ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning

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

Choosing the right hyper-parameter for optimizing a NN training has become a \textbf{dark-art}!

Problems with existing first-order solutions:

\begin{itemize}
  \item Brute force hyper-parameter tuning
  \item No convergence guarantee unless taking many iterations
  \item Even the choice of the optimizer is a hyper-parameter!
\end{itemize}

<table>
<thead>
<tr>
<th>Task</th>
<th>CV</th>
<th>NLP</th>
<th>Recommendation System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimizer Choice</td>
<td>SGD</td>
<td>AdamW</td>
<td>Adagrad</td>
</tr>
</tbody>
</table>
• 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
\[ \Delta \theta_t = g_t \]

Second Order Method
\[ \Delta \theta_t = H_t^{-1} g_t \]
First and Second Order Methods

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

First Order Method
\[ \Delta \theta_t = H_t^0 g_t = g_t \]

Second Order Method
\[ \Delta \theta_t = H_t^{-1} g_t \]

The trajectory of optimizing:
\[ f(x, y) = x^2 + 10y^2 \]
\[
\min_w E(w) = \frac{1}{N} \sum_{i=1}^{N} \text{cost}(w, x_i)
\]

Gradient: \( \frac{\partial E}{\partial w} \in \mathcal{R}^{\mid W\mid} \)

Hessian: \( \frac{\partial^2 E}{\partial w^2} \in \mathcal{R}^{\mid W\mid \times \mid W\mid} \)
Second Derivative (Hessian)

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?
Using Hessian Diagonal

Forming the Hessian is computationally infeasible:

For ResNet50 with 24M params Hessian is a matrix of size $24M \times 24M$

But what if we just approximate the Hessian?

Idea: Use Hessian diagonal

$g = \text{Diag}(H) = \ldots$
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?
  
  o Incorporating momentum for both first and second order term.
  
  o Using blocks to compute the average diagonal approximation:
Important Points for Results

• What hyper-parameters we modified in the experiments:
  o Learning rate
  o Space averaging block size

• What hyper-parameters we did not modify in the experiments:
  o Learning rate schedule
  o Weight decay
  o Warmup schedule
  o Dropout rate
  o First and second order momentum coefficients, $\beta_1 / \beta_2$
## Results on Image Classification

Only learning rate and space averaging block size are tuned for ADAHESSIAN. *Higher is better*

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cifar10 ResNet20</th>
<th>Cifar10 ResNet32</th>
<th>ImageNet ResNet18</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD [36]</td>
<td>92.08 ± 0.08</td>
<td><strong>93.14 ± 0.10</strong></td>
<td>70.03</td>
</tr>
<tr>
<td>Adam [19]</td>
<td>90.33 ± 0.13</td>
<td>91.63 ± 0.10</td>
<td>64.53</td>
</tr>
<tr>
<td>AdamW [22]</td>
<td>91.97 ± 0.15</td>
<td>92.72 ± 0.20</td>
<td>67.41</td>
</tr>
<tr>
<td><strong>ADAHESSIAN</strong></td>
<td><strong>92.13 ± 0.18</strong></td>
<td>93.08 ± 0.10</td>
<td><strong>70.08</strong></td>
</tr>
</tbody>
</table>

### Results on Machine Translation

Only learning rate and space averaging block size are tuned for ADAHESSIAN. **Higher is better**

<table>
<thead>
<tr>
<th>Model</th>
<th>IWSLT14 small</th>
<th>WMT14 base</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>28.45</td>
<td>26.04</td>
</tr>
<tr>
<td>AdamW [22]</td>
<td>35.60</td>
<td>28.19</td>
</tr>
<tr>
<td><strong>AdaHESSIAN</strong></td>
<td><strong>35.87</strong></td>
<td><strong>28.52</strong></td>
</tr>
<tr>
<td>Model</td>
<td>PTB</td>
<td>WikiText-103</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td>Three-Layer</td>
<td>Six-Layer</td>
</tr>
<tr>
<td>SGD</td>
<td>59.7</td>
<td>78.5</td>
</tr>
<tr>
<td>AdamW [22]</td>
<td>53.2</td>
<td>20.9</td>
</tr>
<tr>
<td>ADAHESSIAN</td>
<td>51.4</td>
<td>19.9</td>
</tr>
</tbody>
</table>

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

Results on Recommendation System

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

<table>
<thead>
<tr>
<th>Criteo Ad Kaggle Dataset</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaGrad</td>
<td>79.135</td>
</tr>
<tr>
<td>ADAHESSIAN</td>
<td>79.167</td>
</tr>
</tbody>
</table>

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 $1.2x$.

