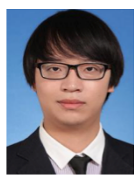
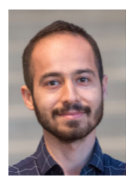
ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning

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


Berkeley
UNIVERSITY OF CALIFORNIA

riselab
UC Berkeley

Executive Summary

- We propose ADAHESSIAN, a novel second order optimizer that achieves new SOTA on various tasks:
  - CV: Up to 5.55% better results than Adam on ImageNet
  - NLP: Up to 1.8 PPL better results than AdamW on PTB
  - Recommendation System: Up to 0.032% better accuracy than Adagrad on Criteo
- ADAHESSIAN achieves these by:
  - Low cost Hessian approximation, applicable to a wide range of NNs
  - A novel exponential moving average which smooths Hessian noise across iterations
  - A new variance reduced estimate of the Hessian diagonal

Z Yao, A Gholami, S Shen, K Keutzer, MW Mahoney, ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning (arXiv:2006.00719)

ADAHESSIAN Motivation

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

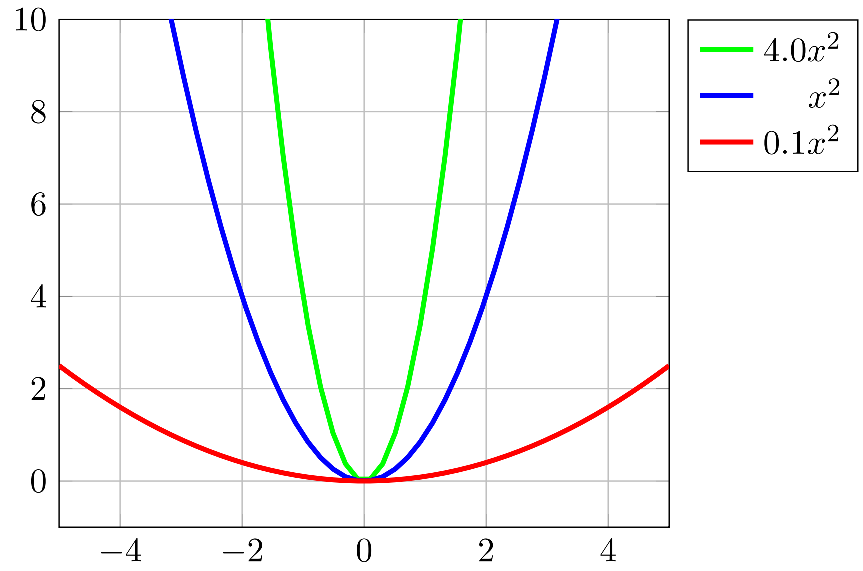
Problems with existing first-order solutions:

- Brute force hyper-parameter tuning
- No convergence guarantee unless taking *many* iterations
- Even the choice of the optimizer is a hyper-parameter!

Task	CV	NLP	Recommendation System
Optimizer Choice	SGD	AdamW	Adagrad

Second Order

- A major source of problems arise from the fact that first-order methods do not consider curvature information
- Question: Can we incorporate this information to guide training?



First and Second Order Methods

General parameter update formula: $\theta_{t+1} = \theta_t - \eta_t \Delta\theta_t$

First Order Method　　　　Second Order Method

$$\Delta\theta_t = g_t$$

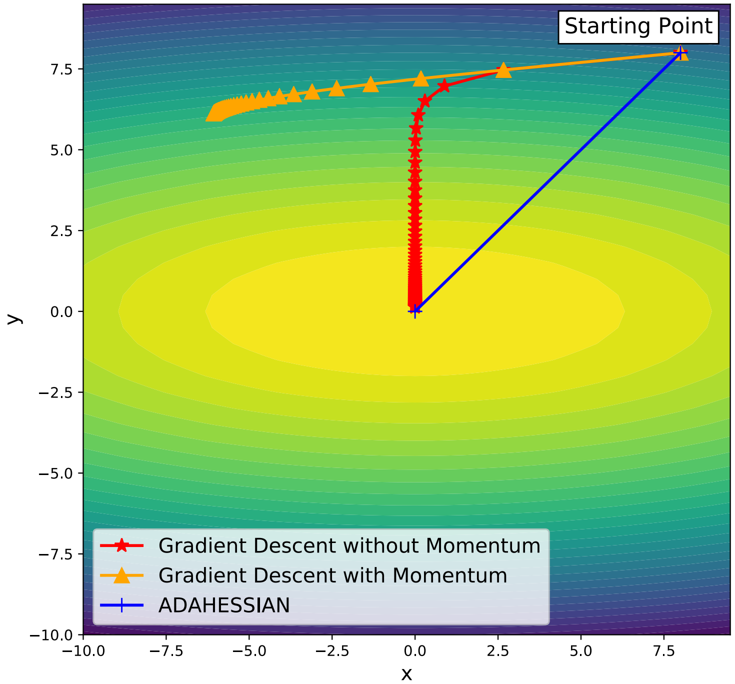
$$\Delta\theta_t = H_t^{-1} g_t$$

First and Second Order Methods

General parameter update formula: $\theta_{t+1} = \theta_t - \eta_t \Delta\theta_t$

First Order Method

$$\Delta\theta_t = H_t^0 g_t = g_t$$



The trajectory of optimizing:

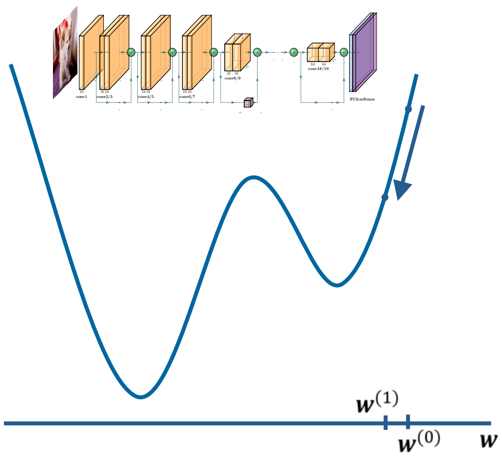
$$f(x, y) = x^2 + 10y^2$$

Second Order Method

$$\Delta\theta_t = H_t^{-1} g_t$$

Second Derivative (Hessian)

