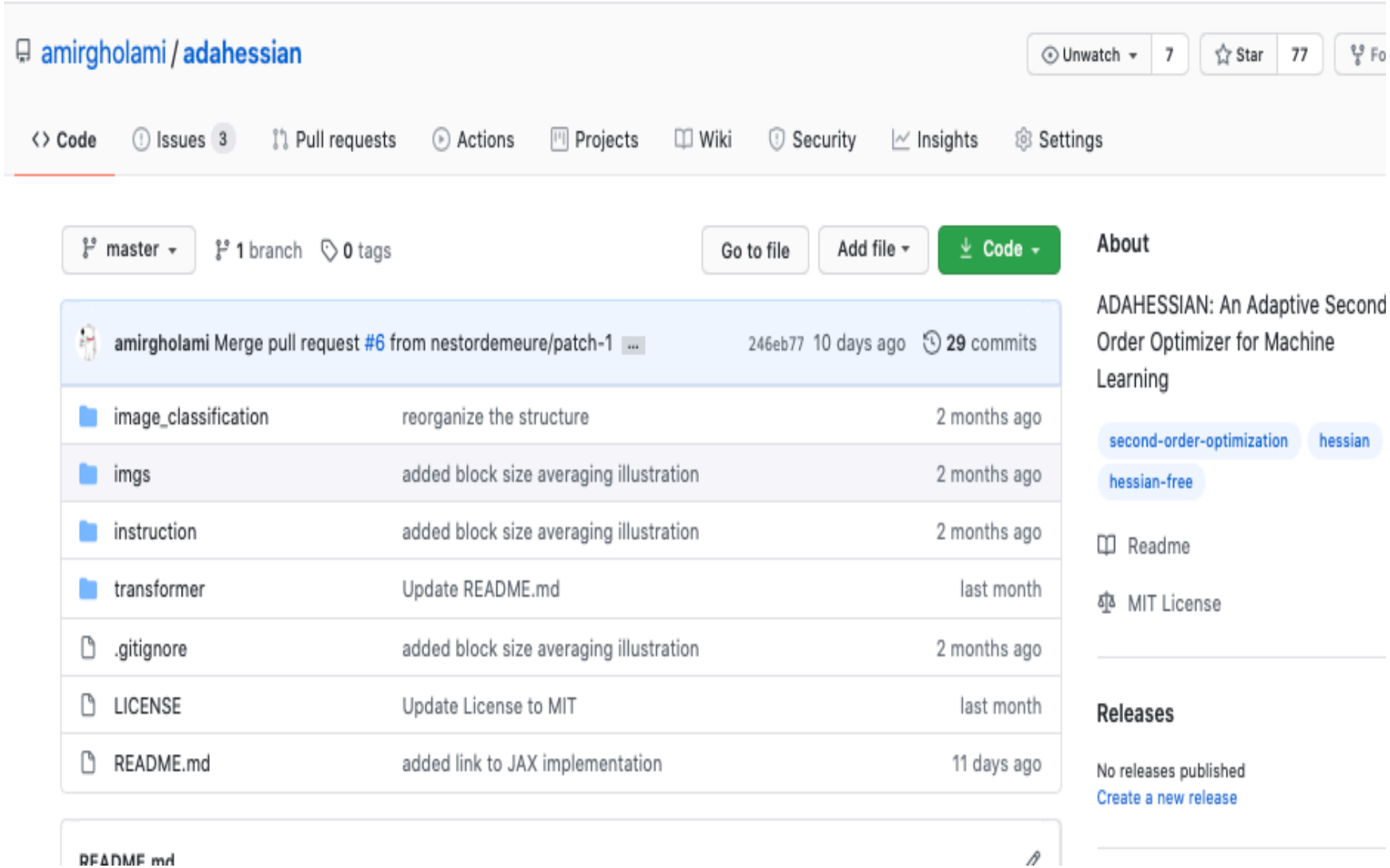
Only learning rate and space averaging block size are tuned for ADAHESSIAN

Lower perplexity is better

Model	PTB Three-Layer	Wikitext-103 Six-Layer
SGD	59.9 ± 3.0	78.5
AdamW [24]	54.2 ± 1.6	20.9
ADAHESSIAN	**51.5 ± 1.2**	**19.9**

