Adaptive Methods for Nonconvex Optimization

Grant Block, Duke Kwon, Priya Sapra

Authors and Publication Date

- Authors
  - Shahnk K. Reddi
  - Manzil Zaheer
  - Devendra Sachan
  - Satyen Kale
  - Sanjiv Kumar
- Published in NeurIPS (Conference on Neural Information Processing Systems), 2018

The Flaws of Current First Order EMA Methods & Motivation

- Exponential moving Average (EMA) methods like RMSProp and Adam fail to converge in certain convex settings
    - Quickly forget gradient information
    - Current gradient is not informative of full problem
    - Easy to undershoot or overshoot the minimum

A Quick, Initial Comparison

Algorithm 1 ADAM

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, decay parameters $0 \leq \beta_1, \beta_2 \leq 1$, $\epsilon > 0$
Set $m_0 = 0$, $v_0 = 0$
for $t = 1$ **to** T **do**
  Draw a sample s_t from $\mathbb{P}$.
  Compute $g_t = \nabla \ell(x_t, s_t)$.
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t = v_{t-1} - (1 - \beta_2)(v_{t-1} - g_t^2)$
  $x_{t+1} = x_t - \eta_t m_t / (\sqrt{v_t} + \epsilon)$
end for

Algorithm 2 YOGI

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, parameters $0 < \beta_1, \beta_2 < 1$, $\epsilon > 0$
Set $m_0 = 0$, $v_0 = 0$
for $t = 1$ **to** T **do**
  Draw a sample s_t from $\mathbb{P}$.
  Compute $g_t = \nabla \ell(x_t, s_t)$.
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t = v_{t-1} - (1 - \beta_2) \text{sign}(v_{t-1} - g_t^2) g_t^2$
  $x_{t+1} = x_t - \eta_t m_t / (\sqrt{v_t} + \epsilon)$
end for

The Main Result: YOGI Method

Algorithm 2 YOGI

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, parameters $0 < \beta_1, \beta_2 < 1$, $\epsilon > 0$

Set $m_0 = 0$, $v_0 = 0$

for $t = 1$ **to** T **do**

  Draw a sample s_t from $\mathbb{P}$.

  Compute $g_t = \nabla \ell(x_t, s_t)$.

  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$

  $v_t = v_{t-1} - (1 - \beta_2)\text{sign}(v_{t-1} - g_t^2) g_t^2$

  $x_{t+1} = x_t - \eta_t m_t / (\sqrt{v_t} + \epsilon)$

end for

Overview

- Magnitude depends only on square of gradient

Advantages

- More control over learning weights
- O(d) memory, O(d) time complexity, O(1/δ^2) SFO
- The sign term gives previous gradient information influence on the direction, but not size of the second moment term

Drawbacks

- Performance compared to ADAM is nearly negligible

In-Depth

(Note that this slide is not shown in the video presentation. We provide more intuition of the differences of the algorithms here, for the interested viewer.)

Algorithm 1 ADAM

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, decay parameters $0 \le \beta_1, \beta_2 \le 1$, $\epsilon > 0$
Set $m_0 = 0, v_0 = 0$
for $t = 1$ **to** T **do**
  Draw a sample s_t from $\mathbb{P}$.
  Compute $g_t = \nabla \ell(x_t, s_t)$.
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t = v_{t-1} - (1 - \beta_2)(v_{t-1} - g_t^2)$
  $x_{t+1} = x_t - \eta_t m_t / (\sqrt{v_t} + \epsilon)$
end for

Algorithm 2 YOGI

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, parameters $0 < \beta_1, \beta_2 < 1$, $\epsilon > 0$
Set $m_0 = 0, v_0 = 0$
for $t = 1$ **to** T **do**
  Draw a sample s_t from $\mathbb{P}$.
  Compute $g_t = \nabla \ell(x_t, s_t)$.
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t = v_{t-1} - (1 - \beta_2)\text{sign}(v_{t-1} - g_t^2) g_t^2$
  $x_{t+1} = x_t - \eta_t m_t / (\sqrt{v_t} + \epsilon)$
end for

What makes the methods different?:

Suppose g_t is small, and v_{t-1} is large.