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

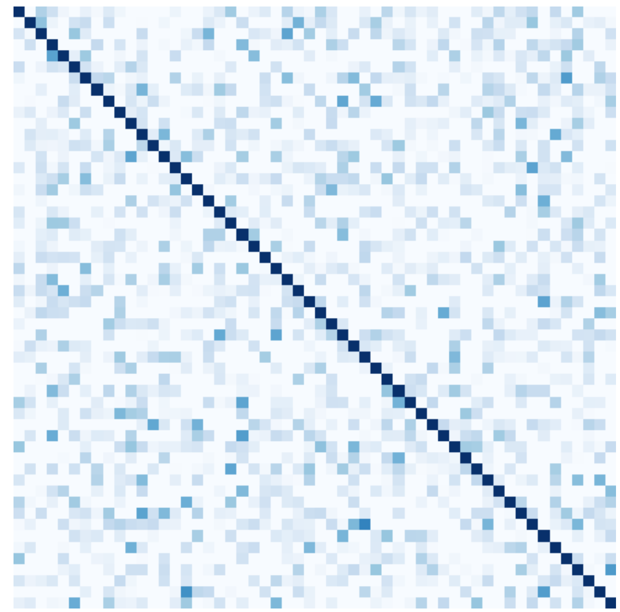


Gradient: $\frac{\partial E}{\partial w} \in \mathcal{R}^{|W|}$

$|W|$

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

$|W|$



$|W|$

Second Derivative (Hessian)

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

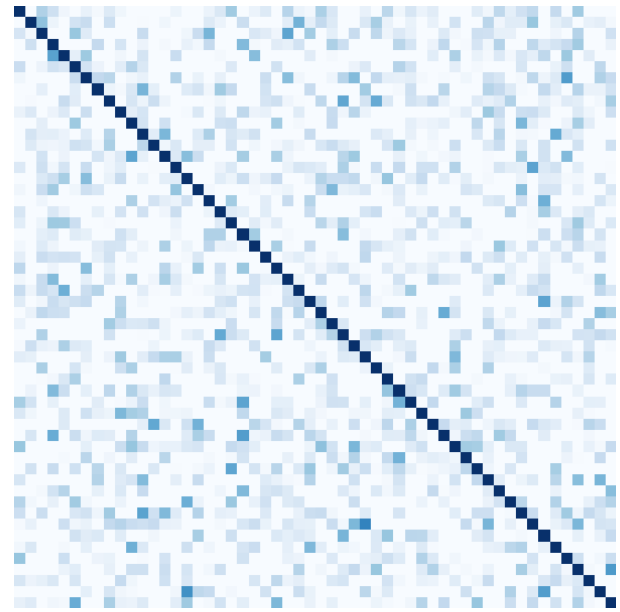
Forming the Hessian is computationally infeasible:

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

But what if we just approximate the Hessian?