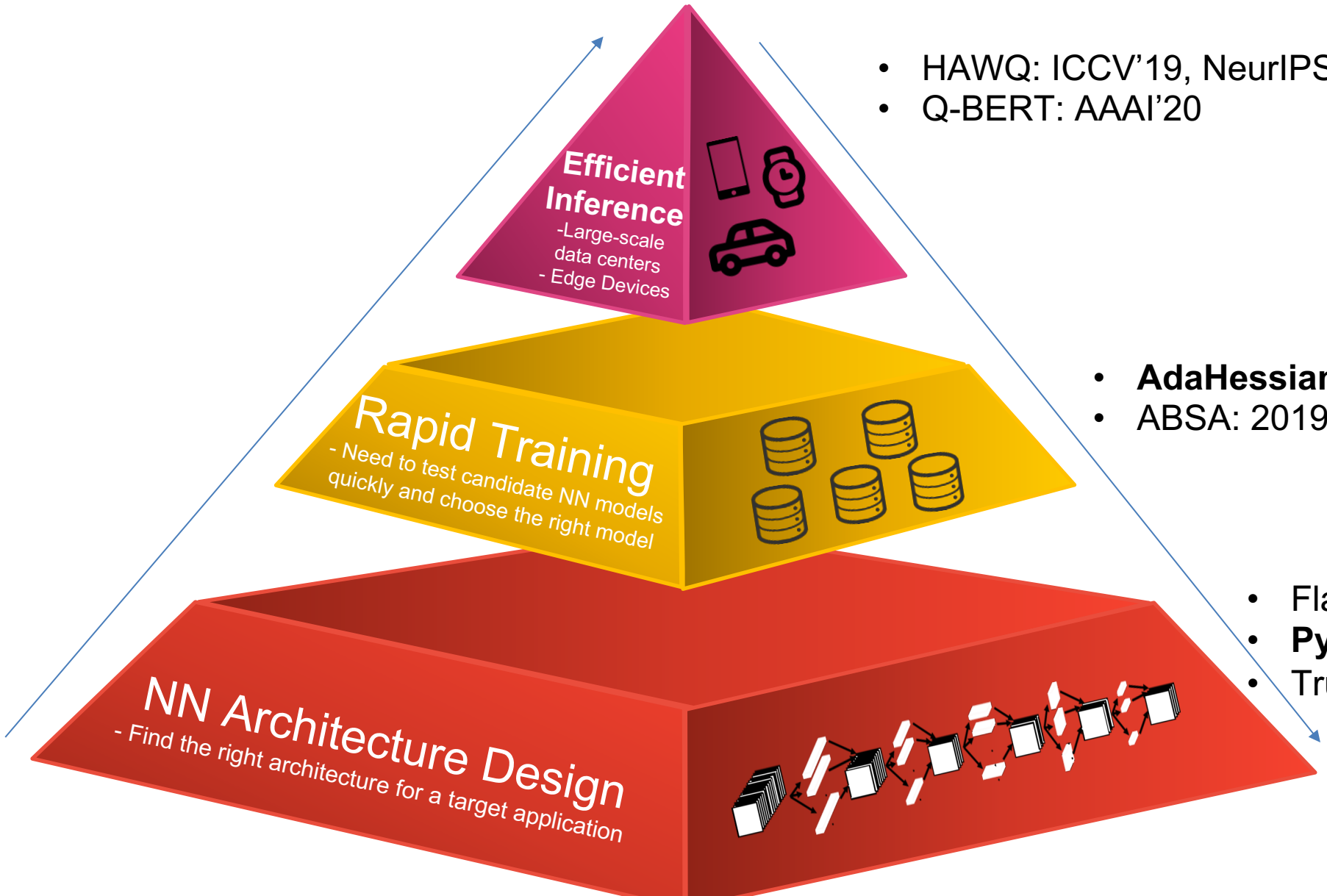
Z Yao*, A Gholami*, S Shen, M. Mustafa, K Keutzer, MW Mahoney, ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning, arXiv: 2006.00719

AdaHessian Library



AdaHessian: https://github.com/amirgholami/adahessian

Second Order Method for DNNs



- HAWQ: ICCV'19, NeurIPS'19
- Q-BERT: AAAI'20

- **AdaHessian: 2020**
- ABSA: 2019

- Flat/Sharp Local Minima: NeurIPS'18
- **PyHessian: ICML'20**
- Trust Region Adversarial: CVPR'20

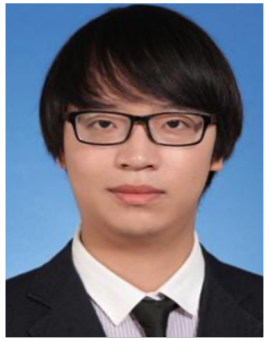
Thank You!

Please contact us if you have any questions:

{zheweiy, amirgh} @ berkeley.edu

Hessian tutorial: https://github.com/amirgholami/PyHessian/tree/master/pyhessian

AdaHessian tutorial: https://github.com/yaozhewei/analyze_ada_hessian




Berkeley
UNIVERSITY OF CALIFORNIA