ADAM:
Then $(v_{t-1} - g_t^2)$ is also large, which means v_t is decreased by a fraction of a large amount. Then, our stepsize increases by a larger multiplicative factor of $\frac{1}{\sqrt{v_t}}$. However, we want small stepsize for small g_t.

YOGI:
Then $(v_{t-1} - g_t^2)$ is positive, but since g_t is small, we decrease v_t by a fraction of a small amount. Then, our stepsize increases by a smaller multiplicative factor of $\frac{1}{\sqrt{v_t}}$. That is, we increase the stepsize less compared to ADAM, when the gradient suddenly goes to 0, which is common in sparse settings.

There's also the case to consider when g_t is large, and v_{t-1} is slightly larger. In ADAM, v_t decreases slightly, and so the stepsize increases only slightly. In YOGI, v_t decreases, which means our step-size increases. YOGI here takes a more aggressive increase in step-size when it sees our gradients decrease.

Finally, when g_t is larger, and v_{t-1} is large, Then v_t increases slightly for ADAM, which means a reduction in step-size. In YOGI, we see v_t increases by a larger factor, which means a more aggressive reduction of step-size when we see the gradient increase.

Takeaways Compared to ADAM:

Yogi increases step-size in a slow, controlled manner when the previous curvature estimate is large, and our current stochastic gradient is small. (Reduce overshooting in sparse settings)

Yogi increases step-size aggressively when our estimate is large, and our current stochastic gradient is slightly smaller. (Improve undershooting)

Yogi decreases step-size aggressively when our current stochastic gradient is slightly larger than our large curvature estimate. (Possibly reduce overshooting)

The sign() in Yogi keeps the "direction" of how our step-size changes normally in Adam: We want large recent gradients to decrease step-size in future iterations, and small recent gradients to increase step-size in future iterations.

Time and Memory Complexity

	SFO Complexity (Convergence - We assume $b = \Theta(T)$)	Memory Costs	Computational Cost per Iteration (mini-batch = 1)
SGD	$O(\frac{1}{\delta^2})$	$O(d)$	$O(d)$
ADAM	$O(\frac{1}{\delta^2})$	$O(d)$	$O(d)$
YOGI	$O(\frac{1}{\delta^2})$	$O(d)$	$O(d)$

- Equivalent convergence & complexity
- Computational Cost for mini-batch > 1 is O(bd)
- SFO Complexity of ADAM & YOGI with large mini-batch is equivalent to SGD

Performance Guarantees and Algorithm Analysis

Corollary 4. *For x_t generated using* YOGI *with constant η (and parameters from Theorem 2), we have*

$$\mathbb{E}[\|\nabla f(x_a)\|^2] \leq O\left(\frac{1}{T} + \frac{1}{b}\right)$$

where x_a is an iterate uniformly randomly chosen from $\{x_1, \cdots, x_T\}$.

-Some assumptions: $1 - \beta_2 \leq \frac{\epsilon^2}{16G^2}$ *and* $\eta \leq \frac{\epsilon\sqrt{\beta_2}}{2L}$

Corollary 5. YOGI *with $b = \Theta(T)$ and constant η (and parameters from Theorem 2) has SFO complexity is $O(1/\delta^2)$ for achieving a δ-accurate solution.*

- Expected stationarity of the objective function is bounded by the number of max iterations T, and mini-batch size b.
- SFO Complexity of $O(1/\delta^2)$ is achieved when b is tightly bound to T, as T goes to infinity.