$|W|$

$|W|$



$|W|$

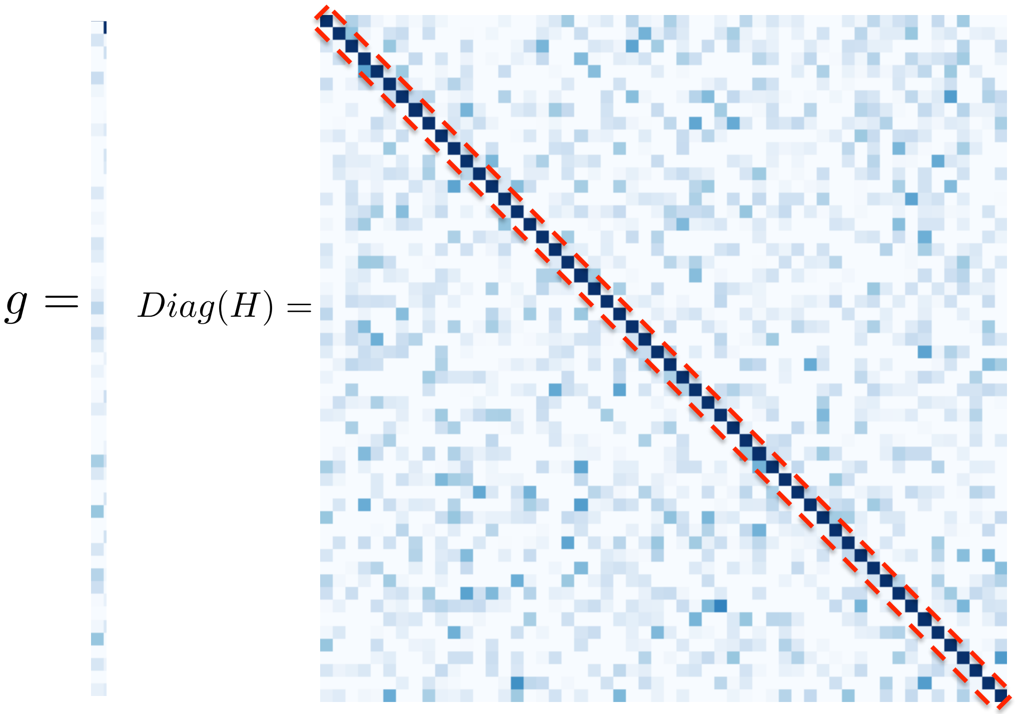
Using Hessian Diagonal

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



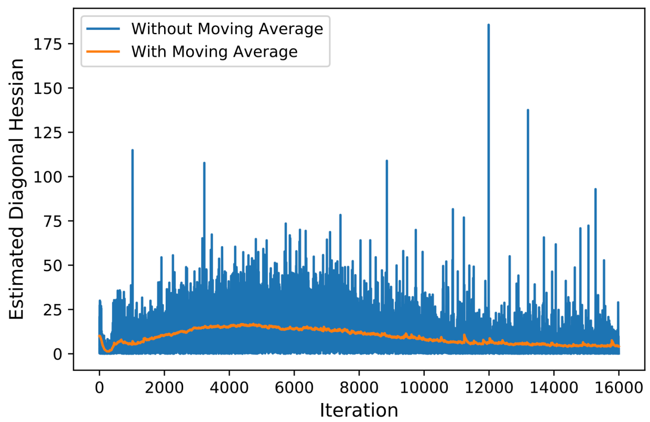
Variance Reduction

- For every iteration, the extra cost is one more backprop as compared to SGD method.
- How can we control the variance?

Variance Reduction

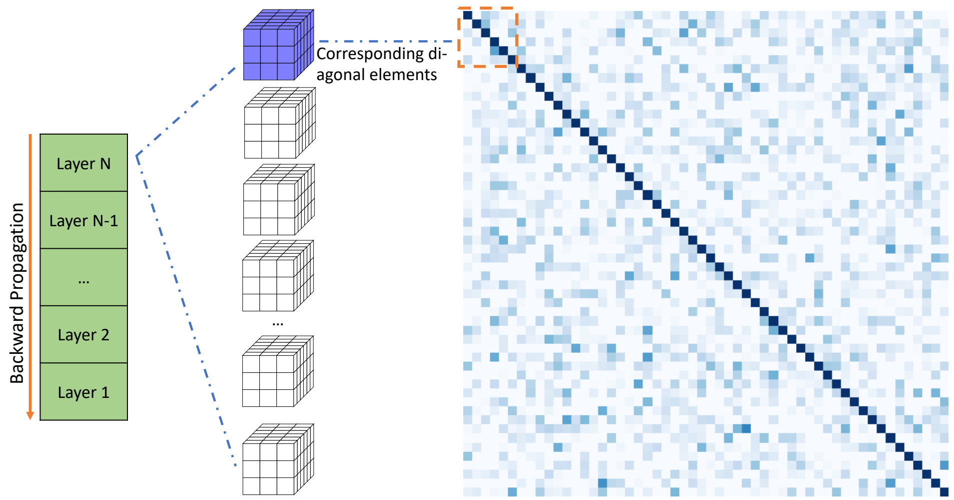
- For every iteration, the extra cost is one more backprop as compared to SGD method.
- How can we control the variance?
  - Incorporating momentum for both first and second order term:

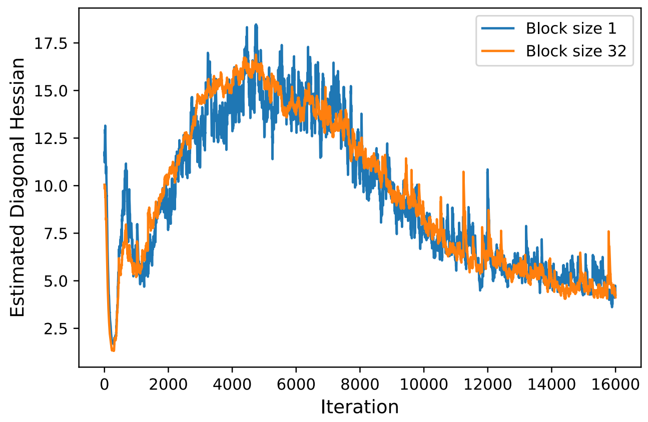
$$m_t = \frac{(1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i} g_i}{1-\beta_1^t}, \quad v_t = \sqrt{\frac{(1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i} D_i D_i}{1-\beta_2^t}}.$$



Variance Reduction

- For every iteration, the extra cost is one more backprop as compared to SGD method.
- How can we control the variance?
  - Incorporating momentum for both first and second order term.
  - Using blocks to compute the average diagonal approximation:





Important Points for Results

- What hyper-parameters we modified in the experiments:
  - Learning rate
  - Space averaging block size
- What hyper-parameters we did not modify in the experiments:
  - Learning rate schedule
  - Weight decay
  - Warmup schedule
  - Dropout rate
  - First and second order momentum coefficients, β_1 / β_2

Results on Image Classification

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

Higher is better

Dataset	Cifar10		ImageNet
	ResNet20	ResNet32	ResNet18
SGD [36]	92.08 ± 0.08	$\mathbf{93.14 \pm 0.10}$	70.03
Adam [19]	90.33 ± 0.13	91.63 ± 0.10	64.53
AdamW [22]	91.97 ± 0.15	92.72 ± 0.20	67.41
AdaHessian	$\mathbf{92.13 \pm 0.18}$	93.08 ± 0.10	**70.08**

Z Yao, A Gholami, S Shen, K Keutzer, MW Mahoney, ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning (arXiv:2006.00719)

Results on Machine Translation

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

Higher is better

Model	IWSLT14 small	WMT14 base
SGD	28.45	26.04
AdamW [22]	35.60	28.19
ADAHESSIAN	**35.87**	**28.52**

Z Yao, A Gholami, S Shen, K Keutzer, MW Mahoney, ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning (arXiv:2006.00719)