<table>
<thead>
<tr>
<th>Delayed Steps</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet20 (Cifar-10)</td>
<td>92.13 ± .08</td>
<td>92.40 ± .04</td>
<td>92.06 ± .18</td>
<td>92.17 ± .21</td>
<td>92.16 ± 0.12</td>
</tr>
<tr>
<td>Transformer (IWSLT14)</td>
<td>35.87</td>
<td>35.90</td>
<td>35.89</td>
<td>35.75</td>
<td>35.75</td>
</tr>
<tr>
<td>Per-iteration Cost ($\times$SGD)</td>
<td>$2x$</td>
<td>1.5×</td>
<td>1.33×</td>
<td>1.25×</td>
<td>1.2×</td>
</tr>
</tbody>
</table>
Conclusions

• We propose ADAHESSIAN, a novel second order optimizer that achieves new SOTA on various tasks:
  o CV: Up to 5.55% better results than Adam on ImageNet
  o NLP: Up to 1.8 PPL better results than AdamW on PTB
  o Recommendation System: Up to 0.032% better accuracy than Adagrad on Criteo

• ADAHESSIAN achieves these by:
  o Low cost Hessian approximation, applicable to a wide range of NNs
  o A novel exponential moving average which smooths Hessian noise across iterations
  o A new variance reduced estimate of the Hessian diagonal
Thank you!

There is also a poster on AdaHessian.

Please contact us if you have any questions:

{zheweiy, amirgh} @ berkeley.edu
Robustness Study

- Robust to LR:

<table>
<thead>
<tr>
<th>× Default LR</th>
<th>0.5</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdamW</td>
<td>35.48</td>
<td>35.60</td>
<td>35.28</td>
<td>34.78</td>
<td>13.75</td>
<td>0.5</td>
</tr>
<tr>
<td>ADAHESSIAN</td>
<td>35.36</td>
<td>35.87</td>
<td>35.12</td>
<td>34.95</td>
<td>34.11</td>
<td>33.32</td>
</tr>
</tbody>
</table>

Result on IWSLT14.

Robustness Study

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

<table>
<thead>
<tr>
<th>Block Size</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AdaHESSIAN</strong></td>
<td>35.72</td>
<td>35.60</td>
<td>35.83</td>
<td>35.70</td>
<td>35.87</td>
<td>35.66</td>
<td>35.62</td>
</tr>
</tbody>
</table>

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

Diagonal Relaxation

- A possible solution is to use the diagonal of the Hessian:
  \[ \Delta \theta_t = H_t^{-1} g_t \quad \rightarrow \quad \Delta \theta_t = D_t^{-1} g_t, \quad D_t = \text{diag}(H_t) \]

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

\[
\begin{align*}
\text{Diag}(H) &= \mathbb{E}[z \odot (Hz)] \\
\text{s.t.} &\quad z \sim \text{Rademacher}(0.5)
\end{align*}
\]
Is $H^{-1}$ practical?

For ResNet50:
- # Parameters is 24M.
- $|g| = 24M \sim 100$ MB
- $|H| = 24M \times 24M \sim 2.4$ PB

Can we:
- compute $H$?
- store $H$?
- compute $H^{-1}$?
Instead of using fully first or second order method, the following formula is used: $\Delta \theta_t = H_t^{-k} g_t$, $0 \leq k \leq 1$
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.
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.
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.
- 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:

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

The remaining question is how to compute $D_t$?

- Hessian-vector product:
  $$\frac{\partial g^T v}{\partial \theta} = \frac{\partial g^T}{\partial \theta} v + g^T \frac{\partial v}{\partial \theta} = \frac{\partial g^T}{\partial \theta} v = H v.$$

- RandNLA:
  $$D = \text{diag}(H) = \mathbb{E}[z \odot (H z)], \quad z \sim \text{Rademacher}(0.5)$$
A possible solution is to use the diagonal of the Hessian:

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

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:
  \[
  \frac{\partial g^T v}{\partial \theta} = \frac{\partial g^T}{\partial \theta} v + g^T \frac{\partial v}{\partial \theta} = \frac{\partial g^T}{\partial \theta} v = H v.
  \]

- Randomized numerical linear algebra (RNLA):
  \[ D = diag(H) = \mathbb{E}[z \odot (H z)], \quad z \sim \text{Rademacher}(0.5) \]
Illustration of diagonal computation

RandNLA:

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

Diag(H) = \mathbb{E}[z \odot (Hz)]

s.t. \quad z \sim \text{Rademacher}(0.5)
• Incorporating momentum for both first and second order terms:

\[ m_t = \frac{(1 - \beta_1) \sum_{i=1}^{t} \beta_1^{t-i} g_i}{1 - \beta_1^t}, \quad v_t = \sqrt{\frac{(1 - \beta_2) \sum_{i=1}^{t} \beta_2^{t-i} D_i D_i}{1 - \beta_2^t}}. \]
First and Second Order Methods

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

First Order Method
\[ \Delta \theta_t = H_t^0 g_t = g_t \]

Second Order Method
\[ \Delta \theta_t = H_t^{-1} g_t \]

How about the middle part?
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

IWSLT14

WMT14
Results on Language Modeling
AdaHessian Algorithm

**Algorithm 1: AdaHessian**

**Require:** Initial Parameter: $\theta_0$

**Require:** Learning rate: $\eta$

**Require:** Exponential decay rates: $\beta_1, \beta_2$

**Require:** Block size: $b$

**Require:** Hessian Power: $k$

Set: $\bar{g}_0 = 0, \bar{D}_0 = 0$

**for** $t = 1, 2, \ldots$ **do** // Training Iterations

$g_t \leftarrow$ current step gradient

$D_t \leftarrow$ current step estimated diagonal Hessian

Update $m_t, v_t$ based on Eq. 10

$\theta_t = \theta_{t-1} - \eta v_t^{-k} m_t$