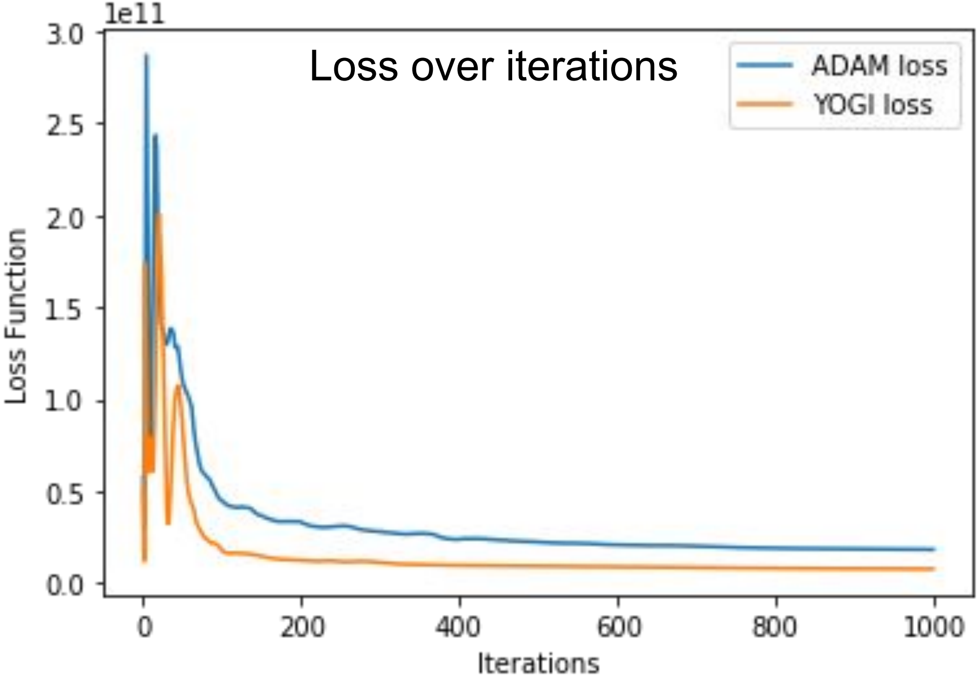
Interpretations: $\boxed{\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla f(x_t)\|^2]} \leq 2(\sqrt{2}G + \epsilon) \times \left[\frac{f(x_1) - f(x^*)}{\eta \boxed{T}} + \left(\frac{G\sqrt{1-\beta_2}}{\epsilon^2} + \frac{L\eta}{2\epsilon^2\sqrt{\beta_2}}\right)\frac{\sigma^2}{\boxed{b}}\right]$

(G - Upper bound of gradient per element)
(L - Lipschitz Constant, where the the loss f is L-smooth)
(σ^2 - Upper bound on variance of stochastic gradients)

In Convex Optimization: $f(x) - f(x^*)$

Empirical Evaluations - Multilogistic Regression on Fashion MNIST

$\beta 1 = 0.9$, $\beta 2 = 0.999$, ε=1e-8, $\alpha = 0.1$ $\eta = 1.0/(1.0+\alpha t)$