Results on Language Modeling

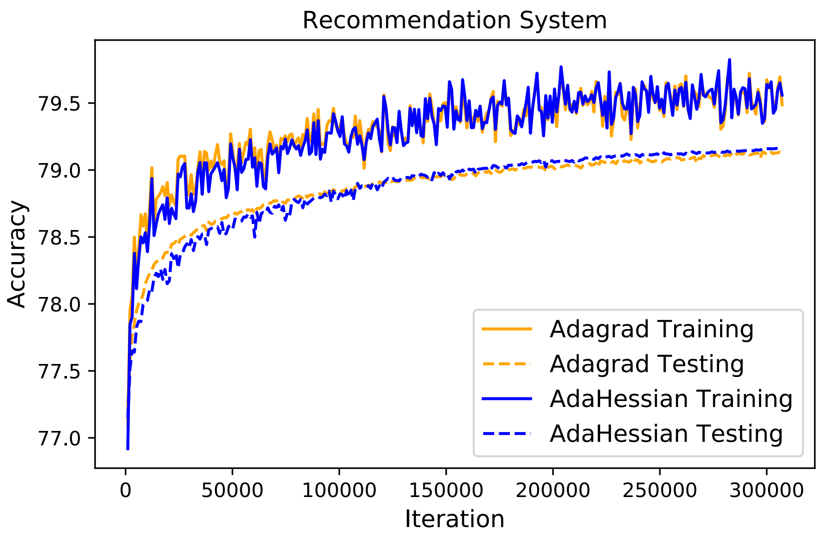
Only learning rate and space averaging block size are tuned for ADAHESSIAN
Lower is better

Model	PTB Three-Layer	Wikitext-103 Six-Layer
SGD	59.7	78.5
AdamW [22]	53.2	20.9
ADAHESSIAN	**51.4**	**19.9**

Z Yao, A Gholami, S Shen, K Keutzer, MW Mahoney, ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning (arXiv:2006.00719)

Results on Recommendation System

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



Criteo Ad Kaggle Dataset	Test Accuracy
AdaGrad	79.135
ADAHESSIAN	**79.167**

Z Yao, A Gholami, S Shen, K Keutzer, MW Mahoney, ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning (arXiv:2006.00719)

Speed Comparison with SGD

- An important advantage is the not only AdaHessian achieves SOTA results but its per iteration cost is comparable to SGD
- Computing Hessian diagonal at every step results in only **2x** overhead compared to SGD
  - This computation can be delayed to reduce this overhead down to **1.2x**

Delayed Steps	**0**	**1**	**2**	**3**	**4**
ResNet20 (Cifar-10)	$92.13 \pm .08$	$92.40 \pm .04$	$92.06 \pm .18$	$92.17 \pm .21$	92.16 ± 0.12
Transformer (IWSLT14)	35.87	35.90	35.89	35.75	35.75
Per-iteration Cost ($\times$SGD)	$2\times$	$1.5\times$	$1.33\times$	$1.25\times$	$1.2\times$

Z Yao, A Gholami, S Shen, K Keutzer, MW Mahoney, ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning (arXiv:2006.00719)

Conclusions

- We propose ADAHESSIAN, a novel second order optimizer that achieves new SOTA on various tasks:
  - CV: Up to 5.55% better results than Adam on ImageNet
  - NLP: Up to 1.8 PPL better results than AdamW on PTB
  - Recommendation System: Up to 0.032% better accuracy than Adagrad on Criteo
- ADAHESSIAN achieves these by:
  - Low cost Hessian approximation, applicable to a wide range of NNs
  - A novel exponential moving average which smooths Hessian noise across iterations
  - A new variance reduced estimate of the Hessian diagonal

Z Yao, A Gholami, S Shen, K Keutzer, MW Mahoney, ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning (arXiv:2006.00719)

Thank you!

There is also a poster on AdaHessian.

Please contact us if you have any questions:

{zheweiy, amirgh} @ berkeley.edu





riselab UC Berkeley

Berkeley UNIVERSITY OF CALIFORNIA

Extra

Robustness Study

- Robust to LR:

$\times$ Default LR	**0.5**	**1**	**2**	**3**	**4**	**5**
AdamW	35.48	35.60	35.28	34.78	13.75	0.5
AdaHessian	35.36	35.87	35.12	34.95	34.11	33.32

Result on IWSLT14.

Z Yao, A Gholami, S Shen, K Keutzer, MW Mahoney, ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning (arXiv:2006.00719)

Robustness Study

- Robust to LR
- Robust to Space Averaging Block Size:

Block Size	2	4	8	16	32	64	128
ADAHESSIAN	35.72	35.60	35.83	35.70	35.87	35.66	35.62

Result on IWSLT14. The BLEU score of AdamW is 35.60.

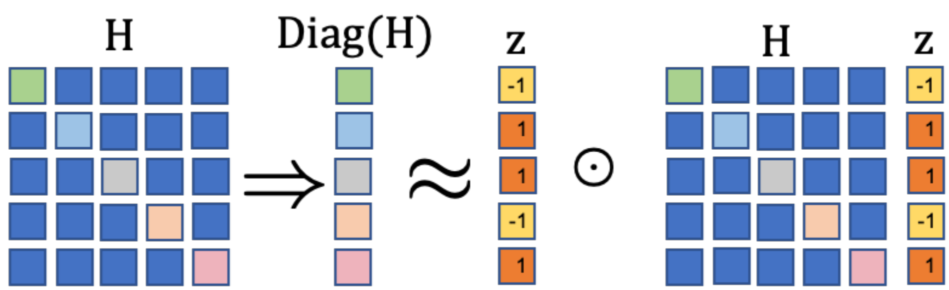
Z Yao, A Gholami, S Shen, K Keutzer, MW Mahoney, ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning (arXiv:2006.00719)

Diagonal Relaxation

- A possible solution is to use the diagonal of the Hessian:

$$\Delta\theta_t = H_t^{-1} g_t \longrightarrow \Delta\theta_t = D_t^{-1} g_t, \quad D_t = diag(H_t)$$

- How do we compute D_t?
  - using Hessian-Free Method and RNLA.



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

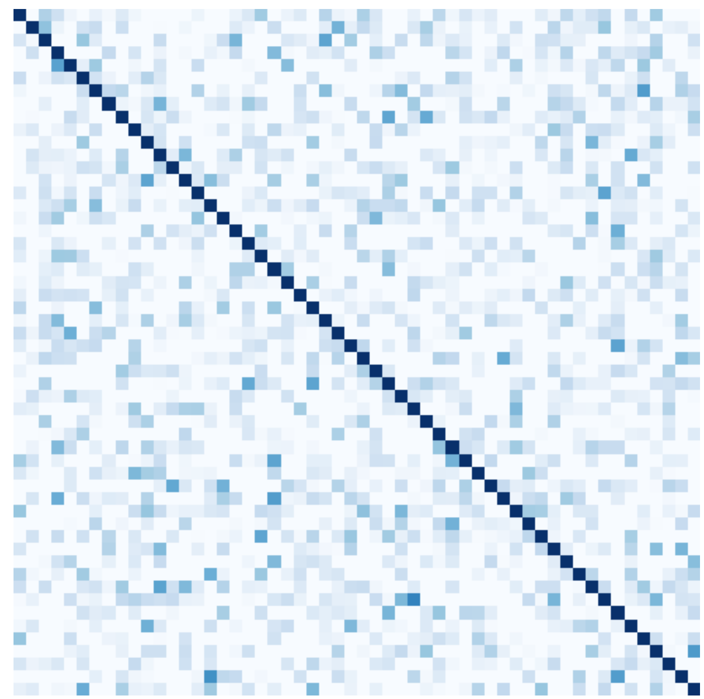
Is H^{-1} practical?

For ResNet50:
- # Parameters is 24M.
- |g| = 24M ~ 100 MB
- |H| = 24Mx24M ~ 2.4 PB

Can we:
- compute H?
- store H?
- compute H^{-1}?

$g =$

$H =$



Mixture Form

Instead of using fully first or second order method, the following formula is used: $\Delta\theta_t = H_t^{-k} g_t, \quad 0 \leq k \leq 1$

Mixture Form

Instead of using fully first or second order method, the following formula is used: $\Delta\theta_t = H_t^{-k} g_t, \quad 0 \leq k \leq 1$

- For convex problem, since $g_t^T H_t^{-k} g_t \geq 0$, $H_t^{-k} g_t$ is a descent direction.

Mixture Form

Instead of using fully first or second order method, the following formula is used: $\Delta\theta_t = H_t^{-k} g_t, \quad 0 \leq k \leq 1$

- For convex problem, since $g_t^T H_t^{-k} g_t \geq 0$, $H_t^{-k} g_t$ is a descent direction.
- For simple problems, computing H_t^{-k} is not a problem and it can be done by an eigen-decomposition.

Mixture Form

Instead of using fully first or second order method, the following formula is used: $\Delta\theta_t = H_t^{-k} g_t, \quad 0 \le k \le 1$

- For convex problem, since $g_t^T H_t^{-k} g_t \ge 0$, $H_t^{-k} g_t$ is a descent direction.
- For simple problems, computing H_t^{-k} is not a problem and it can be done by an eigen-decomposition.
- However, for large scale machine learning problems (e.g. DNNs), forming/storing Hessian are impractical.

Diagonal Relaxation

A possible solution is to use the diagonal of the Hessian:

$$\Delta\theta_t = D_t^{-k} g_t, \quad 0 \le k \le 1, \quad D_t = diag(H_t)$$

The remaining question is how to compute D_t ?

- Hessian-vector product:

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

- RandNLA:

$$D = diag(H) = \mathbb{E}[z \odot (Hz)], \quad z \sim Rademacher(0.5)$$

Diagonal Relaxation

A possible solution is to use the diagonal of the Hessian:

$$\Delta\theta_t = D_t^{-k} g_t, \quad 0 \leq k \leq 1, \quad D_t = diag(H_t)$$

The remaining question is how to compute D_t ?

- Hessian-vector product:

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

Diagonal Relaxation

A possible solution is to use the diagonal of the Hessian:

$$\Delta\theta_t = D_t^{-k} g_t, \quad 0 \leq k \leq 1, \quad D_t = diag(H_t)$$

The remaining question is how to compute D_t ?

- Hessian-vector product:

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

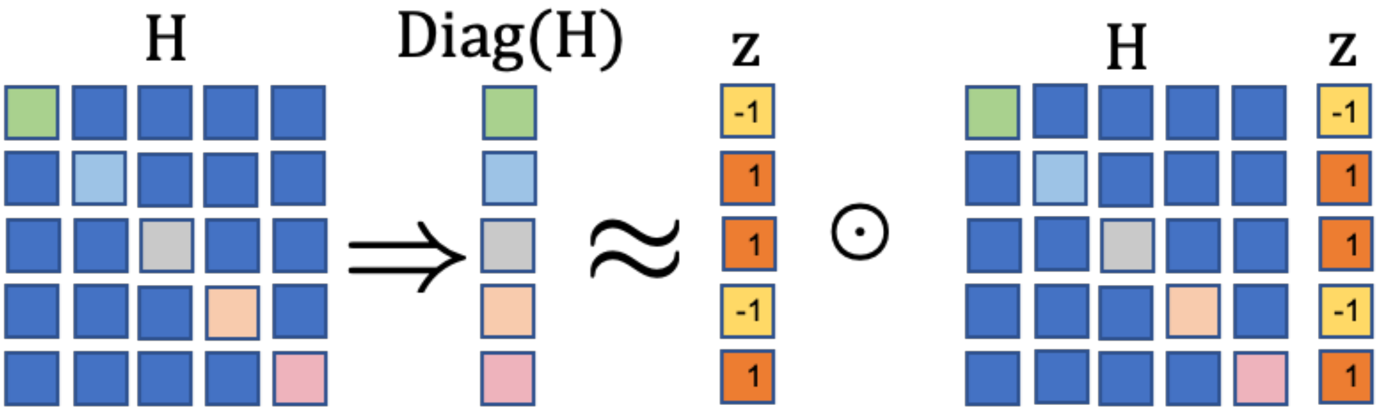
- Randomized numerical linear algebra (RNLA):