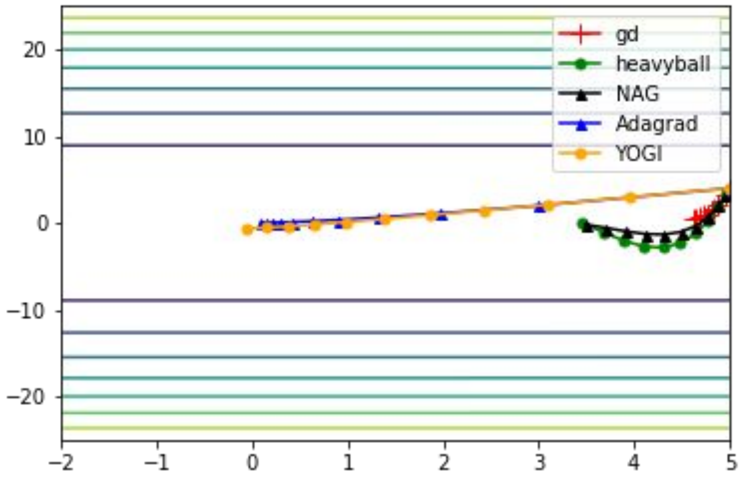


Empirical Evaluations - Adaptive Methods Comparisons

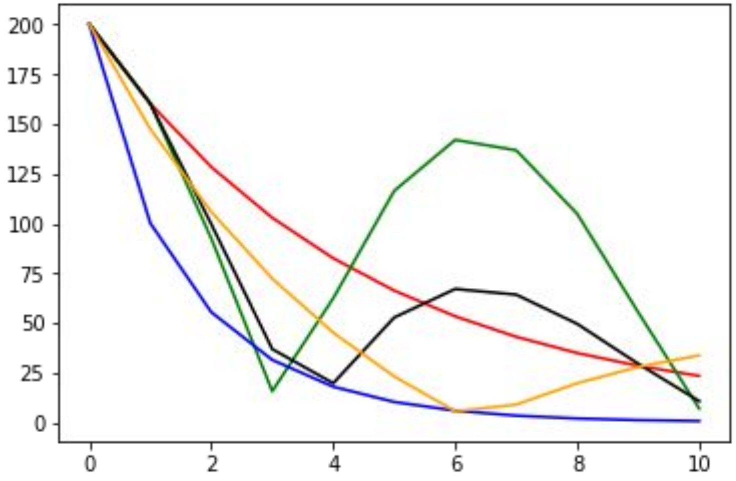
Objective Function $f = \frac{1}{2} * ||Dx||^2$

$\beta1 = 0.9$, $\beta2 = 0.999$, ε=1e-8, $\alpha = 1.0$, $\eta = 1.0/(1.0+\alpha t)$

Contours of f



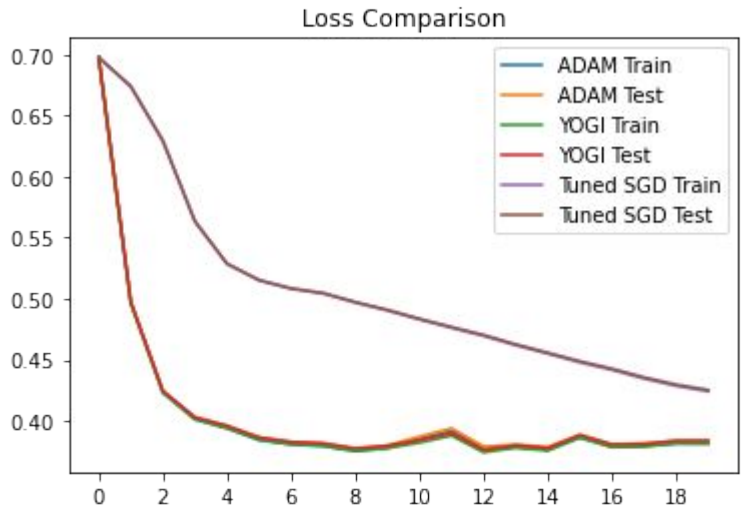
Gradient Size Over Iterations

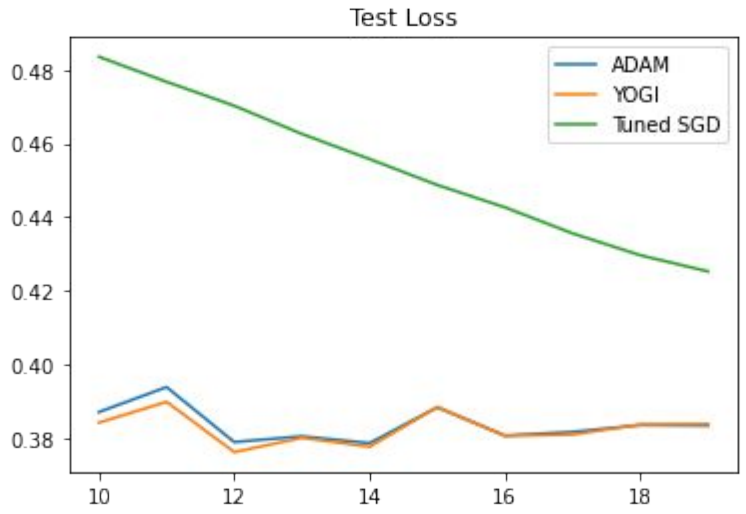


Empirical Evaluations - Autoencoder with Fashion MNIST

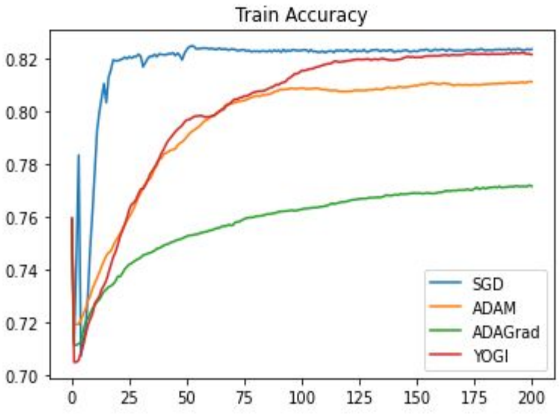
Architecture: Hidden Layer 1: ReLU
100 Neurons, Binary Cross Entropy Loss,
20 Iterations, minibatch = 25