$$D = diag(H) = \mathbb{E}[z \odot (Hz)], \quad z \sim Rademacher(0.5)$$

Illustration of diagonal computation

RandNLA:

$$D = diag(H) = \mathbb{E}[z \odot (Hz)], \quad z \sim Rademacher(0.5)$$

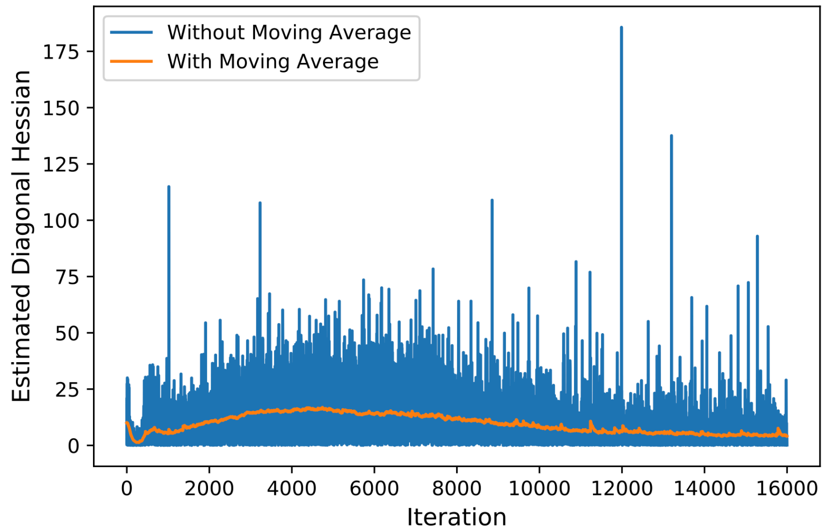


$$\mathrm{Diag(H)} = \mathbb{E}[\mathrm{z} \odot (\mathrm{Hz})]$$
$$\mathrm{s.t.} \quad \mathrm{z} \sim \mathrm{Rademacher(0.5)}$$

Variance Reduction

- Incorporating momentum for both first and second order term:

$$m_t = \frac{(1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i} g_i}{1-\beta_1^t}, \quad v_t = \sqrt{\frac{(1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i} D_i D_i}{1-\beta_2^t}}.$$



First and Second Order Methods

General parameter update formula: $\theta_{t+1} = \theta_t - \eta_t \Delta\theta_t$

First Order Method

Second Order Method

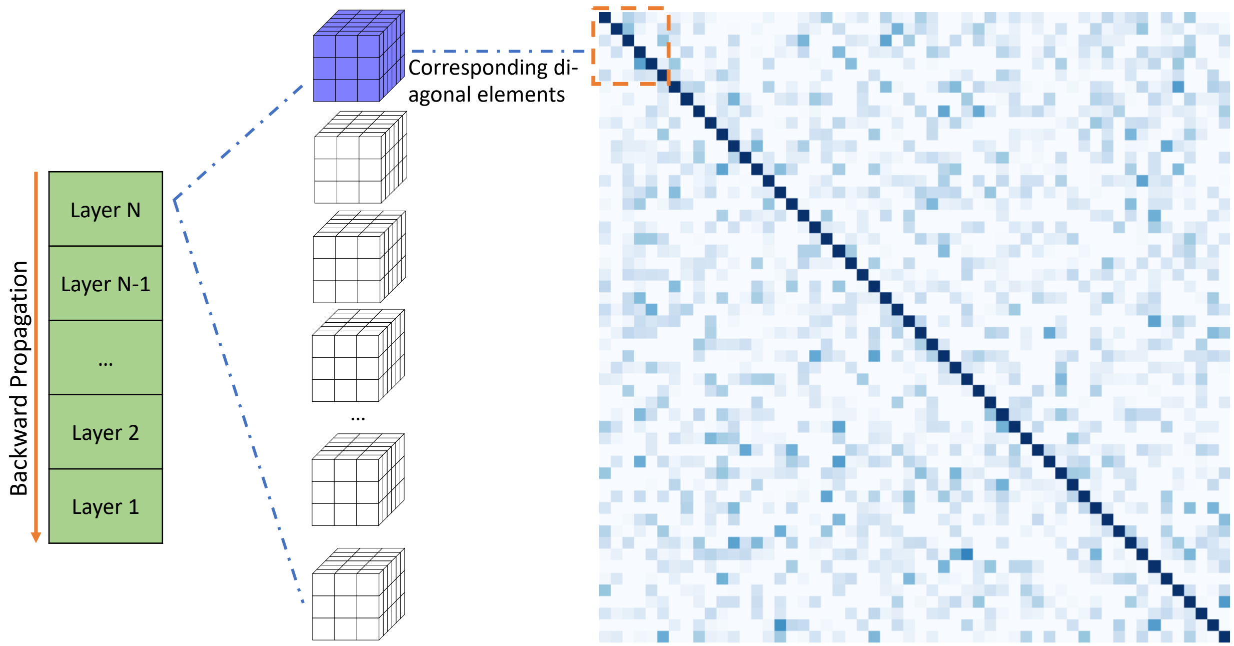
$\Delta\theta_t = H_t^0 g_t = g_t$

How about the middle part?

$\Delta\theta_t = H_t^{-1} g_t$

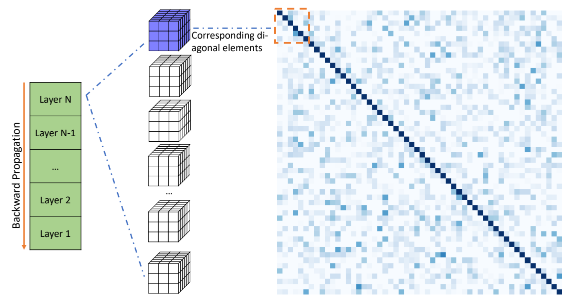
Variance Reduction

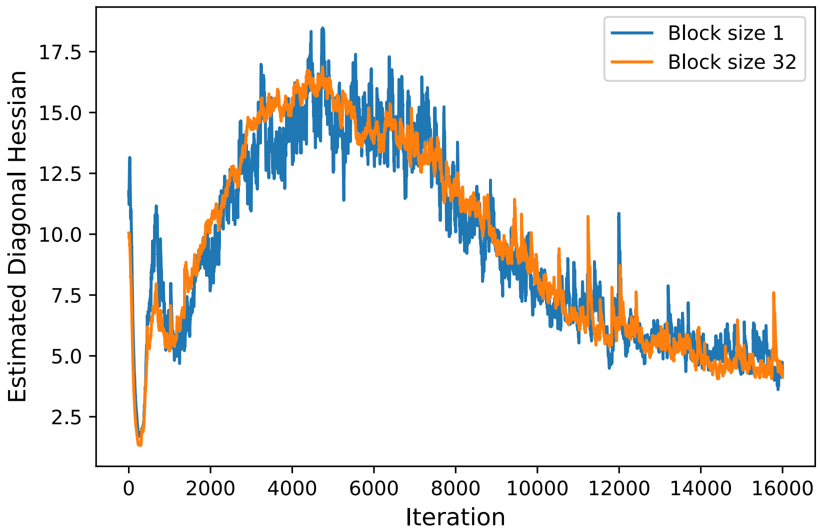
- Incorporating momentum for both first and second order term:
- Using blocks to compute the average diagonal approximation.



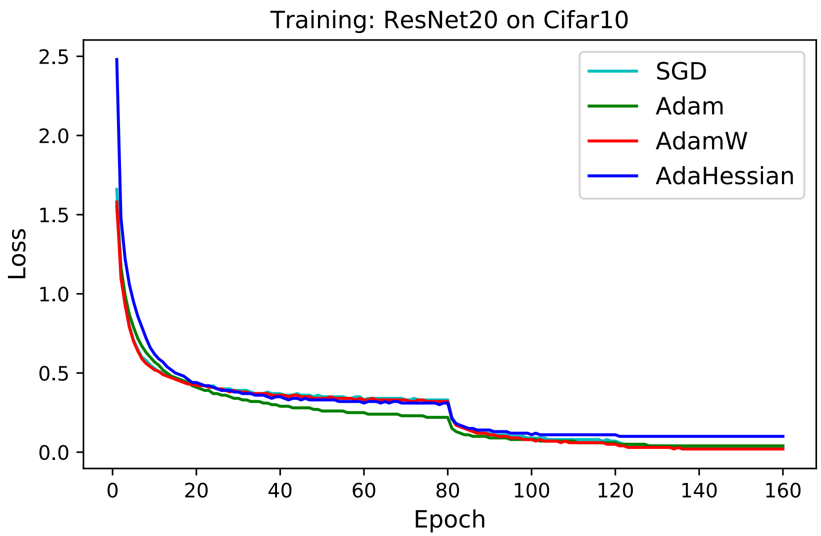
Variance Reduction

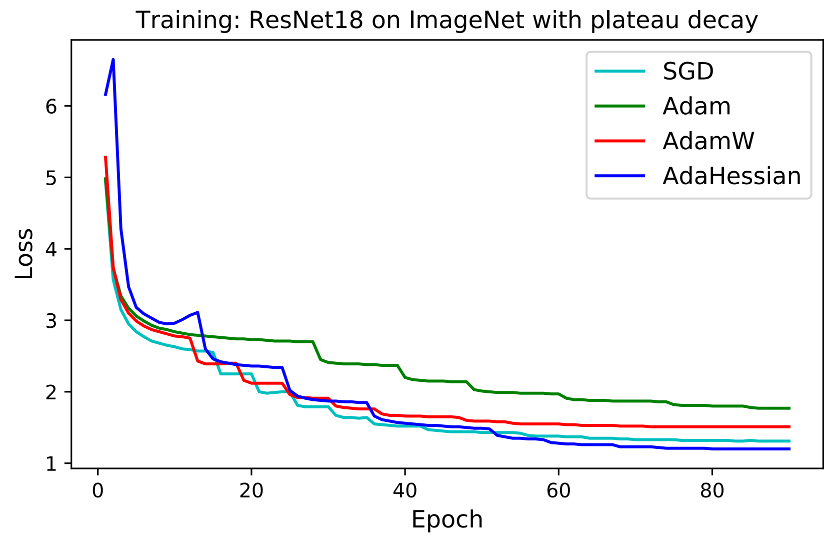
- Incorporating momentum for both first and second order term:
- Using blocks to compute the average diagonal approximation.