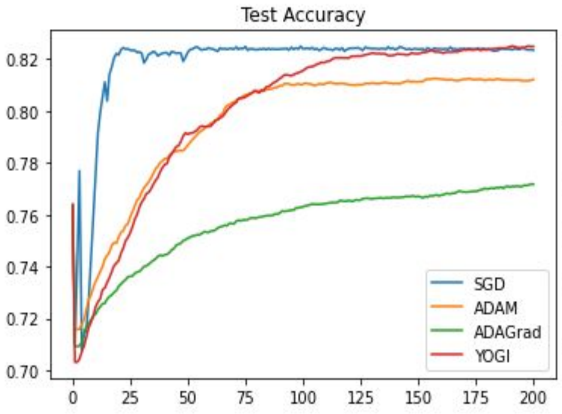
Optimizer	Train Loss	Test Loss
SGD $\eta = 10^{-4}$ $a = 10^{-3}$	0.42403	0.42523
ADAM $\beta_1 = 0.9$ $\beta_2 = 0.999$ $\epsilon = 10^{-8}$	0.38119	0.38337
YOGI $\beta_1 = 0.9$ $\beta_2 = 0.999$ $\epsilon = 10^{-3}$	0.38158	0.38379

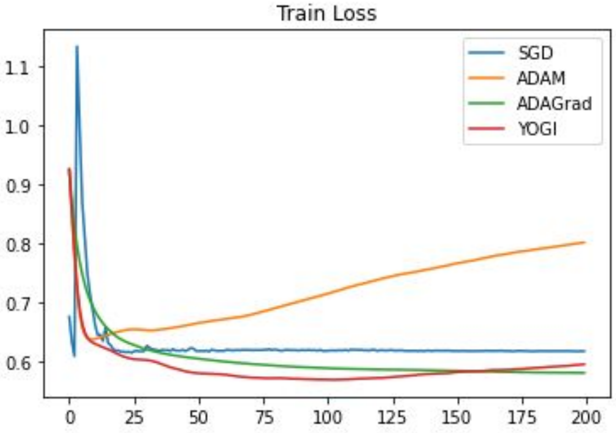




Empirical Evaluations - UCI Adult Classification SVM

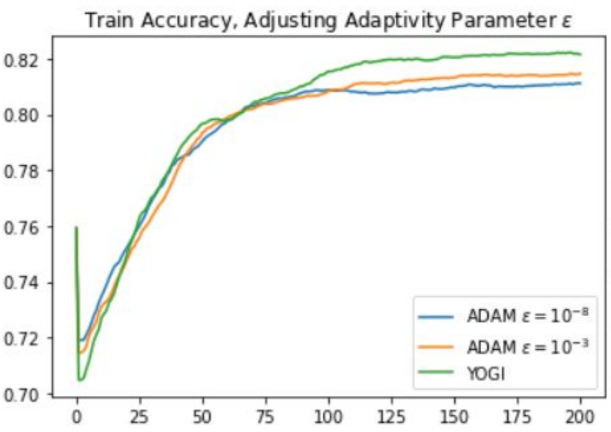


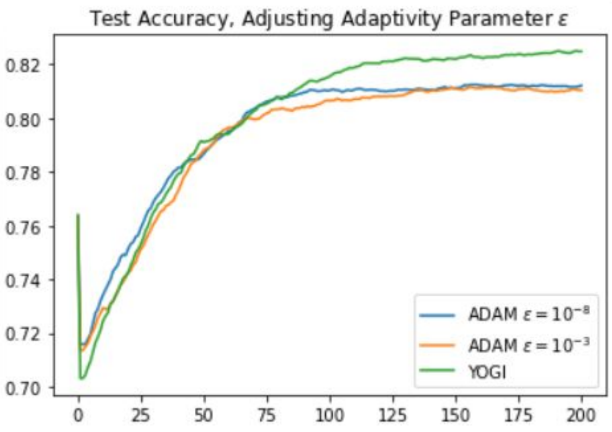


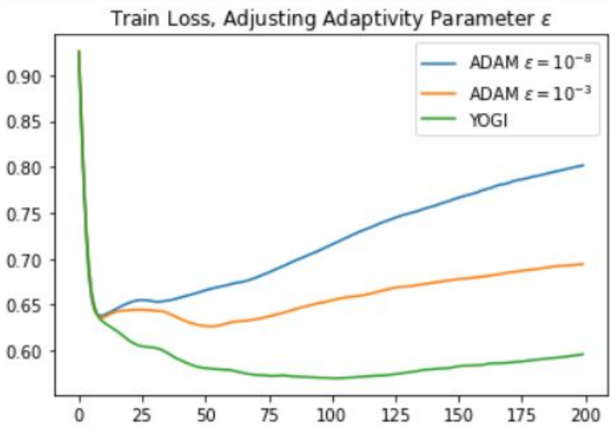


General Parameters Used: Mini-batch = 1000 Methods Below:	Test Accuracy	Train Accuracy	Train Loss
SGD $\alpha = \frac{\eta}{1+a \cdot t}$ $\eta = 0.1$	0.82341	0.82343	0.6178757
ADAGrad $\eta = 0.01$ $\epsilon = 10^{-8}$	0.77169	0.77150	0.581468
ADAM $\eta = 0.01$ $\beta_1 = 0.9$ $\beta_2 = 0.999$ $\epsilon = 10^{-8}$	0.81211	0.81112	0.80178
YOGI $\eta = 0.01$ $\beta_1 = 0.9$ $\beta_2 = 0.999$ $\epsilon = 10^{-3}$	**0.82482**	**0.82147**	**0.59588**

Empirical Evaluations - UCI Adult Classification SVM







General Parameters Used: Mini-batch = 1000 $\beta_1 = 0.9$ $\beta_2 = 0.999$ Methods Below:	Test Accuracy	Train Accuracy	Train Loss
ADAM $\epsilon = 10^{-8}$	0.81211	0.81112	0.80178
ADAM $\epsilon = 10^{-3}$	0.81039	0.81462	0.69412
YOGI $\epsilon = 10^{-3}$	**0.82482**	**0.82147**	**0.59588**

Algorithm 2 YOGI

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, parameters $0 < \beta_1, \beta_2 < 1$, $\epsilon > 0$
Set $m_0 = 0$, $v_0 = 0$
for $t = 1$ **to** T **do**
  Draw a sample s_t from $\mathbb{P}$.