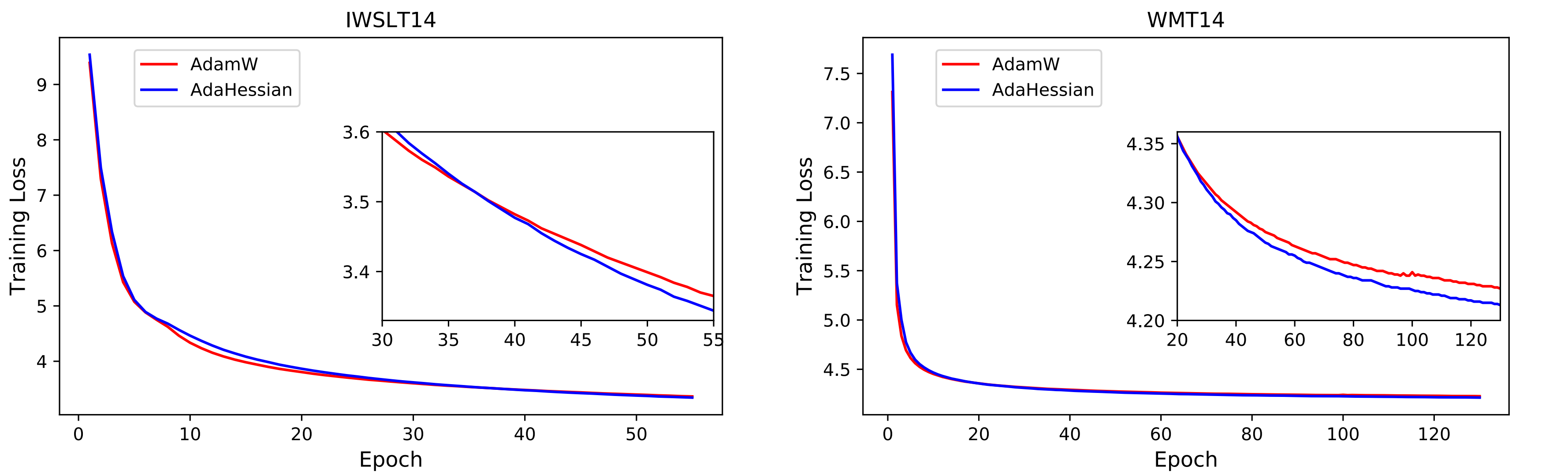


Results on Image Classification

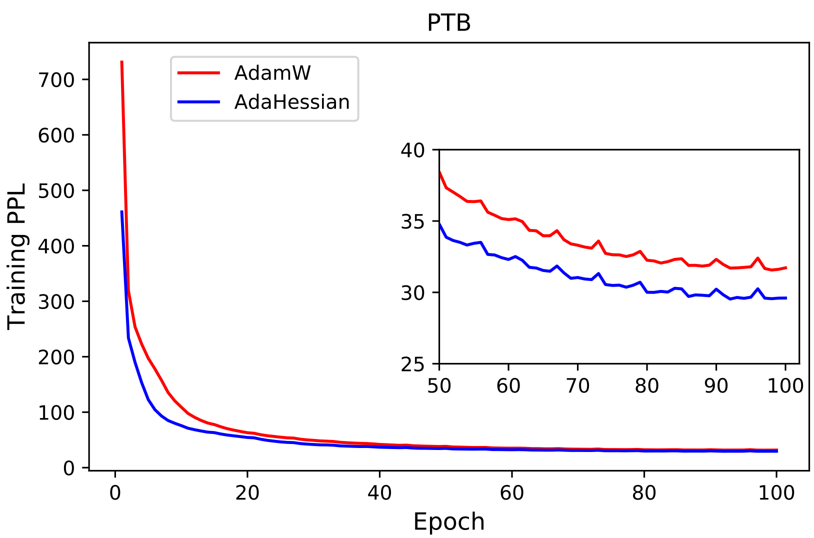


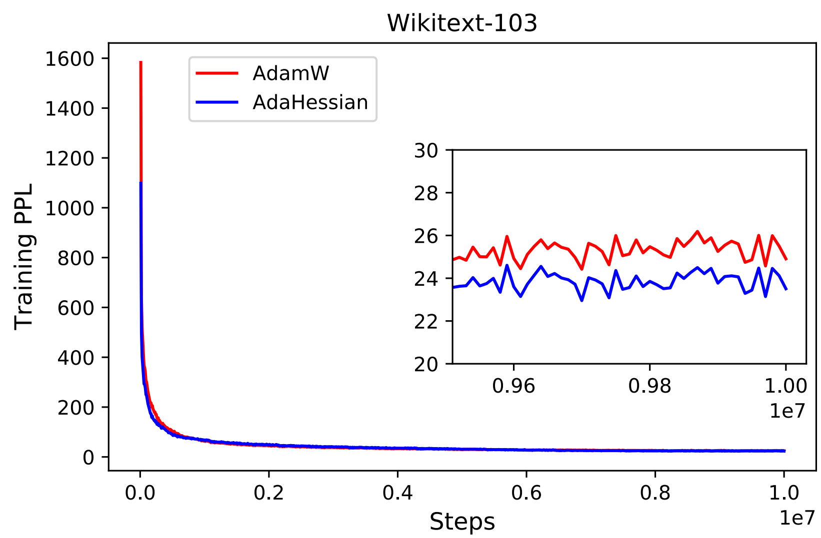


Results on Machine Translation



Results on Language Modeling





AdaHessian Algorithm

Algorithm 1: ADAHESSIAN

Require: Initial Parameter: θ_0
Require: Learning rate: η
Require: Exponential decay rates: β_1, β_2
Require: Block size: b
Require: Hessian Power: k
Set: $\bar{\mathbf{g}}_0 = 0$, $\bar{\boldsymbol{D}}_0 = 0$
for $t = 1, 2, \ldots$ **do** // Training Iterations
  $\mathbf{g}_t \leftarrow$ current step gradient
  $\boldsymbol{D}_t \leftarrow$ current step estimated diagonal Hessian
  Update m_t, v_t based on Eq. 10
  $\theta_t = \theta_{t-1} - \eta v_t^{-k} m_t$