  Compute $g_t = \nabla \ell(x_t, s_t)$.
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t = v_{t-1} - (1 - \beta_2)\,\mathrm{sign}(v_{t-1} - g_t^2) g_t^2$
  $x_{t+1} = x_t - \eta_t m_t / (\sqrt{v_t} + \boxed{\epsilon})$
end for

Use Cases

- The performance of YOGI is comparable to ADAM
- If convergence is an issue, YOGI is a more robust choice than ADAM
- SGD can outperform YOGI/ADAM, but this requires either a highly detailed knowledge of your problem space or a large amount of time tuning your hyperparameters.

Problem Set

- One of the main results of YOGI is that it can be shown that the bound on the stationary condition decreases linearly with increased batch size
  - Implement YOGI in your HW6 autoencoder.
  - Run your autoencoder with YOGI with minibatch sizes of 16, 32, 64, 128
  - Comment on results
- The paper also stated that the optimal YOGI parameters are $\beta 1 = 0.9$, $\beta 2 = 0.999$, $\varepsilon = 1e\text{-}3$
  - With a minibatch size of 128 in your autoencoder, run YOGI with those parameters, and some others of your choice
  - Discuss your observations, and say whether you agree with the paper that those are the optimal parameters.

References & Supplements

Sashank J. Reddi, Manzil Zaheer, Devendra Sachan, Satyen Kale, Sanjiv Kumar. Adaptive Methods for Nonconvex Optimization. NeurIPS, 2018.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond.arXiv preprint arXiv:1904.09237, 2019.

Dbouk Hassan. On The Convergence of SGD, ADAM & AMS-GRAD. ECE 543 Project Report, 2019.